SEKONIC

OPTICAL
MARK READER

Windows API
Reference Manual

Index

1. Introduction

This manual explains the use of API(*) for USB communications(ver.1.1/2.0) control of Sekonic's OMR SR-2300/5500/3500/6000/6500/1800.

The Microsoft Windows platform environment was utilized for developing efficient OMR control applications. Regarding further details on the platform environment, please refer to the "System Requirements" section.

In order to control OMR, it is necessary to have a personal computer (PC) connected to the OMR by a USB cable.

OMR will operate according to control commands from the PC.

Functions will be created for each command (and response), so that they can be utilized as API reference.

This document provides detailed explanation on each function, and explains the parameters that can be transferred to each function. Actual usage examples are included for reference as well.

API usage can be classified into the following 4 categories:

(1) System control : interface initialization (connection) etc.
(2) Setting Parameters : conduct necessary mark-sheet reading settings.
(3) Action Command : Commands for reading and disposing paper, etc.
(4) Data Reques : Collects data such as mark sheet reading results and OMR condition, etc.

Each of the sample functions mentioned above are simple, but do include how to declare variables and how to augment the utility of the functions.

Please also refer to the OMR users guide.

(*)API=Application Programming Interface

2. Provided Files

The API that is necessary for controlling Sekonic's OMR through USB communication (Ver.1.1/2.0) is provided by the following files.

(1) OMRAPI.DLL
Main file - dynamic library, that provides API functions
(2) OMRAPI.h
Heading file for OMRAPI.
(3) OMRAPI.LIB
Insertion library that can be linked to OMRAPI.DLL when this is being used.

Below are USB device driver.

(4) Sksr6500.sys
A USB device driver used only for Sekonic's OMR.
(5) SkSr6500.sys / SkSr1800.inf
Information file for the device driver

3. System Requirements

3.1 System Requirements

Sekonic's OMR's USB connection (ver. 1.1) operates in the following environment.

(1) Operating System (*1)
Microsoft Windows 2000 Professional
Microsoft Windows XP (32-bit)
Microsoft Windows Vista (32-bit)
Microsoft Windows 7 (32-bit)

(2) Hard Disk (HD) Capacity
At least 5 MB available memory space.

(3) Memory
At least 8 MB available memory when using Windows.

(4) Device Driver
The included device driver must be installed.

(*1) When the Microsoft finished the support for the system, our support for these operation system will be completed.

3.2 Development Tools

This DLL uses only Windows standard API.
Other libraries are not used, therefore, it can be used from within common Windows application development tools (Visual C++, Visual Basic, Borland C++ Builder, etc.).

When using DLL from the development tool, please refer to the operation manual of each development tool and/or reference documents. You can also read the various tool manuals and references in order to use DLL from within the Development tools. It is also possible to document DLL call routine within the code using the Win32API LoadLibrary function. For further details, please refer to Win32API reference materials.

When actually writing code, note that OMRAPI.h must be included.

When executing a compiled application, it must be located in the same folder as the OMRAPI.DLL or must be located in a folder with a pass.

4. API Function Reference

4.1 Constant

API return value is based on OMR_STATUS type.

OMRSTATUS is defined as, typed unsigned int OMR_STATUS, and includes the following constants.

C Constant Name	Actual Value	Content
SR_SUCCESS	0x00000000	Normal
SR_UNSUCCESSFUL	0x00000001	Error stop (details unknown)
SR_DISCONNECTED	0x00000002	OMR not yet connected
SR_WRONG_PARAMETER	0x00000003	Parameter Incorrect
SR_MEMORY_ERROR	0x00000004	Failure to obtain memory with in DLL
SR_TIMEOUT	0x00000005	Connection Time-Out
SR_RECEIVE_NAK	0x00000006	Received NAK from OMR
SR_WRONG_RESPONSE	0x00000007	Response Format Incorrect
SR_ERROR_STATUS_A1	0x00010000	Main body: Memory Error 1
SR_ERROR_STATUS_A2	0x00010001	Main body: Memory Error 2
SR_ERROR_STATUS_A3	0x00010002	Mainbody: Hopper Activation Error
SR_ERROR_STATUS_A4	0x00010003	Mainbody: Download Error
SR_ERROR_STATUS_A5	0x00010004	Mainbody: Sensor Type Error
SR_ERROR_STATUS_A6	0x00010005	Mainbody: Option Error
SR_ERROR_STATUS_A7	0x00010006	Hardware: Unit Not Connected
SR_ERROR_STATUS_A8	0x00010007	Mainbody: Power Supply Error
SR_ERROR_STATUS_B1F	0x01020000	Front Sensor Unit: Network Communication Error
SR_ERROR_STATUS_B2F	0x01020001	Front Sensor Unit: Internal Communication Error
SR_ERROR_STATUS_B3F	0x01020002	Front Sensor Unit: Memory Error
SR_ERROR_STATUS_B4F	0x01020003	Front Sensor Unit: Adjust Value Error
SR_ERROR_STATUS_B5F	0x01020004	Front Sensor Unit: Download Error
SR_ERROR_STATUS_B6F	0x01020005	Front Sensor Unit: Internal Error
SR_ERROR_STATUS_B7F	0x01020006	Front Reader Unit: Version Compatibility Erro
SR_ERROR_STATUS_B1B	0x02020000	Back Sensor Unit: Network Communication Error
SR_ERROR_STATUS_B2B	0x02020001	Back Sensor Unit: Internal Communication Error
SR_ERROR_STATUS_B3B	0x02020002	Back Sensor Unit: Memory Error
SR_ERROR_STATUS_B4B	0x02020003	Back Sensor Unit: Adjust Value Error
SR_ERROR_STATUS_B5B	0x02020004	Back Sensor Unit: Download Error
SR_ERROR_STATUS_B6B	0x02020005	Back Sensor Unit: Internal Error
SR_ERROR_STATUS_B7B	0x02020006	Back Sensor Unit: Version Compatibility Error
SR_ERROR_STATUS_C1	0x03030000	Barcode Reader Unit: Network Communication Error
SR_ERROR_STATUS_C2	0x03030001	Barcode Reader Unit: Internal Communication Error
SR_ERROR_STATUS_C3	0x03030002	Barcode Reader Unit: Memory Error
SR_ERROR_STATUS_C4	0x03030003	Barcode Reader Unit: Sensor Error
SR_ERROR_STATUS_C5	0x03030004	Barcode Reader Unit: Download Error
SR_ERROR_STATUS_C6	0x03030005	Barcode Reader Unit: Internal Error
SR_ERROR_STATUS_C7	0x03030006	Barcode Unit: Version Compatibility Error
SR_ERROR_STATUS_D1	0x04040000	Printer Unit: Network Communication Error
SR_ERROR_STATUS_D2	0x04040001	Printer Unit: Internal Communication Error
SR_ERROR_STATUS_D3	0x04040002	Printer Unit: Memory Error
SR_ERROR_STATUS_D4	0x04040003	Printer Unit: Download Error
SR_ERROR_STATUS_D5	0x04040004	Printer Unit: Internal Erroe
SR_ERROR_STATUS_D6	0x04040005	Printer Unit: Version Compatibility Error

