SuperWedge™

Software Keyboard Wedge for Windows CE .NET

Version 3.210

Technical Reference

Revision E

July 20, 2006

Compsee, Inc.
www.compsee.com

Revision History

Rev.	Description of Change	Date
A	Initial Release for *SuperWedge* v3.110.	1/18/2006
B	Release for *SuperWedge* v3.120. Edited for better PDF conversion; Added copyright statement; 3 Changed "double-tapping" to "single tapping" to access the pop-up menu from the taskbar icon to reflect new operation; 4 Added a few more bullet points to further detail operation; 4.1 Moved DefaultAll to section 4.11; 4.2 Added "NumericKeypad" to describe new functionality; Changed "Window=" examples to be easier to read; Added extra verbiage to "Window=" to clarify its operation; 4.5 Clarified default settings for Custom Code IDs and how to clear a Custom Code ID that has been set; 4.8 Changed "TLC-39" to "TLC39" to be consistent with industry usage; 4.9 Added clarification for *vk* and *flags* usage; 4.10 Added description of returning edit modifiers to defaults; 4.11 Created new section; Added "DefaultEdits" to document new functionality. 7 Added Supported Symbologies section	3/10/2006
C	Release for *SuperWedge* v3.200 Added IT5180 imager support; Added Hand Held Products code ID support; Replaced "CompositeCodeAB" and "CompositeCodeC" with "CompositeCode"; 3.7 Added Command Line Parameter section; 4 Expanded description of INI parsing; Added sample INI; 7 Changed AIM Code ID for IATA25 from "S" to "R" to agree with AIM standards; Added Code 39 Full ASCII conversion chart; Changed Menu labels from Codabar to Code 128	5/15/2006
D	Release for *SuperWedge* v3.210 4.4 Changed "SE15234" to "SE1524" in Aim Dot enable description; 4.8 Added IT5180 to list of scanners that support Code 11; Changed "Industrial 2 of" to "Industrial 2 of 5"; Added clarification for ISBT-128	6/30/2006
E	Updated address information on last page	7/20/2006

Table of Contents

1 Introduction

1.1 Overview

SuperWedge™ is an extremely powerful Microsoft® Windows® CE utility that is designed to take raw data from multiple input sources, format the data and output the modified data to an application program. The application program can be a standard Windows CE application that is unaware of *SuperWedge* or it can be a custom written application that interfaces directly with *SuperWedge*.

Input data can come from up to two channels simultaneously:

- Built-in bar code scanner
- ASCII data from wired RS232 or Bluetooth® device via COM port emulation

Typical external devices used with *SuperWedge* include bar code readers and magnetic stripe readers although any device capable of serial communication can be used.

Output data can be directed to the keyboard so that data may be "wedged" into an existing application or data can be sent directly to an application via the Application Program Interface (API).

SuperWedge offers an array of features and benefits including:

- Keyboard wedge interface - Automatically directs data to the application window with focus or to a single, user-defined window
- Application Program Interface (API)
- Data Filtering and Validation
- Data Formatting
- Preambles and Postambles
- Key Remapping
- Code IDs
- Function Codes
- Keyboard Function Records
- Time and Date stamping
- Operates as a stand-alone task tray application
- Simple to install and configure

1.2 Supported Platforms

Windows CE 4.2
This version is for ARMV4 only!

Windows CE 4.1
This version is for ARMV4I only!

2 Installation

SuperWedge is distributed in two forms: as a full installation and as a minimum installation. The choice of which installation to perform is based on your usage needs.

When frequent, on-going changes to *SuperWedge* configuration parameters are required, the full installation is recommended.

The full installation requires approximately 860K of free program storage space.

The minimum installation works well in the following cases

- Your unit's program storage space is limited.
- You do not expect to make frequent changes to *SuperWedge* configuration parameters.
- You have a custom application program that uses the *SuperWedge* API to make configuration changes for you.

The minimum installation requires approximately 112K of free program storage space.

2.1 Full Installation

Full installation is found on the product support CD and consists of *SuperWedge* as well as the *SuperWedge Configuration Manager*.

The *Configuration Manager* is a graphical user interface for easily setting up all *SuperWedge* parameters. Use of the *Configuration Manager* is outside the scope of this document.

1. Install the *SuperWedge Configuration Manager* via its CAB file found on the product support CD.

   ➢ Use of *SuperWedge* is independent of the *Configuration Manager* so the *Configuration Manager* need not be installed. For applications that do not require on-going modifications, the *Configuration Manager* can be eliminated and *SuperWedge* used alone.

   ➢ The *Configuration Manager* is dependent on *SuperWedge*. If the *Configuration Manager* is used, *SuperWedge* MUST be installed and running when the *Configuration Manager* is started.

   Installation of the *Configuration Manager* automatically installs *SuperWedge* and a short-cut in the **`\Windows\StartUp`** folder.

2. Start *SuperWedge* by double tapping the large program icon in the device folder where you installed *SuperWedge* OR by restarting your device. Note: it is NOT necessary to restart your device, but you can if you prefer.

3. After double tapping the large program icon, the small *SuperWedge* icon will appear in the task tray. After a few seconds you should hear a double beep indicating *SuperWedge* successfully loaded and is ready for use. If an error message occurs, refer to Section 3.1.

2.2 Minimum Installation

The *SuperWedge* minimum installation is distributed as a ZIP file, `SWvXXXX.zip` where XXXX represents the version number. The files included in the ZIP file are listed in Table 2-1.

Table 2-1: Minimum Installation Distribution

Component	Folder	Description
`changes.txt`		Revision history.
`SuperWedge Technical Reference.pdf`		This document.
`Sw.exe`	**`\ARMV4Rel`**	Executable for Windows CE 4.2
`Sw.exe`	**`\ARMV4IRel`**	Executable for Windows CE 4.1
`swapi.h`	**`\Include`**	Header file for application program development.

Follow these steps to perform the minimum installation:

1. Unzip the *SuperWedge* package into a temporary folder on your PC.

2. Copy the appropriate `Sw.exe` from your PC to a destination folder on your device.

3. If you want *SuperWedge* to launch automatically when the unit is cold booted, create a shortcut in **`\Windows\StartUp`** on your device.

4. If you have an initialization file, `SuperWedge.ini`, copy it from your PC to the `Sw.exe` destination folder on your device OR to the **`\Windows`** folder.

5. Start *SuperWedge* by double tapping the large `Sw.exe` program icon in the device folder where you installed *SuperWedge*. Note: it is NOT necessary to restart your device.

6. After double tapping the large program icon, the small *SuperWedge* icon will appear in the task tray. After a few seconds you should hear a double beep indicating *SuperWedge* successfully loaded and is ready for use. If an error message occurs, refer to Section 3.1.

2.3 Additional Files

The following files are expected to be installed on the system and are not distributed with either of the *SuperWedge* installation packages.

Table 2-2: Additional Files

Component	Folder	Description
`sw_buffer_full.wav`	**`\Windows`**	Sound played when the *SuperWedge* internal buffer is full.
`sw_double_beep.wav`	**`\Windows`**	Sound played at *SuperWedge* initialization.
`sw_error.wav`	**`\Windows`**	Sound played during menu label setup if an invalid selection is made.
`sw_good_read.wav`	**`\Windows`**	Default sound played following successful data input.
`sw_menu_entry.wav`	**`\Windows`**	Sound played when a bar code menu label is scanned.

If one of these WAV files is missing, *SuperWedge* will use the Windows default sound in its place.

3 User Interface

In normal use, *SuperWedge* presents no User Interface (UI) and all incoming data is sent to the foreground application via keyboard or Application Program Interface (API) messages.

SuperWedge will display its pop-up menu when it is activated by single tapping the small taskbar icon if it is visible OR by double tapping the large program icon after *SuperWedge* has initially loaded.

The pop-up menu consists of 6 items as outlined in the following sections.

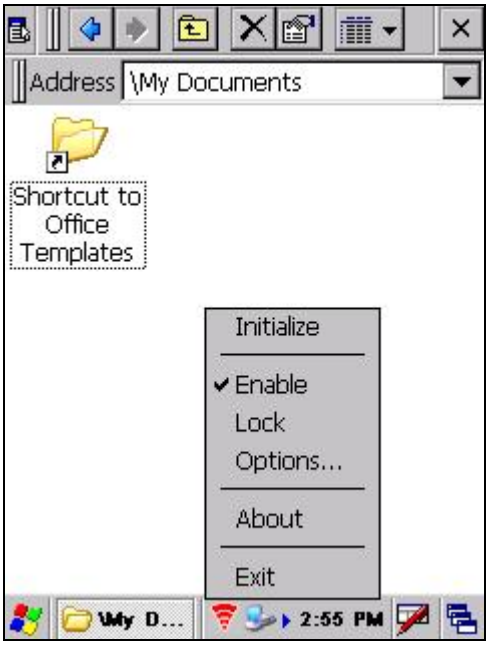


Figure 3-1: SuperWedge Pop-up Menu

3.1 Initialize

Initialization occurs when **Initialize** is selected from the pop-up menu.

During initialization all *SuperWedge* parameters are returned to the factory defaults, then the initial configuration file, `SuperWedge.ini`, is parsed. Refer to Section 4 for a description of the configuration file syntax and *SuperWedge* default settings.

If an error is encountered during INI parsing, the user is prompted with a message as shown in Figure 3-2. Following user acknowledgement of the error message, *SuperWedge* exits. The error in the configuration file needs to be corrected and *SuperWedge* restarted. All errors in the configuration file must be corrected before *SuperWedge* will successfully load and remain running.

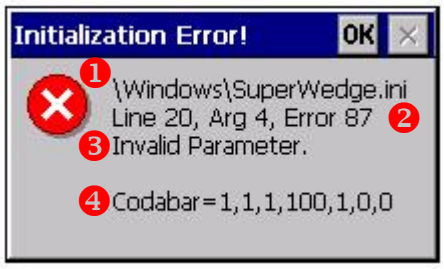


Figure 3-2: Initialization Error! Message Box

The text in position ❶ indicates the complete path and file name of the configuration file being parsed.

Line ❷ indicates the line number, argument number and error number that has been encountered. Line numbering begins with 1 and increments each time a carriage return is encountered. Argument 0 is the command name appearing to the left of the "=". Arguments to the right of the "=" begin with 1 and increment for each comma.

Line ❸ provides a definition of the error number.

The offending line of the configuration file is shown on Line ❹ in its entirety to aid in diagnosing the problem. In the above example, the value of 100 is invalid because it is out of range.