C Constant Name	Actual Value	Content
SR_ERROR_STATUS_E1	0x05050000	Stacker Unit: Network Communication Error
SR_ERROR_STATUS_E2	0x05050001	Stacker Unit: Internal Communication Error
SR_ERROR_STATUS_E3	0x05050002	Stacker Unit: Memory Error
SR_ERROR_STATUS_E4	0x05050003	Stacker Unit: Download Error
SR_ERROR_STATUS_E5	0x05050004	Stacker Unit: Internal Error
SR_ERROR_STATUS_E6	0x05050005	Stacker Unit: Version Compatibility Error
SR_ERROR_STATUS_F1	0x1f060000	(reserved)
SR_ERROR_STATUS_F2	0x1f060001	(reserved)
SR_ERROR_STATUS_F3	0x1f060002	(reserved)
SR_ERROR_STATUS_F4	0x1f060003	(reserved)
SR_ERROR_STATUS_F5	0x1f060004	Command Error
SR_ERROR_STATUS_F6	0x1f060005	Parameter Error
SR_ERROR_STATUS_F7	0x1f060006	Protocol Error
SR_ERROR_STATUS_G1	0x20070000	Main body: Cover Open
SR_ERROR_STATUS_G2	0x25070001	Stacker Unit: Cover Open
SR_ERROR_STATUS_H1	0x30080000	Main body: No Paper
SR_ERROR_STATUS_H2	0x30080001	Main body: Paper Feed Sensor Jam
SR_ERROR_STATUS_H3	0x30080002	Main body: Read Sensor Paper Jam
SR_ERROR_STATUS_H4	0x30080003	Main body: Output Section Paper Jam
SR_ERROR_STATUS_I1	0x35090000	Stacker Unit: Printer Paper Sensor Unit Jam
SR_ERROR_STATUS_I2	0x35090001	Stacker Unit: Main Paper Outlet Jam
SR_ERROR_STATUS_I3	0x35090002	Stacker Unit: Selected Paper Outlet Jam
SR_ERROR_STATUS_P1	0x42100000	Back Sensor Unit: Disconnected
SR_ERROR_STATUS_P2	0x43100001	Barcode Unit: Disconnected
SR_ERROR_STATUS_P3	0x44100002	Printer Unit: Disconnected
SR_ERROR_STATUS_P4	0x45100003	Stacker Unit: Disconnected
SR_ERROR_STATUS_Q1	0x40110000	Sheet Empty
SR_ERROR_STATUS_Q2	0x40110001	Double Feed Error
SR_ERROR_STATUS_Q3	0x40110002	Left Side Skewer Error
SR_ERROR_STATUS_Q4	0x40110003	Mark Scewer Error
SR_ERROR_STATUS_R1	0x40120000	Main body: Hopper Emergency Termination
SR_ERROR_STATUS_R2	0x40120001	Main body: Extract Error
SR_ERROR_STATUS_R3	0x40120002	Main body: Sheet Insert Time-Out
SR_ERROR_STATUS_R4M	0x40120003	Front/Back Sensor Unit: Timing Mark Error
SR_ERROR_STATUS_R4F	0x41120003	Front Sensor Unit: Timing Mark Error
SR_ERROR_STATUS_R4B	0x42120003	Back Sensor Unit: Timing Mark Error
SR_ERROR_STATUS_R5M	0x40120004	Front/Back Reader Unit: Configuration Error
SR_ERROR_STATUS_R5F	0x41120004	Front Reader Unit: Configuration Error
SR_ERROR_STATUS_R5B	0x42120004	Back Reader Unit: Configuration Error
SR_ERROR_STATUS_S1F	0x41130000	Front Sensor Unit: White Label Error
SR_ERROR_STATUS_S2F	0x41130001	Front Sensor Unit: Black Label Error
SR_ERROR_STATUS_S1B	0x42130002	Back Sensor Unit: White Label Error
SR_ERROR_STATUS_S2B	0x42130003	Back Sensor Unit: Black Label Error
SR_ERROR_STATUS_S3	0x40130002	Main body: Read Sensor Stain Error
SR_ERROR_STATUS_T1	0x40140000	Main body: Paper Remaining near Paper Feed Sensor
SR_ERROR_STATUS_T2	0x40140001	Main body: Paper Remaining near Beginning Reader Sensor
SR_ERROR_STATUS_T3	0x40140002	Main body: Paper Remaining near Paper Output Sensor
SR_ERROR_STATUS_T4	0x45140003	Stacker Unit: Paper Remaining in Printer Paper Sensor Unit
SR_ERROR_STATUS_T5	0x45140004	Stacker Unit: Paper Remaining in Main Paper Outlet
SR_ERROR_STATUS_T6	0x45140005	Stacker Unit: Paper Remaining in Selected Paper Outlet
SR_ERROR_TERM	0xffffffff	Nondefined Status Information

Rule of Constant for OMR STATUS

With the exemption of SR_SUCCESS, the smaller the number displayed, the more serious the error. There are two types of status information: for the front sensor unit and for the back sensor unit. The higher priority item is selected, and if both front and back sensor unit items are at the same level of priority, the front sensor unit will be selected first.

Bit31	Priority (4 Bits)
:	0x0 : Hardware Error (release disabled)
:	0x1 : Connection Error
:	0x2 : Cover Open
:	0x3 : Paper Jam
Bit28	0x4 : Warning/Operation error
Bit27	Problematic Area (4 Bits)
:	0x0 : Main body
:	0x1 : Front Sensor Unit
:	0x2 : Back Sensor Unit
:	0x3 : Barcode Unit
:	0x4 : Printer Unit
:	0x5 : Stacker Unit
Bit24	0xf : Others
Bit23	Page Number (8 Bits)
:	The pages are divided in alphabetical order according to the first digit of the status infomation.
:	0x00 : Error during Communication (an error occurring prior to gaining status information)
:	0x01 : Status information of 1st digit=A
:	:
Bit16	0x1A : Status information of 1st digit=Z
Bit15	Through Number (16 Bits)
:	Through number per each page
Bit0	

4.2 API System Control

4.2.1 OMR_OpenDeviceUSB

Prototype	OMR_STATUS OMR_OpenDeviceUSB(void)	
Process	Detects a device connected to the USB and opens the device.	
Parameter	None	
Response Value	SR_SUCCESS	Successful
	SR_UNSUCCESSFUL	Failure (there is no device that can be opened, or is preoccupied by another connection).
Details	When a multiple number of OMR devices are connected, internal control will allow priority connection with the initial OMR device.	

4.2.2 OMR_CloseDevice

Prototype	OMR_STATUS OMR_CloseDevice(void)	
Process	Closes a device handler opened by an OMR_OpenDeviceUSB function. Must be conducted when closing down an application.	
Parameter	None	
Response Value	SR_SUCCESS	Successful
	SR_UNSUCCESSFUL	Failure

4.2.3 OMR_GetLastError

Prototype	OMR_STATUS OMR_GetLastError(void)
Process	Most control API can only retrieve success or failure response values. If unsuccessful, one method to find the cause is to use the OMR_STATUS function value as the last recorded data.
Parameter	None
Response Value	The last recorded OMR_STATUS value
Details	The OMR_STATUS is defined as typedef unsigned int OMR_STATUS. Please refer to the "Constant" section for further details on storage. When executing OMR_OpenDeviceUSB/OMR_CloseDevice/OMR_GetLastError, the OMR_STATUS will not be recorded.

4.2.4 OMR_FormatMessage

Prototype	CHAR *OMR_FormatMessage (OMR_STATUS status, int iLanguageFlag)	
Prosess	Convert OMR_STATUS value into text string.	
Parameter	status	OMR_STATUS value to be converted
	iLanguageFlag	Output Language Setting SR_STRING_NORMAL: not defined (English) SR_STRING_ENGLISH: English (only ASCII Code) SR_STRING_JAPANESE: Japanese (Shift-JIS Code)
Return Value	Pointer to the converted text string (fixed value)	
Details	Refer to the OMR_STATUS constant list for conversion results	
Example	The following usage is possible combined with OMR_GetLastEerror. print(OMR_Format Message (OMR_GetLastError0, ST_STRING_NORMAL	

4.2.5 OMR_GetST

Prototype	const CHAR *OMR_GetST(int iPage)	
Prosess	Directly output status information received during last response.	
Parameter	iPage	SR_PAGE_FRONT : Assign ST1 data SR_PAGE_BACK : Assign ST2 data
Return Value	Pointer to status data text string (fixed value). Double bite text string. If the text string is empty (text lwnfth is 0), there is no data or the deduction value is incorrect.	
Details	Example PRINT("%s"),OMR_GetST(SR_PAGE_FRONT));	

4.3 Read Settings

4.3.1 OMR_SetNumberOfColumnsToRead

Prototype	BOOL OMR_SetNumberOfColumnsToRead(int iColumns)	
Prosess	Use SetNumberOfColumnsToRead comand to use the value set by the parameter to set the number of columns to read.	
Parameter	iColumns	Set the designated line number. If selecting setting the value of 0, it will return to the initial setting.
Return Value	TRUE	Successful
	FALSE	Unsuccessful
Details	The line setting will differ depending on the reader unit instaled in each OMR hardware.	