3.2 Enable

Selecting **Enable** toggles global *SuperWedge* enabling or disabling without exiting *SuperWedge*.

When *SuperWedge* is globally enabled, a check mark appears beside **Enable** and *SuperWedge* is ready to receive data from any enabled input source. When globally enabled, the taskbar icon looks like this: .

If there is no check mark beside **Enable**, *SuperWedge* is globally disabled. While disabled, the scanner will not fire and *SuperWedge* will not process data input from any source. When *SuperWedge* is disabled, the taskbar icon looks like this: .

3.3 Lock

To protect *SuperWedge* from inadvertent settings changes, *SuperWedge* can be locked. To lock *SuperWedge*, set the password via the INI or API using the "Password=" command. Once locked a user has to supply the correct password to gain access to the pop-up menu. After providing the correct password, access is granted to the pop-up menu; however, *SuperWedge* remains locked until specifically unlocked.

Figure 3-3: Password Entry Window

If a check mark appears beside the **Lock** menu item, *SuperWedge* is locked. If there is no check mark, *SuperWedge* is unlocked.

> ➢ The password is stored in human readable format in the *SuperWedge* configuration file. The password is not meant as a security feature. Should the password be forgotten, simply open the configuration file with a text editor.

3.4 Options...

Selecting **Options...** from the pop-up menu will launch the *SuperWedge Configuration Manager*.

> ➢ The *Configuration Manager* may not be present in all installations. If it is not installed and **Options...** is selected, an error message will occur.

Reference Section 2.1 for additional information regarding the *Configuration Manager*.

3.5 About

Select this menu item to display the About box, showing application name, version, and copyright information.

3.6 Exit

When **Exit** is selected from the pop-up menu, you will be prompted to confirm this action. If accepted, *SuperWedge* exits.

3.7 Command Line Parameters

SuperWedge is normally used without a command line parameter but does support use of a single command line parameter. This single command line parameter, -m, puts *SuperWedge* into Manufacturing Mode.

While in Manufacturing Mode, *SuperWedge* will recognize the default hot keys and will energize the attached scanning device but will not process any scanned data. This functionality is used primarily during the manufacturing process to set-up the scanning device using bar code menus. Typically a "default all" label or other set-up label is scanned to put the scanning device into a known state.

To exit Manufacturing Mode, tap the *SuperWedge* icon in the task tray and select **Exit**.

Option	Description
-m	Starts *SuperWedge* in Manufacturing Mode. This is a special operational mode used during the production process. Its use should not be necessary during normal operation.

The following labels can be scanned to return the scanning device to its default states. This should not be necessary under normal operating conditions.

>  These labels should only be scanned while in the Manufacturing Mode. Scanning these labels while in normal operational mode will cause *SuperWedge* to cease to operate properly.

When using *SuperWedge* with a Symbol laser scan engine, scan this label to return all scan engine parameters to the default values.

Default All Laser Parameters

When using *SuperWedge* with a Hand Held Products' IT5180SR-1212AOR imager, scan these labels to default all imager parameters.

MNUENA1.

DEFALT.

4 Settings

Initial *SuperWedge* parameters are customized via a configuration file, `SuperWedge.ini`. This file is read at *SuperWedge* start-up, when a user selects the **Initialize** option from the pop-up menu or when a connected application program issues the initialize command using the Application Program Interface (API).

SuperWedge first looks for `SuperWedge.ini` in the same directory as `Sw.exe`. If it is not found there, *SuperWedge* then looks in **`\Windows`**. If the INI is not found in either location, factory default values are used for all parameters. In addition, *SuperWedge* uses default settings for all parameters not specified in an INI. Because of this operation, it is not necessary to specify all *SuperWedge* parameters in the INI. Only those pertinent to the application that differ from the defaults need be included.

Should an error be encountered during INI file parsing, the user is prompted with an error message box. Refer to Section 3.1 for a description of the error box.

On-the-fly changes to *SuperWedge* parameters can be performed by a connected application via the API using commands with the same syntax as used in the INI. An error box does not occur for on-the-fly API parameter change requests. These errors are returned to the application program and should be checked and acted upon by the application program as appropriate.

4.1 INI File Syntax

`SuperWedge.ini` is an ASCII text file divided into sections containing options and modifiers.

- Section names begin with a [and end with a]. Section names are used to make the INI more easily human readable but are ignored by *SuperWedge*.
- General parameter syntax is *Option=modifer1*,*modifier2*,...,*modifierN*
- Options are case insensitive.
- A period (.) or an equals (=) may be used to separate an option from the modifiers. Multiple modifiers can be separated by an equals (=) or a comma (,). For example, ScanButton.1=1,235 is equivalent to ScanButton=1,1,235.
- If the option appears more than once in the file, then the last occurrence prevails.
- Parameters apply to all scan engine types unless otherwise indicated. Attempting to enable an option for a non-supported scan engine type will result in an error.
- Preambles, Postambles and Code ID characters are only added to the data if the input record does not match a programmed and enabled edit. Preambles, Postambles and Code ID characters are added to the input data as shown in Figure 4-1.

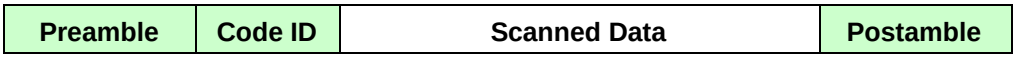


Figure 4-1: Output Data Record Format

4.1.1 Spacing and Line Termination

- Both Space (decimal code 32) and Horizontal Tab (decimal code 9) are acceptable spacing characters.
- Options cannot contain spacing characters.
- Lines are terminated by a CR (decimal code 13) and LF (decimal code 10) character. A new line is counted whenever a LF is encountered.
- Comments are introduced by a semicolon character (;) and terminate at the end of the line.

4.1.2 Numeric and String Values

- Numeric parameter values are decimal values unless otherwise indicated.
- In each section, the following values apply unless otherwise specified: 0 = Disable; 1 = Enable
- If a numeric parameter value is not specified, the previous setting is unchanged. This allows symbologies to be enabled (or disabled) with *SymbologyName*=1(or 0) without regard for the other options.
- If a string setting is not specified, the string is cleared.

4.1.3 Sample INI File

```
;Sample INI File for use with SuperWedge

[CONFIGURATION PROPERTIES]
Name       = SuperWedge Test INI
Description= This is a test INI.
TaskTrayIcon=1

[GENERAL]
Volume=3
Inter Character=0
Inter Function=0
NumericKeypad=0
Buffering=1
Output Edits Only=0
Pause Time=30
Good Read=4,1,\windows\latched.wav
Vibrate Time=50
LED Time=3
Menu Label Setup=1
Caps Lock Correction=1
Window=
Date Format=dddd',' MMMM dd',' yyyy
Time Format=hh':'mm':'ss tt

[SCANNER]
Scanner=1
Redundancy=0
Scanner Function Codes=1
```

```
Code ID Type=0
Aim Dot=0,10
ScanButton.1=1,235
ScanButton.2=0,233
ScanButton.3=0,234
ScanButton.4=1,236
Scanner Preamble=26,9
Scanner Postamble=13

[CUSTOM CODE ID]
Code ID.NoCode=78
Code ID.Aztec=65
Code ID.BooklandEan=66
Code ID.Codabar=67
Code ID.Code11=71
Code ID.Code16K=75
Code ID.Code32=84
Code ID.Code39=78
Code ID.Code49=70
Code ID.Code93=110
Code ID.Code128=111
Code ID.CompositeCode=99
Code ID.CouponCode=117
Code ID.DataMatrix=68
Code ID.EAN8=69
Code ID.EAN13=69
Code ID.Identicode2of5=105
Code ID.Industrial2of5=105
Code ID.Interleaved2of5=73
Code ID.ISBT128=111
Code ID.MaxiCode=109
Code ID.MicroPDF417=112
Code ID.MSIPlessey=77
Code ID.PDF417=80
Code ID.RSS14=82
Code ID.RSSLimited=82
Code ID.RSSExpanded=82
Code ID.TLC39=76
Code ID.TriopticCode39=79
Code ID.UCCEAN128=111
Code ID.UPCA=85
Code ID.UPCE=85
Code ID.UPCE1=85
Code ID.VIN=86

[AUX PORT]
AuxPort=1,9600,8,N,1,0
AuxPort Protocol=0
AuxPort PreAmble=
AuxPort PostAmble=13
AuxPort Function Codes=1
AuxPort Record Terminator=13
AuxPort Block Size=16

[RCODES]
EAN8=1,0,0
EAN13=1,1,0
```

```
UPCA=1,1,1,0,1,0
UPCE=1,1,1,0,0,0
UPCE1=1,1,1,0,0,0
Supplementals=2
RSS14=0
RSSLimited=0
RSSExpanded=0

[ICODES]
Codabar=1,0,2,48,5,0,0
Code11=1,1,4,55,1,0,1,0
Code128=1,0
Code39=1,0,1,48,1,0,0,0,1,1,0,1
Code93=1,1,4,55,1,0
Industrial2of5=1,1,4,48,12,0
Interleaved2of5=1,1,4,64,14,0,2,0,0
Isbt128=1
MsiPlessey=1,0,4,48,6,0,1,1,1
TriopticCode39=0
UccEan128=1

[2DCODES]
CompositeCode=0
MicroPDF417=0
PDF417=0
TLC39=0
Aztec=0
Code16K=0
Code49=0
DataMatrix=0
MaxiCode=0

[VK MAPPING]


[EDITS]
```

4.2 Configuration Properties

Section [CONFIGURATION PROPERTIES] in the INI.

Table 4-1: Configuration Properties

Parameter and Default Values	Description
Name=	User defined string. Not used by *SuperWedge*.
Author=	User defined string. Not used by *SuperWedge*.
Customer=	User defined string. Not used by *SuperWedge*.
Date=	User defined string. Not used by *SuperWedge*.
Version=	User defined string. Not used by *SuperWedge*.
Description=	User defined string. Not used by *SuperWedge*.
Password=	Locks *SuperWedge* to prevent inadvertent settings changes. The string can be up to 6 characters long. To clear the password, set "Password=" such that no characters follow the "=" other than a CR/LF.
TaskTrayIcon=1	Shows/hides the small task tray icon.

4.3 General Settings

Section [GENERAL] in the INI.