Reader Unit Sensor Pitch	Setting Value	Initial Value
1/6 inch	1 - 48	48
0.2 inch	1 - 40	40
0.2 inch S	1 - 40	40
0.25 inch	1 - 33	33
0.3 inch	1 - 27	27
0.3 inch F	1 - 24	24

4.3.2 OMR_GetNumberOfColumnsToRead

Prototype	int OMR_GetNumberOfColumnsToRead(void)	
Process	This command is used to get the number of columns to be read that have been set by the OMR.	
Parameter	None	
Reterne Value	0	Failure
	Other than 0	Number of columns as set by the OMR

4.3.3 OMR_SetReadingMethod

Prototype	BOOL OMR_SetReadingMethod(int iControlType, int iMultipleValue)	
Process	This command RM sets the method which the OMR reads marks.	
Parameter	iControlType	Set read control values SR_READ_INITIAL : Retern to initial value (direct control value=3) SR_READ_FRONT_EDGE: Front edge control method SR_READ_REAR_EDGE : Rear edge control method SR_READ_DIRECT : Direct method SR_READ_FACOM : FACOM method SR_READ_BETWEEN_MARK_NO_SPACE : No mark space method (without reading front edge margin) SR_READ_BETWEEN_MARK : Between mark method (with reading of front margins)
	iMultiple Value	Multiple Control Front control type is set from 1-9 Rear control type is set from 2-9 However, these values are ignored when not being a control type.
Return Value	TRUE	Successful
	FALSE	Failure

4.3.4 OMR_GetReadingMethod

Prototype	BOOL OMR_GetReadingMethod(int *iControlType, int *iMultipleValue)	
Process	The set value can be gained by selecting the reading method command RM.	
Parameter	*iControlType	An address that stores the value of the reading method command RM.
	*iMultipleValue	The address that stores the control multiple value that was read.
Return Value	TRUE	Successful
	FALSE	Failure

Details	Example
	int ctl_type, multi_val:
	
	//Gain and execute
	OMR_GetReadingMethod(&ctl_type, &multi_val)	

4.3.5 OMR_SetBackSensorUnit

Prototype	BOOL OMR_SetBackSensorUnit(int iDirective)	
Process	The Set Back Sensor Unit command can be used to set whether or not the Back Sensor Unit should be used.	
Parameter	iDirective	Command setting SR_INTERNAL : reset to initial setting (for use) SR_ENABLE : Setting enabled SR_DISABLE : Setting disabled
Return Value	TRUE	Successful
	FALSE	Failure

4.3.6 OMR_GetBackSensorUnit

Prototype	int OMR_GetBackSensorUnit(void)	
Process	Use the Get Back Sensor Unit command to attain the enabling, disabling of the back reading sensor unit.	
Parameter	None	
Return Value	SR_ENABLE	Setting enabled
	SR_DISABLE	Setting disabled

4.3.7 OMR_SetSheetFeedMode

Prototype	BOOL OMR_SetSheetFeedMode(int iMode, int iInsertTime)	
Process	Set paper feed mode by using paper feed mode command FM.	
Parameter	iMode	Paper feed mode setting. The following settings are required. SR_MODE_AUTO : Automatic paper feed mode SR_MODE_MANUAL : Manual paper feed mode SR_INITIAL : Return to initial setting (automatic paper feed mode timing 10[sec])
	iInsertTime	Sheet insert time. 0-99[sec]. When set to 0, there is no time limit. Other than manual feed mode, settings are denied.
Return Value	TRUE	Successful
	FALSE	Failure

4.3.8 OMR_GetSheetFreeMode

Prototype	BOOL OMR_GetSheetFeedMode(int *iMode, int *iInsertTime)	
Process	Set feed mode by using feed mode command FM.	
Parameter	*iMode	Storage address for of set feed mode. SR_MODE_AUTO : Automatic paper feed mode SR_MODE_MANUAL : Manual paper feed mode
	*iInsertTime	Address for set sheet insert timing
Return Value	TRUE	Successful
	FALSE	Failure

Details	Example
	int feed_mode, insert_time;
	
	//Execute
	OMR_GetsheetFeedMode(&feed_mode,&insert_time);	

4.3.9 OMR_SetSheetThickness

Prototype	BOOL OMR_SetSheetThickness(int iThickness)	
Process	Set the ream weight (thickness) using the ream weight setting command FT.	
Parameter	iThickness	Sheet ream weight (thickness) setting SR_THICKNESS_AUTO_DETECT : Automatic detection SR_THICKNESS_64_GPM2 : 64g/m2(55 kg sheet) SR_THICKNESS_84_GPM2 : 84g/m2(72 kg sheet) SR_THICKNESS_105_GPM2 : 105g/m2(90 kg sheet) SR_THICKNESS_128_GPM2 : 128g/m2(110 kg sheet) SR_THICKNESS_157_GPM2 : 157g/m2(135 kg sheet) SR_THICKNESS_INTERNAL : Return to default value (90 kg sheet)
Return Value	TRUE	Successful
	FALSE	Failure

4.3.10 OMR_GetSheetThickness

Prototype	int OMR_SetSheetThickness(void)	
Process	Get the ream weight (thickness) using the ream weightsetting command FT.	
Parameter	None	
Return Value	SR_FUNCTION_FAIL	Error
	Other than mentioned above	Sheet ream weight (thickness) SR_THICKNESS_AUTO_DETECT : Automatic detection SR_THICKNESS_64_GPM2 : 64g/m2(55 kg sheet) SR_THICKNESS_84_GPM2 : 84g/m2(72 kg sheet) SR_THICKNESS_105_GPM2 : 105g/m2(90 kg sheet) SR_THICKNESS_128_GPM2 : 128g/m2(110 kg sheet) SR_THICKNESS_157_GPM2 : 157g/m2(135 kg sheet)

4.3.11 OMR_SetWarningError

Prototype	BOOL OMR_SetWarning Error(DWORD dwConfigDate, int iSkewCol, int iSkewLevel)	
Process	Use the SetWarningError command WE to designate the conditions for warnings.	
Deduction	dwConfigData	Set warning data by the following bits SR_WARN_AUTO_REJECT SR_WARN_SHEET_EMPTY SR_WARN_TM_ERROR SR_WARN_ DF_ERROR
		SR_WARN_LEFT_SKEW SR_WARN_MARK_SKEW SR_WARN_INITIAL These values can be enabled simultaneously using the logical sum. However, SR_WARN_INITIAL must be used alone.
	iSkewCol	Mark skew finding column Assign 1-155.
	iASkewLevel	Mark skew detection level Assign 1-16.
Retern Value	TRUE	Successful
	FALSE	Failure
Details	Refer to OMR_GetWarningError dwConfigData initial value SR_WARN_AUTO_REJECT : Automatic paper output enabled SR_WARN_SHEET_EMPTY : Sheet empty detected (disabled) SR_WARN_TM_ERROR : Timing marker error detected (enabled) SR_WARN_ DF_ERROR : Double feed detected (enabled) SR_WARN_LEFT-SKEW : Left skew detected (enabled) SR_WARN_MARK_SKEW : Marked skew detected (disabled) iSkewCol initial value : 001 iSkewLevel initial value : 4	