Table 4-2: General Settings

Parameter and Default Values	Description
Volume=7	Audible indication volume. 0=off, 1 (min) to 7 (max)
Inter Character=0	This parameter specifies the time to delay between transmission of emulated keystrokes for characters (non-function codes). The intercharacter delay appears after each character is emulated and gives the system time to perform other tasks. Valid values for this parameter are integers from 0 to 99 representing 0 to 990ms in 10ms increments.
Inter Function=0	This parameter specifies the time to delay between transmission of emulated keystrokes for function codes. The interfunction delay appears after each function code is emulated and gives the system time to perform other tasks. Valid values for this parameter are integers from 0 to 99 representing 0 to 990ms in 10ms increments.
NumericKeypad=0	Under normal conditions the number keys at the "top" of the keyboard are emulated by the keyboard wedge when entering numeric data. In the event that an application calls for the depression of the numeric keys on the right-hand keypad, this parameter should be enabled.

Parameter and Default Values	Description
Buffering=1	When this parameter is enabled (1), the wedge will buffer up to 32 records while data is being output. When disabled (0), the wedge will not accept another record until the previous record is completely output. If rapid scanning occurs, it is possible to fill the *SuperWedge* internal buffer. If this happens, an error beep will occur, and the current record will be lost.
Solicit Record=0,0,0,0	This parameter is not yet implemented; however, the inclusion of this parameter in an INI will not cause an error.
Output Edits Only=0	When this parameter is enabled (1), a record is only passed through *SuperWedge* if it matches one of the defined and enabled edits. When disabled (0), all records will be passed.
Pause Time=0	Defines the amount of time to delay when the {Pause} function is encountered. When ASCII 0x15 is found in the output record, it is interpreted as the {Pause} function. Valid values for this parameter are integers from 0 to 99 representing 0 to 9.9s in 100ms increments.
Good Read=1,0,	Good Read=*device,type,file* *device* Output device used for good read annunciation. 0 = none 1 = beeper 2 = speaker 3 = vibrator 4 = vibrator + beeper 5 = vibrator + speaker *type* 0 to use default WAV, **`\Windows\sw_good_read.wav`** 1 to use custom WAV file specified by *file*. *file* String specifying the WAV file to play following a successful read. Specify path, filename and extension.
Vibrate Time=50	Sets the length of time the vibrator is on following a successful read. The "Good Read" device must be set to use the vibrator for this setting to have any affect. Valid values for this parameter are integers from 0 to 500 representing 0 to 5s in 10ms increments.
LED Time=3	Sets the length of time the green LED is on following a

Parameter and Default Values	Description
	successful read. To set the programmable LED on time, use integers from 1 to 10 representing 1 to 10s in 1s increments. A value greater than 10 will leave the green LED on until the start of the next decode.
Menu Label Setup=0	When enabled (1), special menu labels can be used to instruct *SuperWedge* to perform functions. To disable scanning of menu labels, set this parameter to 0. Reference Section 7.7 for a list of menu labels.
Caps Lock Correction=1	When Caps Lock Correction is enabled (1), alpha characters in the output record will appear exactly as the input data appears, e.g. if "abc" is scanned, the output will be "abc" regardless of the caps lock key state. With Caps Lock Correction disabled (0), the current caps lock state determines the actual output, e.g. if "abc" is scanned and the caps lock key is on, the output will be "ABC" whereas with the caps lock key off, the output will be "abc".
Window=	Directs wedged data to a single, user specified window rather than to the active window. When wedge data are output, if the specified window does not have focus, it will be brought to the foreground. Window=*ClassName,WindowName* *ClassName* the window's class name. Case-sensitive. *WindowName* the window's title. Case-sensitive. Optional. For example, Window=WordPad,Doc1 Window=iExplore, Window=PTAB Application, When this parameter is set, the scanner is not energized unless the window is open. If SuperWedge is registered with an application via the API, this command is ignored. Default forces use of the window with current focus.

Parameter and Default Values	Description
Date Format=M'/'d'/'yyyy	Specifies the format for the {Date} function using a "format picture string". When ASCII 0x19 is found in the output record, it is interpreted as the {Date} function. Use the following elements to construct a format picture string. If you use spaces to separate the elements in the format string, these spaces will appear in the same location in the output string. The letters must be in uppercase or lowercase as shown in the table (for example, "MM" not "mm"). Characters in the format string that are enclosed in single quotation marks will appear in the same location and unchanged in the output string. Picture — Meaning d — Day of month as digits with no leading zero for single-digit days. dd — Day of month as digits with leading zero for single-digit days. ddd — Day of week as a three-letter abbreviation. The function uses the LOCALE_SABBREVDAYNAME value associated with the specified locale. dddd — Day of week as its full name. The function uses the LOCALE_SDAYNAME value associated with the specified locale. M — Month as digits with no leading zero for single-digit months. MM — Month as digits with leading zero for single-digit months. MMM — Month as a three-letter abbreviation. The function uses the LOCALE_SABBREVMONTHNAME value associated with the specified locale. MMMM — Month as its full name. The function uses the LOCALE_SMONTHNAME value associated with the specified locale. y — Year as last two digits, but with no leading zero for years less than 10. yy — Year as last two digits, but with leading zero for years less than 10. yyyy — Year represented by full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars both have five digit years. The "yyyy" pattern will show five digits for these two calendars, and four digits for all other supported calendars. yyyyy — Behaves identically to "yyyy". gg — Period/era string. The function uses the CAL_SERASTRING value associated with the specified locale. This element is ignored if the date to be formatted does not have an associated era or period string. Maximum formatted string length is 63 characters.

Parameter and Default Values	Description
Time Format=H':'mm':'ss	Specifies the format for the {Time} function using a "format picture string". When ASCII 0x20 is found in the output record, it is interpreted as the {Time} function. Use the following elements to construct a format picture string. If you use spaces to separate the elements in the format string, these spaces will appear in the same location in the output string. The letters must be in uppercase or lowercase as shown (for example, "ss", not "SS"). Characters in the format string that are enclosed in single quotation marks will appear in the same location and unchanged in the output string. Picture — Meaning h — Hours with no leading zero for single-digit hours; 12-hour clock. hh — Hours with leading zero for single-digit hours; 12-hour clock. H — Hours with no leading zero for single-digit hours; 24-hour clock. HH — Hours with leading zero for single-digit hours; 24-hour clock. m — Minutes with no leading zero for single-digit minutes. mm — Minutes with leading zero for single-digit minutes. s — Seconds with no leading zero for single-digit seconds. ss — Seconds with leading zero for single-digit seconds. t — One character time-marker string, such as A or P. tt — Multicharacter time-marker string, such as AM or PM. For example, to get the time string "11:29:40 PM" use the following picture string: "hh':'mm':'ss tt" Maximum formatted string length is 31 characters.

4.4 Scanner Settings

Section [SCANNER] of the INI.

Table 4-3: Scanner Settings

Parameter and Default Values	Description
Scanner=1	Enables (1) or disables (0) bar code scanning via the integrated scanner.
Scanner Type=0	Verifies integrated scan engine matches expected. This parameter does not change the installed scan engine type. Instead, it compares the actual scan engine hardware with the expected type provided by this parameter. If the actual does not match the expected, an error occurs. Valid scan engine types are as follows: 0=None 1=SE824 2=SE923 3=SE1223 4=SE2223 5=SE1524ER 6=SE955 7=SE1224 8=IT5180 Default depends on installed scan engine. When performing a Status check, *SuperWedge* uses this parameter to report the installed scan engine type.
Redundancy=0	When this parameter is enabled, a bar code must be successfully scanned in both directions (forward and reverse) before being decoded. This parameter is ignored for non-laser based scan engines, e.g. IT5180.
Scanner Function Codes=0	If function codes are embedded within the bar codes being read and the keyboard function is desired, this parameter must be enabled (1). See Section 7.1 for a list of keyboard functions and virtual key codes associated with each function. When disabled (0), function codes are stripped from the output record only if a data edit is not performed.

Parameter and Default Values	Description
Aim Dot=0,10	Sets the duration the aimer is seen before a scan attempt begins. Aim Dot=*enable,duration* *enable* 1 to enable, 0 to disable For the SE1524 laser scan engine, the aimer is enabled by default. For all other scanners, the aimer is disabled by default. *duration* For laser scan engines, valid values for this parameter are integers from 0 to 99 representing 0 to 9.9sec in 100ms increments. For the IT5180 imager, valid values for this parameter are integers from 0 to 40 representing 0 to 4.0sec in 100ms increments. For laser scan engines, a single dot is presented before scanning begins. For the IT5180 imager, during the delay time the green aiming light will appear, but the decode attempt will not occur until the delay time is over.
ScanButton.1=1,233 ScanButton.2=1,234 ScanButton.3=1,235 ScanButton.4=1,236	Allows assigning up to 4 keyboard keys as the "hotkey" for activation of the integrated laser. ScanButton.*n=enable,vk* *n* Key number. Valid range is 1 through 4. *enable* 1 to enable. 0 to disable *vk* Virtual key code to monitor as the "hotkey". This is a decimal value. Note: Most references give virtual key codes in hex. Valid range is 0 to 255. Once assigned to *SuperWedge*, this virtual key cannot be assigned as a hotkey for another application. The default enables the left scan button, the right scan button, the center scan button and the trigger.
Scanner Preamble=	Defines the ASCII keys to add to the beginning of non-edited

Parameter and Default Values	Description
	scanned data. Up to 16 ASCII keys can be defined. Values are separated by a comma.
Scanner Postamble=	Defines the ASCII keys to add to the end of non-edited scanned data. Up to 16 ASCII keys can be defined. Values are separated by a comma.
Code ID Type=0	Specifies the symbology identifier. The Code ID is a single character that identifies the code type of a scanned bar code. The Code ID character is inserted between the prefix character(s) (if programmed) and the decoded data of a non-edited record. Valid settings are 0=None 1=AIM 2=Compsee 3=Symbol 4=Custom 5=Hand Held Products Reference Section 7.5 for a list of the code identification characters. Reference Section 4.6 for a description of the "Custom" settings.

4.5 Auxiliary Port Settings

Auxiliary Port input can come from hard-wired sources such as magnetic stripe readers and scales or from wireless (Bluetooth) devices such as bar code readers.

Auxiliary Port settings are found in Section [AUX PORT] of the *SuperWedge* configuration file.