4.3.12 OMR_GetWarningError

<table>
<tr><td>Prototype</td><td colspan="2">BOOL OMR_GetWarningError(DWORD *dwConfigDate, int *iSkewCol, int *iSkewLevel)</td></tr>
<tr><td>Process</td><td colspan="2">Use the GetWarningError command WE to obtain the designated conditions for warnings.</td></tr>
<tr><td rowspan="3">Parameter</td><td>*dwConfigData</td><td>Return whether warning should be enabled/disabled using bits.
SR_WARN_AUTO_REJECT : Automatic paper output enabled
SR_WARN_SHEET_EMPTY : Sheet empty detected
SR_WARN_TM_ERROR : Timing marker error detected
SR_WARN_ DF_ERROR : Double feed detected
SR_WARN_LEFT-SKEW : Left skew detected
SR_WARN_MARK_SKEW : Marked skew detected
The logic sum of the above.</td></tr>
<tr><td>*iSkewCol</td><td>Address to save the mark skew detection column.</td></tr>
<tr><td>*iSkewLevel</td><td>Address to save the mark skew detection level.</td></tr>
<tr><td rowspan="2">Retern Value</td><td>TRUE</td><td>Successful</td></tr>
<tr><td>FALSE</td><td>Failure</td></tr>
<tr><td>Details</td><td colspan="2">Example
DWORD warm_info;
int SkewCol;
int SkewLevel;
warm_info = OMR_GetWarningError();
if(!OMR_GetWarningError(&warm_info, &SkewCol, &SkewLevwl)){
//Error process is noted here
}
if((warn_info&SR_WARN_DF_EFFOR)!=0){
//Double field detection enabled at this point
}

//In addition to the current setting, hopper empty enabled, timing mark error enabled, mark skew enabled columns=13 set to detection level 7.
OMR_SetWarningError((warm_info | SR_WARN_EMPTY |
SR_WARN_MARK_SKEW)& ~SR_WARN_TM_ERROR, 13, 7);</td></tr>
</table>

4.3.13 OMR_SetPanelConfig

Prototype	BOOL OMR_SetPanelConfig(int iPanelEnable)	
Process	Use SetPanelConfig command PO to enable/disable panel operations.	
Parameter	iPanelEnable	Panel operation enable/disable SR_DISABLE : Disable panel operations SR_ENABLE : Enable panel operations SR_INITIAL : Return to Initial value (enabled)
Retern Value	TRUE	Successful
	FALSE	Failure

4.3.14 OMR_GetPanelConfig

Prototype	int OMR_GetPanelConfig(void)	
Process	Use GetPanelConfig command PO to obtain setting for enable/disable panel operation.	
Parameter	None	
Retern Value	SR_FUNCTION_FAIL	Failure
	Others	SR_DISABLE : Panel opperation is set at disabled SR_ENABLE : Panel operation is set at enabled

4.3.15 OMR_SetBuzzerConfig

Prototype	BOOL OMR_SetBuzzerConfig(int iVolume, int iTone)	
Process	Use the SetBuzzerConfig command BZ to set the volume and tone of the buzzer. If the buzzer is disabled, all buzzers are disabled, so that error messages will be indicated only through status or panel display.	
Parameter	iVolume	Buzzer Volume 1-5 SR_BUZZER_DISSABLE: Buzzer disabled SR_BUZZER_INITIAL: Buzzer set to initial value (buzzer enabled/volume:3/sound:2)
	iTone	Buzzer sound 1-3 Disabled when iVolume is set to SR_BUZZER_DISABLE,SR_BUZZER_INITIAL
Retern Value	TRUE	Successful
	FALSE	Failure

4.3.16 OMR_GetBuzzerConfig

Prototype	BOOL OMR_GetBuzzerConfig(int *iVolimu, int *iTone)	
Process	Obtain volume and sound setting using the GetBuzzerConfig command BZ.	
Parameter	*iVolume	Address to store the buzzer volume
	*Tone	Address to store the buzzer tone
Retern Value	TRUE	Successful
	FALSE	Failure

4.3.17 OMR_SetID

Prototype	BOOL OMR_SetID(CHAR *pID)	
Process	The Set ID command is used for setting the OMR identification code. Setting the identification code makes it possible to distinguish it from applications, etc.	
Parameter	*pID	CHAR type front pointer that saves the text string. The text string code is set at 0x00-0xFF. Set at 20 characters.
Return value	TRUE	Successful
	FALSE	Failure

4.3.18 OMR_GetID

Prototype	BOOL OMR_GetID(CHAR *pID)	
Process	Use GetID command for obtaining the OMR's ID code.	
Parameter	*pID	Address for storing the identification address (initial setting) 'SEKONIC machine name'
Return value	TRUE	Successful
	FALSE	Failure
Details	Initialization example: 'SEKONIC SR-2300' Within 20 characters with no NULL at the end.	

4.3.19 OMR_SetSleepTime

Prototype	BOOL OMR_SetSleepTime(int iSleepTime, int iStandbyTime)	
Process	Use the energy-saver command ES to set the time interval before the machine goes into sleep/standby mode. This option may not be available on some models. Please consult your user's manual.	
Parameters	iSeepTime	Time to sleep mode 1-60 (minutes) SR_SLEEPTIME_DISABLE :Disable SR_SLEEPTIME_INITIAL :Return to Default Setting
	iStandbyTime	Time to standby mode 1-60 (minutes) SR_STANDBYTIME_DISABLE :Disable
Retern Value	TRUE	Successful
	FALSE	Failed

4.3.20 OMR_GetSleepTime

Prototype	BOOL OMR_GetSleepTime(int* piSleepTime, int* piStandbyTime)	
Process	Use the energy-saver command ES to get the time interval before the machine goes into sleep/standby mode. This option may not be available on some models. Please consult your user's manual.	
Parameters	piSleepTime	Time to sleep mode 1-60 (minutes) SR_SLEEPTIME_DISABLE :Disable
	piStandbyTime	Time to standby mode 1-60 (minutes) SR_STANDBYTIME_DISABLE :Disable
Retern Value	TRUE	Successful
	FALSE	Failed

4.4 Operation Requirements

4.4.1 OMR_Reset

Prototype	BOOL OMR_Reset(void)	
Process	Use Reset command SR to re-set the OMR to initial status before power was initiated.	
Parameter	None	
Return value	TRUE	Successful
	FALSE	Failure

4.4.2 OMR_FeedSheet

Prototype	BOOL OMR_FeedSheet(void)	
Process	Use the FeedSheet command SF to send one sheet through the OMR.	
Parameter	None	
Return value	TRUE	Successful
	FALSE	Failure

4.4.3 OMR_MoveHopper

Prototype	BOOL OMR_MoveHopper(int iDrection)	
Process	Use the MoveHopper command HU to enable up/down movement of the hopper.	
Parameter	iDirection	Hopper Direction SR_HOPPER_DOWN : down SR_HOPPER_UP : up
Return value	TRUE	Successful
	FALSE	Failure

4.4.4 OMR_EjectSheet

Prototype	BOOL OMR_EjectSheet(int iDirection)	
Process	Use the EjectSheet command ER to activate sheet eject fanction.	
Parameter	iDirection	Setting eject action SR_EJECT_MAIN :Eject sheet to Main Stacker SR_EJECT_SELECT :Eject sheet to Select Stacker SR_EJECT_MAIN_ON_NEXT :Eject sheet to Main Stacker on the next SF command. SR_EJECT_SELECT_ON_NEXT :Eject sheet to Select Stacker on the next SF command.
Return value	TRUE	Successful
	FALSE	Failure

4.4.5 OMR_InitialSetting

Prototype	BOOL OMR_InitialSetting(void)	
Process	Use InitialSetting command is to return and save OMR setting to time of shipment from factory.	
Parameter	None	
Return value	TRUE	Successful
	FALSE	Failure

4.4.6 OMR_CancelError

Prototype	OMR_STATUS OMR_CancelError(void)	
Process	Use CancelError command CE to gain status data from OMR.	
Parameter	None	
Retern Value	OMR_STATUS	Refer to OMR_STATUS constant list for further details
Details	Also reflected in OMR_STATUS available from OMR_GetLastError	