Table 4-4: Auxiliary Port Settings

Parameter and Default Values	Description
AuxPort=0,9600,8,N,1,0	Specifies Auxiliary Port settings AuxPort=*port,baud,databits,parity,stopbits,flowcontrol* *port* 0 to disable, 1 through 9 for general purpose RS-232. Cannot set for COM2 since it is used for the internal scanner port. *baud* 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 *databits* Valid settings are 7 or 8 *parity* N = none O = odd E = even *stopbits* Valid settings are 1 or 2. *flowcontrol* 0 = none 1 = XON/XOFF 2 = RTS/CTS

Parameter and Default Values	Description
AuxPort Protocol=0	Specifies the Aux Port communication method. Valid settings for this parameter are 0 = Record Mode 5 = Block Mode Reference "Record Terminator" for more information on Record Mode operation. Refer to "AuxPort Block Size" for additional information on Block Mode operation.
AuxPort PreAmble=	Defines the ASCII keys to add to the beginning of non-edited data received from the aux port. Up to 16 ASCII keys can be defined. Values are separated by a comma.
AuxPort PostAmble=	Defines the ASCII keys to add to the end of non-edited data received from the aux port. Up to 16 ASCII keys can be defined. Values are separated by a comma.
AuxPort Function Codes=0	If function codes are embedded within the aux port data and the keyboard function is desired, this parameter must be enabled (1). See Section 7.1 for a list of keyboard functions and virtual key codes associated with each function. When disabled (0), function codes are stripped from the output record.
AuxPort Record Terminator=13	When AuxPort Protocol is set for record mode, aux port data is buffered until the record terminator character is received. Once received the record terminator will be discarded and the record will be processed by *SuperWedge*. Valid values are 1 through 254. Maximum record size (including terminator) is 256.
AuxPort Block Size=16	When AuxPort Protocol is set for block mode, buffered aux port data are passed for processing when the number of received characters equals the AuxPort Block Size. The block size can be set for integer values from 1 to 255.

4.6 Custom Code Identification Settings

Custom Code Identification characters can be assigned for existing application compatibility when one of the pre-defined Code ID sets does not suffice. "Code ID Type" must be set to custom for the [CUSTOM CODE ID] settings to have an affect.

The code type modifier is required. The ID modifier is optional. If the ID modifier is not specified, it is cleared and returned to its default. Reference Section 7.1 for the default listing of keys.

Table 4-5: Custom Code ID Settings

Parameter and Default Values	Description
Code ID.NoCode=	Valid range for all Custom Code ID settings is 33 through 126.
Code ID.Aztec=	
Code ID.BooklandEan=	
Code ID.Codabar=	
Code ID.Code11=	
Code ID.Code16K=	
Code ID.Code32=	
Code ID.Code39=	
Code ID.Code49=	
Code ID.Code93=	
Code ID.Code128=	
Code ID.CompositeCode=	
Code ID.CouponCode=	
Code ID.DataMatrix=	
Code ID.EAN8=	
Code ID.EAN13=	
Code ID.Identicode2of5=	
Code ID.Industrial2of5=	
Code ID.Interleaved2of5=	
Code ID.ISBT128=	
Code ID.MaxiCode=	
Code ID.MicroPDF417=	
Code ID.MSIPlessey=	
Code ID.PDF417=	
Code ID.RSS14=	
Code ID.RSSLimited=	
Code ID.RSSExpanded=	
Code ID.TLC39=	

Parameter and Default Values	Description
Code ID.TriopticCode39=	
Code ID.UCCEAN128=	
Code ID.UPCA=	
Code ID.UPCE=	
Code ID.UPCE1=	
Code ID.VIN=	Code ID for Vehicle Identification Number recognized bar code labels.

As an example of Custom Code ID use, suppose an existing application requires code identification per Intermec definition. Since this is not one of the pre-defined Code ID sets, custom IDs can be set to accommodate this application requirement.

4.7 Retail Codes Settings

Section [RCODES] of the INI.

Table 4-6: Retail Code Settings

Parameter and Default Values	Description
EAN8=1,0,0	EAN8=*enable,ToEan13,ToGtin* *enable* *ToEan13* When enabled, this parameter adds five leading zeros to decoded EAN-8 symbols to make them compatible in format to EAN-13 symbols. Code type is changed from EAN-8 to EAN-13. *ToGtin* When enabled, this parameter adds six leading zeros to decoded EAN-8 symbols. Code type is changed from EAN-8 to Interleaved 2 of 5.
EAN13=1,0,0	EAN13=*enable,isbn,ToGtin* *enable* *isbn* Enables or disables Bookland EAN. *ToGtin* When enabled, this parameter adds a leading zero to decoded EAN-13 symbols. Code type is changed from EAN-13 to Interleaved 2 of 5.
UPCA=1,1,1,0,0,0	UPCA=*enable,CdXmit,NumSysXmit,CountryCode,CouponCode, ToGtin* *enable* *CdXmit* This option selects whether or not to transmit the check digit. *NumSysXmit* This option selects whether or not to transmit the number system character. *CountryCode* This option selects whether or not to transmit the country code