4.5 Data Request

4.5.1 OMR_GetMarks

<table>
<tr><td>Prototype</td><td colspan="2">BOOL OMR_GetMarks(int iPage, OMR_MARK_INFO* pMarkInfo, CHAR* pMarks)</td></tr>
<tr><td>Process</td><td colspan="2">Gain mark data and save in the OMR_MARK structure sent by the parameter.</td></tr>
<tr><td rowspan="3">Parameter</td><td>iPage</td><td>Front/back detection flag
SR_PAGE_FRONT : front mark data
SR_PAGE_BACK : back mark data</td></tr>
<tr><td>pMark</td><td>Pointer to the structure for mark information storage use.
Before this function executes, it sets the maximum values to be acquired for its constituent variables, “rows” and “columns”. The third parameter (pMarks) reserves memory in excess of that expressed by the variable “rows X columns”.
When this function executes, it stores the mark information.
typedef struct tagOMR_MARK_INFO
{
long type; //density 16/256 not including gray scale
long rows; //number of columns
long columns; //number of rows
}OMR_MARK_INFO, *POMR_MARK_INFO;</td></tr>
<tr><td>pMarks</td><td>Mark information takes up one byte of memory for each mark and is stored as 0x0 to 0x10 binary code.
Memory in excess of that for the constituent variables “rows X columns” for the second parameter is reserved.</td></tr>
<tr><td rowspan="2">Return value</td><td>TRUE</td><td>Successful</td></tr>
<tr><td>FALSE</td><td>Failure</td></tr>
<tr><td>Details</td><td colspan="2">Before this function executes, it sets the maximum number of marks for the constituent variables “rows” and “columns” in the second parameter.
Also, the third parameter pointer pMarks reserves memory in excess of that expressed by the variable “rows X columns”.
If the actual amount of mark data exceeds that which was previously set in “rows X columns”, pMarks will store up to the amount that was previously set.
Sample
OMR_MARK_INFO MarkInfo;
CHAR Marks[48*155];
MarkInfo.columnss = 48;
MarkInfo.rows = 155;
if(!OMR_GetMarks(SR_PAGE_FRONT, &MarkData, Marks))
{
// Error Handling
}
else{
// Mark Data Handling
}</td></tr>
</table>

4.5.2 OMR_GetStatus

Prototype	OMR_STATUS OMR_GetStatus(void)
Process	Obtain status information from the OMR by using the GetStatus command.
Parameter	None
Return Value	OMR_STATUS See OMR_STATUS constant chart.
Details	Reflected in OMR_STATUS, which can be obtained through OMR_GetLastError.

4.5.3 OMR_GetSensorInfo

Prototype	DWORD OMR_GetSensorInfo(void)	
Process	Use GetSensorInfo command DS to find On/Off information of sensors other than the reading sensors.	
Parameters	None	
Return Value	0xXXXXXXXX	Successful
	SR_FUNCTIONAL_FAIL	Failure

Details

When the return value selects a value other than SR_SENSOR_FAIL,
DWORD type 32 bit will respond to the on/off data of each bit per sensor.
Each bit response is as listed below.

Bit	Response Sensor	Mask Constant	Conatant Value
Bit31	None:0 (fixed)		
Bit30	None:1 (fixed)		
Bit29	Main Body paper eject detection	SR_SENSOR_OUTPS	0x20000000
Bit28		SR_SENSOR_RDPS	0x10000000
Bit27	Sheet feed start detection	SR_SENSOR_INPS	0x08000000
Bit26	0 paper detection	SR_SENSOR_PS0	0x04000000
Bit25	Hopper upper limit detection	SR_SENSOR_UPPS	0x02000000
Bit24	Hopper lower limit detection	SR_SENSOR_DWPS	0x01000000
Bit23	None:0 (fixed)		
Bit22	None:1 (fixed)		
Bit21	None:0 (for extenuation)		
Bit20	Skew sensor	SR_SENSOR_SKS	0x00100000
Bit19	None:0 (for extenuation)		
Bit18	None:0 (for extenuation)		
Bit17	None:0 (for extenuation)		
Bit16	Main Body door open/close detection	SR_SENSOR_MAIN_CVR	0x00010000
Bit15	None:0 (fixed)		
Bit14	None:1 (fixed)		
Bit13	None:0 (for extension)		
Bit12	None:0 (for extension)		
Bit11	None:0 (for extension)		
Bit10	Selected Paper Outlet	SR_SENSOR_SPS	0x00000400
Bit9	Main Paper Outlet Sensor	SR_SENSOR_MPS	0x00000200
Bit8	Printer 2 Print Start Sensor	SR_SENSOR_P2PS	0x00000100
Bit7	None:0 (fixed)		
Bit6	None:1 (fixed)		
Bit5	None:0 (for extenuation)		
Bit4	None:0 (for extenuation)		
Bit3	None:0 (for extenuation)		
Bit2	None:0 (for extenuation)		
Bit1	None:0 (for extenuation)		
Bit0	Stacker Unit Cover Sensor	SR_SENSOR_STK_CVR1	0x00000001

Example (receive Main Body paper eject detection)

```
DWORD sen_info;
sen_info=OMR_GetSensorInfo);
if(sen_info==SR_FUNCTION_FAIL){
        //error procedure should be noted here
}
if((sen_info&SR_SENSOR_OUTPUTS)!=0){
        //main body paper feed sensor ON data should be noted here.
}
```

4.5.4 OMR_GetDeviceInfo

Prototype	DWORD OMR_GetDeviceInfo(void)
Process	Use the GetDeviceInfo command DI to check what unit(s) are connected.
Parameter	None
Return Value	0xXXXXXXXX Successful (see details below)
	SR_FUNCTIONAL_FAIL Failure
Details	When the return value selects a value other than SR_SENSOR_FAIL, DWORD type 32 bit will respond to the on/off data of each bit per sensor. Each bit response is as listed below.

Bit	Content		Mask Constant	Constant Value
Bit31 Bit30 Bit29 Bit28	Stacker unit	0x0: Not Connected 0x1: Connected 0x8: No Cartridge	SR_DEVICE_UNIT_STACKER_MASK SR_DEVICE_UNIT_STACKER SR_DEVICE_UNIT_STACKER_ERR	0xf0000000 0x10000000 0x80000000
Bit27 Bit26 Bit25 Bit24	Printer unit	0x0: Not Connected 0x1: Connected 0x8: Not Cartridge	SR_DEVICE_UNIT_PRINTER_MASK SR_DEVICE_UNIT_PRINTER_ SR_DEVICE_UNIT_PRINTER_ERR	0x0f000000 0x01000000 0x08000000
Bit23 Bit22 Bit21 Bit20	Barcode unit	0x0: Disconnected 0x1: Connected vertical 0x2: Connected horizontal 0x8: Connecting Error	SR_DEVICE_UNIT_BARCODE_MASK SR_DEVICE_UNIT_BARCODE_V SR_DEVICE_UNIT_BARCODE_H SR_DEVICE_UNIT_BARCODE_ERR	0x00f00000 0x00100000 0x00200000 0x00800000
Bit19 Bit18 Bit17 Bit16	Back Sensor unit	0x0: Not Connected 0x1: Connected	SR_DEVICE_UNIT_BACK_MASK SR_DEVICE_UNIT_BACK	0x000f0000 0x00080000
Bit15 : Bit8	(aria out of use)			
Bit7 Bit6 Bit5 Bit4	Sensor type	0x0: Visible red 0x1: Near infra red 0x8: Connecting Error	SR_DEVICE_SENSOR_TYPE_MASK SR_DEVICE_SENSOR_TYPE_ERR	0x000000f0 0x00000080
Bit3 Bit2 Bit1 Bit0	Sensor pitch type	0x1:1/6 inches 0x2: 0.2 inches 0x3: 0.2 inches S 0x4: 0.25 inches 0x5: 0.3 inches 0x6: 0.3 inches F 0x7: 6mm 0x8: 0.2 Inch K 0x9: 0.2 Inch Special 0xA: 0.2 Inch C 0xF: Connecting Error	SR_DEVICE_SENSOR_PITCH_MASK SR_DEVICE_SENSOR_PITCH_ERR	0x0000000f 0x0000000f

Example (Stacker Unit Connection Conformation)

```
DWORD dev_info;
dev_info=OMR_GetDeviceInfo0;
if(dev-info==SR_FUNCTION_FAIL){
    //enter error procedure here
}
if((dev_info&SR_DEVICE_UNIT_STACKER)!=0){
    //stacker unit is connected at this point
}
```

4.5.5 OMR_GetMachineName

Prototype	BOOL OMR_GetMachineName(CHAR * pResult)	
Process	Use machine name command MN to store the machine name text string to the region defined by the parameter pointer.	
Parameter	*pResult	Storage address of text string for machine name
Return Value	TRUE	Successful
	FALSE	Unsuccessful
Details	Example: CHAR pBuf[16]; OMR_GetMachineName(pBuf); print(pBuf);	

4.5.6 OMR_GetVersion

Prototype	BOOL OMR_GetVersion(int iTarget, CHAR *pResult)	
Process	Use GetVersion command FV to store the machine name text string to the region defined by the parameter pointer.	
Parameter	iTarget	Select the unit for which version information is needed. SR_UNIT_MAIN : Main body SR_UNIT_FRONT : Front sensor unit SR_UNIT_BACK : Back sensor unit SR_UNIT_BARCODE : Barcode unit SR_UNIT_PRINTER : Printer unit SR_UNIT_STACKER : Stacker unit
	*pResult	Version storage address
Return Value	TRUE	Successful
	FALSE	Unsuccessful
Details	Example: Gain main body version data CHAR pBuf[16]; OMR_GetVersion(SR_UNIT_MAIN,pBuf); print(pBuf);	

4.6 Printer Configuration

4.6.1 OMR_SetPrinterUnit

Prototype	BOOL OMR_ SetPrinterUnit (int iDirective)	
Procedure	Sets printer controls using PR print setting command.	
Parameter	iDirective	Indicates setting status. SR_INITIAL :Re-set to default values SR_ENABLE :Printer controls enabled SR_DISABLE :Printer controls disabled
Return Values	TRUE	Successful
	FALSE	Failure

4.6.2 OMR_GetPrinterUnit

Prototype	int OMR_GetPrinterUnit (void)	
Procedure	Gets printer controls using PR print setting command.	
Parameters	None	
Return Values	SR_ENABLE	Printer controls enabled
	SR_DISABLE	Printer controls disabled
	SR_FUNCTION_FAIL	Acquisition failed

4.6.3 OMR_SetPrintOrder

Prototype	BOOL OMR_SetPrintOrder (int iFirst, int iSecond, int iThird)	
Process	Use the printer setting command PR to set the print order. This setting will clear after printing.	
Parameter	iFirst	Number of the first buffer to print. SR_PRINT_BUFFER_NULL : No printing SR_PRINT_BUFFER_1 : Buffer #1 SR_PRINT_BUFFER_2 : Buffer #2 SR_PRINT_BUFFER_3 : Buffer #3
	iSecond	Number of the second buffer to print.
	iThird	Number of the third buffer to print.
Return Value	TRUE	Successful
	FALSE	Failure

4.6.4 OMR_GetPrintOrder

Prototype	BOOL OMR_GetPrintOrder (int* pFirst, int* pSecond, int* pThird)	
Process	Use the printer setting command PR to get the print mode.	
Parameter	pFirst	Pointer storing the number of the first buffer to print.
	pSecond	Pointer storing the number of the second buffer to print.
	pThird	Pointer storing the number of the third buffer to print.
Return Value	TRUE	Successful
	FALSE	Failure

4.6.5 OMR_SetPrinterMode

Prototype	BOOL OMR_SetPrinterMode(int iMode)	
Process	Use the printer setting command PR to set the print mode.	
Parameter	iMode	Print Mode SR_PRINTER_MODE_AFTER_FEED : Print after feeding SR_PRINTER_MODE_FEED_AND_PRINT : Print while feeding
Return Value	TRUE	Successful
	FALSE	Failure

4.6.6 OMR_GetPrinterMode

Prototype	BOOL OMR_GetPrinterMode(int* piMode)	
Process	Use the printer setting command PR to get the print mode.	
Parameter	piMode	Address storing the print mode
Return Value	TRUE	Successful
	FALSE	Failure

4.6.7 OMR_SetPrintPosition

Prototype	BOOL OMR_SetPrintPosition (int iPosition)	
Process	Use the printer setting command PR to set the print position.	
Parameter	iPosition	Print Position (Unit: mm) * : For range of value settings, please refer to the appropriate user's manual.
Return Value	TRUE	Successful
	FALSE	Failure

4.6.8 OMR_GetPrintPosition

Prototype	BOOL OMR_GetPrintPosition(int* piPosition)	
Process	Use the printer setting command PR to get the print position.	
Parameter	piMode	Address storing the print position
Return Value	TRUE	Successful
	FALSE	Failure

4.6.9 OMR_SetPrintAngle

Prototype	BOOL OMR_SetPrintAngle(DWORD dwAngle)	
Process	Use the printer setting command PR to set the print angle.	
Parameter	dwAngle	Print Angle SR_PRINT_ANGLE_0 : Print normally SR_PRINT_ANGLE_180 : Rotate print string 180 degrees
Return Value	TRUE	Successful
	FALSE	Failure

4.6.10 OMR_GetPrintAngle

Prototype	BOOL OMR_GetPrintAngle(DWORD *dwAngle)	
Process	Use the printer setting command PR to get the print angle.	
Parameter	dwAngle	Address storing the print angle
Return Value	TRUE	Successful
	FALSE	Failure

4.6.11 OMR_SetPrintFontSize

Prototype	BOOL OMR_SetPrintFontSize(int iSize)	
Process	Use the printer setting command PR to set the print font size.	
Parameter	iSize	Print Font Size (Unit: 0.1mm) * : For range of value settings, please refer to the appropriate user's manual.
Return Value	TRUE	Successful
	FALSE	Failure

4.6.12 OMR_GetPrintFontSize

Prototype	BOOL OMR_GetPrintFontSize(int *iSize)	
Process	Use the printer setting command PR to get the print font size.	
Parameter	iSize	Address storing the print font size.
Return Value	TRUE	Successful
	FALSE	Failure

4.6.13 OMR_SetPrintFontPitch

Prototype	BOOL OMR_SetPrintFontSize(int iSize)	
Process	Use the printer setting command PR to set the print font pitch.	
Parameter	iPitch	Print Font Pitch (Unit: 0.1mm) * : For range of value settings, please refer to the appropriate user's manual.
Return Value	TRUE	Successful
	FALSE	Failure

4.6.14 OMR_GetPrintFontPitch

Prototype	BOOL OMR_GetPrintFontPitch(int *iPitch)	
Process	Use the printer setting command PR to get the print font pitch.	
Parameter	iPitch	Address storing the character spacing
Return Value	TRUE	Successful
	FALSE	Failure

4.6.15 OMR_SetPrintString

Prototype	OMR_SetPrintString(int iBufDirec, CHAR *pString)	
Process	Use the printer setting command PR to set the print string in the buffer.	
Parameter	iBufDirec	Designates the number of the buffer being configured
	pString	Pointer of the string being set String length can be set for 1 to 42 characters
Return Value	TRUE	Successful
	FALSE	Failure

4.6.16 OMR_GetPrintString

Prototype	BOOL OMR_GetPrintString(int iBufDirec, CHAR *pString)	
Process	Use the printer setting command PR to get the print string stored in the buffer.	
Parameter	iBufDirec	Designates the buffer being gotten
	pString	Pointer storing the string being gotten
Return Value	TRUE	Successful
	FALSE	Failure

4.7 Bar Code Controls

4.7.1 OMR_GetBarcodeInfo

Prototype	OMR_GetBarcodeInfo(int* piReadCount, int* piSettingCount)	
Procedure	Use the command BD to call up the bar code data and get the number of data scanned on a page.	
Parameters	piReadCount	Variable pointer for storing the number of scanned data (1-10).
	piSettingCount	Variable pointer for storing the number of scan position settings (1-10).
Return Values	TRUE	Successful
	FALSE	Failed

4.7.2 OMR_GetBarcodeData

Prototype	OMR_GetBarcodeData(int iIndex, CHAR* pBcType, int* piDataLen, CHAR* pBarcode)	
Procedure	Use the command BD to call up the bar code data and get the designated bar code data.	
Parameters	iIndex	Numerical designator of bar code data to be acquired (1-10).
	pBcType	Pointer for storing the one-byte indicator of bar code type. (See “Details” for values settings.)
	piDataLen	Pointer for storing the number of bar code data bytes.
	pBarcode	Pointer for storing the bar code data.
Return Values	TRUE	Successful
	FALSE	Failed
Details	Bar code types are expressed by the following constants.	

Name of Constant	Value	Meaning
SR_BARCODE_TYPE_UNKNOWN	‘@’	None/Unknown
SR_BARCODE_TYPE_CODE39	‘a’	CODE-39
SR_BARCODE_TYPE_ITF	‘b’	ITF
SR_BARCODE_TYPE_NW7	‘c’	NW-7(Codabar)
SR_BARCODE_TYPE_JAN_EAN_UPC	‘d’	JAN/EAN/UPC
SR_BARCODE_TYPE_CODE128	‘e’	CODE-128
SR_BARCODE_TYPE_INDUSTRIAL2OF5	‘f’	Industrial 2of5
SR_BARCODE_TYPE_COOP_2OF5	‘g’	COOP2of5
SR_BARCODE_TYPE_CODE93	‘h’	CODE-93

4.7.3 OMR_SetBarcodeReaderUnit

Prototype	OMR_SetBarcodeReaderUnit(int iDirective)	
Procedure	Use the bar code setting command BC to enable or disable the bar code scanner.	
Parameters	iDirective	Indicates whether to set or not. SR_INITIAL :Return to/Use Initial Default Setting SR_ENABLE :Enable SR_DISABLE :Disable
Return Values	TRUE	Successful
	FALSE	Failed

4.7.4 OMR_GetBarcodeReaderUnit

Prototype	OMR_GetBarcodeReaderUnit(void)	
Procedure	Use the bar code setting command BC to check the bar code reader for enabled or disabled status.	
Parameters	None	
Return Values	SR_ENABLE	Bar Code Scanner Enabled
	SR_DISABLE	Bar Code Scanner Disabled
	SR_FUNCTION_FAIL	Status Check Failed

4.7.5 OMR_SetBarCodeArea

Prototype	OMR_SetBarCodeArea(OMR_BARCODE_AREA* pAreaArray)	
Procedure	Use the bar code setting command BC to set the bar code scan area.	
Parameters	pAreaArray	Pointer to the array for the structure of the areas to be read. The array is in decimals. (See “Details” for more on the structure or value settings.)
Return Values	TRUE	Successful
	FALSE	Failed

Details	Scan Area Structure
	typedef struct tagOMR_BARCODE_AREA
	{
	long unread; // excluded area [mm]
	long read; // included area [mm]
	}OMR_BARCODE_AREA, *POMR_BARCODE_AREA;	

4.7.6 OMR_GetBarCodeArea

Prototype	OMR_GetBarCodeArea(OMR_BARCODE_AREA* pAreaArray)	
Procedure	Use the bar code setting command BC to get the bar code scan area.	
Parameters	pAreaArray	Pointer to the array for the structure of the areas to be read. The array is in decimals. Uses the caller to store this information in memory.
Return Values	TRUE	Successful
	FALSE	Failed

4.7.7 OMR_SetBarCodeReadType

Prototype	OMR_SetBarCodeReadType(DWORD dwReadType)	
Procedure	Use the bar code setting command BC to designate the type of bar code.	
Parameters	dwReadType	Sets the type of bar code to be read as a 32-bit DWORD type. (See “Details” for bit definitions.)
Return Values	TRUE	Successful
	FALSE	Failed

Details

The types of bar code to be read are expressed by the following constants.

Name of Constant	Value	Meaning
SR_BARCODE_READ_CODE39	0x00000001	CODE-39
SR_BARCODE_READ_ITF	0x00000002	ITF
SR_BARCODE_READ_NW7	0x00000004	NW-7(Codabar)
SR_BARCODE_READ_JAN_EAN_UPC	0x00000008	JAN/EAN/UPC
SR_BARCODE_READ_CODE128	0x00000010	CODE-128 EAN-128
SR_BARCODE_READ_INDUSTRIAL2OF5	0x00000020	Industrial 2of5
SR_BARCODE_READ_COOP2OF5	0x00000040	COOP 2of5
SR_BARCODE_READ_CODE93	0x00000080	CODE-93

4.7.8 OMR_GetBarCodeReadType

Prototype	OMR_GetBarCodeReadType(DWORD* pdwReadType)	
Procedure	Use the bar code setting command BC to get the type of bar code to be read.	
Parameters	pdwReadType	Displays the type of bar code to be read as a 32-bit DWORD type. (See details of setting functions for bit definitions.)
Return Values	TRUE	Successful
	FALSE	Failed

4.7.9 OMR_SetBarCodeCheckDigit

<table>
<tr><td>Prototype</td><td colspan="2">OMR_SetBarCodeCheckDigit(CHAR cBcType, int iCdType)</td></tr>
<tr><td>Procedure</td><td colspan="2">Use the bar code setting command BC to assign a check digit to each bar code type.
In the case of an invalid bar code or an invalid setting, a parameter error will occur.</td></tr>
<tr><td rowspan="2">Parameters</td><td>cBcType</td><td>Sets the type of bar code to be assigned.
(See “Details” for value settings.)</td></tr>
<tr><td>iCdType</td><td>Enables and sets the bar code type for or disables the check digit test.
(See “Details” for value settings.)</td></tr>
<tr><td>Return Values</td><td>TRUE</td><td>Successful</td></tr>
<tr><td></td><td>FALSE</td><td>Failed</td></tr>
<tr><td>Details</td><td colspan="2">Bar code types are expressed by the following constants.
<table>
<tr><th>Name of Constant</th><th>Value</th><th>Meaning</th></tr>
<tr><td>SR_BARCODE_TYPE_CODE39</td><td>‘a’</td><td>CODE-39</td></tr>
<tr><td>SR_BARCODE_TYPE_ITF</td><td>‘b’</td><td>ITF</td></tr>
<tr><td>SR_BARCODE_TYPE_NW7</td><td>‘c’</td><td>NW-7(Codabar)</td></tr>
</table>
Check digit types are expressed by the following constants.
For enabling or disabling the check digit test only:“CODE-39”/“gITF”
<table>
<tr><th>Name of Constant</th><th>Value</th><th>Meaning</th></tr>
<tr><td>SR_BARCODE_CD_INITIAL</td><td>SR_INITIAL</td><td>Initialize</td></tr>
<tr><td>SR_BARCODE_CD_DISABLE</td><td>0</td><td>Disable check digit test</td></tr>
<tr><td>SR_BARCODE_CD_ENABLE</td><td>1</td><td>Enable check digit test</td></tr>
</table>
When there is no check digit test or the bar code is of an assignable type: “NW-7”
<table>
<tr><th>Name of Constant</th><th>Value</th><th>Meaning</th></tr>
<tr><td>SR_BARCODE_CD_INITIAL</td><td>SR_INITIAL</td><td>Initialize</td></tr>
<tr><td>SR_BARCODE_CD_DISABLE</td><td>0</td><td>Disable check digit test</td></tr>
<tr><td>SR_BARCODE_CD_MODULO16</td><td>1</td><td>Modulus 16</td></tr>
<tr><td>SR_BARCODE_CD_MODULO11</td><td>2</td><td>Modulus 11</td></tr>
<tr><td>SR_BARCODE_CD_MODULO10WAIT2</td><td>3</td><td>Modulus 10/2 wait</td></tr>
<tr><td>SR_BARCODE_CD_MODULO10WAIT3</td><td>4</td><td>Modulus 10/3 wait</td></tr>
<tr><td>SR_BARCODE_CD_7DR</td><td>5</td><td>7DR</td></tr>
<tr><td>SR_BARCODE_CD_MODULO11W</td><td>6</td><td>Weighted Modulus 11</td></tr>
<tr><td>SR_BARCODE_CD_RUNES</td><td>7</td><td>Runes</td></tr>
</table></td></tr>
</table>

4.7.10 OMR_GetBarCodeCheckDigit

<table>
<tr><td>Prototype</td><td colspan="2">OMR_GetBarCodeCheckDigit(CHAR cBcType, int* piCdType)</td></tr>
<tr><td>Procedure</td><td colspan="2">Use the bar code setting command BC to get the assigned check digit for each bar code type.
For invalid bar codes or scan settings, a parameter error will occur.</td></tr>
<tr><td rowspan="2">Parameters</td><td>cBcType</td><td>Sets the type of bar code to be acquired.
(See details of setting functions for values.)</td></tr>
<tr><td>piCdType</td><td>Pointer storing the enabled or disabled status of the check digit test or the type of bar code. (See details of setting functions for values.)</td></tr>
<tr><td>Return Values</td><td>TRUE</td><td>Successful</td></tr>
<tr><td></td><td>FALSE</td><td>Failed</td></tr>
</table>

4.7.11 OMR_SetBarCodeUpcOption

<table>
<tr><td>Prototype</td><td colspan="2">OMR_SetBarCodeOption(CHAR cBcType, DWORD dwOption)</td></tr>
<tr><td>Procedure</td><td colspan="2">Use the bar code setting command BC to set each bar code.
For invalid bar codes or scan settings, a parameter error will occur.</td></tr>
<tr><td rowspan="2">Parameters</td><td>cBcType</td><td>Designates the type of bar code to be set.
(See “Details” for value settings.)</td></tr>
<tr><td>dwOption</td><td>Detail settings are expressed as 32-bit DWORD values.
(See “Details” for value settings.)</td></tr>
<tr><td rowspan="2">Return Values</td><td>TRUE</td><td>Successful</td></tr>
<tr><td>FALSE</td><td>Failed</td></tr>
<tr><td>Details</td><td colspan="2">Bar code types are expressed by the following constants.
<table>
<tr><th>Name of Constant</th><th>Value</th><th>Meaning</th></tr>
<tr><td>SR_BARCODE_TYPE_JAN_EAN_UPC</td><td>‘d’</td><td>JAN/EAN/UPC</td></tr>
</table>
Detail settings are expressed by the following constants.
Settings initialization(Can be used alone)
<table>
<tr><th>Name of Constant</th><th>Value</th><th>Meaning</th></tr>
<tr><td>SR_BARCODE_OPT_INITIAL</td><td>0xffffffff</td><td>Initialize</td></tr>
</table>
UPC-A Settings for Number of Output Digits (Either setting OK for JAN/EAN/UPC)
<table>
<tr><th>Name of Constant</th><th>Value</th><th>Meaning</th></tr>
<tr><td>SR_BARCODE_OPT_UPC_12DIGIT</td><td>0x00000001</td><td>12-digit output</td></tr>
<tr><td>SR_BARCODE_OPT_UPC_13DIGIT</td><td>0x00000002</td><td>13-digit output</td></tr>
</table>
UPC-E System Code Settings (Either setting OK for JAN/EAN/UPC)
<table>
<tr><th>Name of Constant</th><th>Value</th><th>Meaning</th></tr>
<tr><td>SR_BARCODE_OPT_UPC_NO_CODE</td><td>0x00000010</td><td>Do not affix code</td></tr>
<tr><td>SR_BARCODE_OPT_UPC_ADD_CODE</td><td>0x00000020</td><td>Affix code</td></tr>
</table></td></tr>
</table>

4.7.12 OMR_GetBarCodeUpcOption

<table>
<tr><td>Prototype</td><td colspan="2">OMR_GetBarCodeOption(CHAR cBcType, DWORD* pdwOption)</td></tr>
<tr><td>Procedure</td><td colspan="2">Use the bar code setting command BC to get the UPC output format.
For invalid UPCs, a parameter error will occur.</td></tr>
<tr><td rowspan="2">Parameters</td><td>cBcType</td><td>Designates the type of bar code to be set.
(See “Details” for value settings.)</td></tr>
<tr><td>pdwOption</td><td></td></tr>
<tr><td rowspan="2">Return Values</td><td>TRUE</td><td>Successful</td></tr>
<tr><td>FALSE</td><td>Failed</td></tr>
</table>

4.8 API List

No.	API Name	Content
1	OMR_OpenDeviceUSB	Open USB devise
2	OMR_CloseDevice	Close USB devise handler
3	OMR_GetLastError	Gain OMR_STATUS value
4	OMR_FormatMessage	Convert text string to OMR_STATUS value
5	OMR_GetST	Gain OMR_STATUS value
6	OMR_SetNumberOfColumnsToRead	Set number of columns to read
7	OMR_GetNumberOfColumnsToRead	Gain number of columns to read
8	OMR_SetReadingMethod	Set mark reading method
9	OMR_GetReadingMethod	Gain mark reading method
10	OMR_SetBackSensorUnit	Set enable/disable of back sensor unit
11	OMR_GetBackSensorUnit	Find enable/disable of back sensor unit
12	OMR_SetSheetFeedMode	Set sheet feed mode
13	OMR_GetSheetFeedMode	Gain sheet feed mode
14	OMR_SetSheetThickness	Set ream weight
15	OMR_GetSheetThickness	Gain ream weight
16	OMR_SetWarningError	Set warning conditions
17	OMR_GetWarningError	Gain warning conditions
18	OMR_SetPanelConfig	Set enable/disable panel operations
19	OMR_GetPanelConfig	Gain enable/disable panel operations
20	OME_SetBuzzerConfig	Set buzzer volume and tone
21	OME_GetBuzzerConfig	Gain buzzer volume and tone
22	OMR_SetID	Set identification code
23	OMR_GetID	Gain identification code
24	OMR_SetSleepTime	Sets the Sleep/Standby time
25	OMR_GetSleepTime	Gets the Sleep/Standby time
26	OMR_Reset	Reset to condition when power was turned on
27	OMR_FeedSheet	Send one sheet
28	OMR_MoveHopper	Raise/lower the hopper
29	OMR_EjectSheet	Eject sheet
30	OMR_InitialSetting	Initialize each setting
31	OMR_CancelError	Gain status information
32	OMR_GetMarks	Gain mark data
33	OMR_GetStatus	Gain status information
34	OMR_GetSensorInfo	Gain position sensor ON/OFF data
35	OMR_GetDeviceInfo	Check connected devices
36	OMR_GetMachineName	Gain machine name
37	OMR_GetVersion	Gain firmware version
38	OMR_SetPrinterUnit	Sets printer controls.
39	OMR_GetPrinterUnit	Gets printer controls.
40	OMR_SetPrinterOrder	Sets print order.
41	OMR_GetPrinterOrder	Get print order.
42	OMR_SetPrintMode	Set print mode
43	OMR_GetPrintMode	Get print mode
44	OMR_SetPrintPosition	Set print position
45	OMR_GetPrintPosition	Get print position
46	OMR_SetPrintAngle	Set print angle
47	OMR_GetPrintAngle	Get print angle
48	OMR_SetPrintFontSize	Set print font size
49	OMR_GetPrintFontSize	Get print font size

50	OMR_SetPrintFontPitch	Set print font pitch
51	OMR_GetPrintFontPitch	Get print font pitch
52	OMR_SetPrintString	Set print string in buffer
53	OMR_GetPrintString	Get print string of buffer
54	OMR_GetBarcodeInfo	Gets the number of bar codes per page
55	OMR_GetBarcodeData	Gets the scanned bar code data
56	OMR_SetBarcodeReaderUnit	Enables/disables the bar code reader
57	OMR_GetBarcodeReaderUnit	Gets the bar code reader enabled or disabled status
58	OMR_SetBarCodeArea	Sets the bar code area
59	OMR_GetBarCodeArea	Gets the bar code area
60	OMR_SetBarCodeReadType	Sets the type of bar code to be read
61	OMR_GetBarCodeReadType	Gets the type of bar code to be read
62	OMR_SetBarCodeCheckDigit	Assigns a check digit to each type of bar code
63	OMR_GetBarCodeCheckDigit	Gets the check digit assigned to each type of bar code
64	OMR_SetBarCodeOption	Gives separate settings to individual bar codes
65	OMR_GetBarCodeOption	Gets the settings for individual bar codes

OPTICAL MARK READER SR-2300/5500/3500/6000/6500
SR-1800

Windows API Reference Manual

May 2010 version 4.1

7-24-14, Oizumi Gakuen-cho, Nerima-ku, Tokyo 178-8686, Japan

Telephone: +81-3-3978-2375 Facsimile: +81-3-3978-5229

http://www.sekonic.co.jp

E-mail:omr@sekonic.co.jp

* The specifications of this product may change without notice.