Parameter and Default Values	Description
	character. *NumSysXmit* must be enabled for this option to have significance. *CouponCode* When enabled, this parameter decodes UPC-A, UPC-A with 2 supplemental characters, UPC-A with 5 supplemental characters, and UPC-A/EAN128 bar codes. Autodiscriminate UPC/EAN Supplementals must be enabled. *ToGtin* When enabled, this parameter adds two leading zeros to decoded UPC-A symbols. Code type is changed from UPC-A to Interleaved 2 of 5.
UPCE=1,1,1,0,0,0	UPCE=*enable,CdXmit,NumSysXmit,CountryCode,ToUpca, ToGtin* *enable* *CdXmit* This option selects whether or not to transmit the check digit. *NumSysXmit* This option selects whether or not to transmit the number system character. *CountryCode* This option selects whether or not to transmit the country code character. *NumSysXmit* must be enabled for this option to have significance. *ToUpca* Enables conversion of UPC-E (zero suppressed) decoded data to UPC-A format before transmission. After conversion, data follows UPC-A format and is affected by UPC-A programming selections (e.g. Check Digit). *ToGtin* When enabled, this parameter expands the data to UPC-A format and adds two leading zeros. Code type is changed from UPC-E to Interleaved 2 of 5.
UPCE1=0,1,1,0,0,0	UPCE1=*enable,CdXmit,NumSysXmit,CountryCode,ToUpca, ToGtin* *enable* *CdXmit*

Parameter and Default Values	Description
	This option selects whether or not to transmit the check digit. *NumSysXmit* This option selects whether or not to transmit the number system character. *CountryCode* This option selects whether or not to transmit the country code character. *NumSysXmit* must be enabled for this option to have significance. *ToUpca* Enables conversion of UPC-E1 (zero suppressed) decoded data to UPC-A format before transmission. After conversion, data follows UPC-A format and is affected by UPC-A programming selections (e.g. Check Digit). *ToGtin* When enabled, this parameter expands the data to UPC-A format and adds two leading zeros. Code type is changed from UPC-E to Interleaved 2 of 5.
Supplementals=0	Sets the type of UPC/EAN supplemental decoding performed. Valid values are 0=Ignore - the scanner decodes UPC/EAN symbols but ignores any supplemental characters. 1=Require - the scanner only decodes UPC/EAN symbols with supplemental characters. 2=Autodiscriminate - the scanner decodes UPC/EAN symbols with or without supplemental characters.
RSS14=0	Enables/disables decoding of RSS-14 bar codes. Supported by the SE824, SE955, SE1224, SE2223, and IT5180.
RSSLimited=0	Enables/disables decoding of RSS Limited bar codes. Supported by the SE824, SE955, SE1224, SE2223, and IT5180.
RSSExpanded=0	Enables/disables decoding of RSS Expanded bar codes. Supported by the SE824, SE955, SE1224, SE2223 and IT5180.

4.8 Industrial Codes Settings

The Industrial codes settings are found in Section [ICODES] of the *SuperWedge* configuration file.

Many of the Industrial codes allow programming acceptable code lengths. The length of a code refers to the number of characters (i.e., human readable characters), including check digit(s) the code contains. Lengths may be set for any length, one or two discrete lengths, or lengths within a specific range.

- Any Length - This option allows decoding symbols containing any number of characters.
- Length Within Range - This option limits decodes to only those symbols within a specified range.
- One Discrete Length - This option limits decodes to only those symbols containing a selected length.
- Two Discrete Lengths - This option limits decodes to only those symbols containing either of two selected lengths.

Table 4-7: Industrial Code Settings

Parameter and Default Values	Description
Codabar=1,1,5,48,5,0,0	Codabar=*enable,type,min,max,d1,d2,startstop* *enable* Enables/disables decoding of Codabar bar codes. *type* 0=variable length 1=range 2=discrete *min* Range minimum. Valid range is 2 to 48. *type* must be set to "range" for this parameter to have significance otherwise it is ignored. *max* Range maximum. Valid range is 2 to 48. *type* must be set to "range" for this parameter to have significance otherwise it is ignored. *d1* Discrete length #1. Valid range is 0 to 48 where 0 means to ignore. *type* must be set to "discrete" for this parameter to have significance otherwise it is ignored. *d2* Discrete length #2. Valid range is 0 to 48 where 0 means to ignore. *type* must be set to "discrete" for this parameter to have significance otherwise it is ignored. *startstop* Specifies the start/stop character processing 0=None 1=NOTIS - strip start/stop characters 2=CLSI - strip start/stop characters and insert a space after the first, fifth and tenth characters of a 14 character Codabar symbol Note: lengths do not include start/stop characters

Parameter and Default Values	Description
Code11=0,1,4,55,1,0,2,0	Code11=*enable,type,min,max,d1,d2,CdVerify,CdXmit* *enable* Enables/disables decoding of Code 11 bar codes. *type* 0=variable length 1=range 2=discrete *min* Range minimum. Valid range is 1 to 64. *type* must be set to "range" for this parameter to have significance otherwise it is ignored. *max* Range maximum. Valid range is 1 to 64. *type* must be set to "range" for this parameter to have significance otherwise it is ignored. *d1* Discrete length #1. Valid range is 0 to 64 where 0 means to ignore. *type* must be set to "discrete" for this parameter to have significance otherwise it is ignored. *d2* Discrete length #2. Valid range is 0 to 64 where 0 means to ignore. *type* must be set to "discrete" for this parameter to have significance otherwise it is ignored. *CdVerify* This option allows the scanner to check the integrity of all Code 11 symbols. This selects the check digit mechanism for the decoded Code 11 bar code. The options are to check for one check digit or check for two check digits. 1=verify 1 check digit, 2=verify 2 check digits *CdXmit* This option selects whether or not to transmit the check digit(s). Supported by SE824, SE955, SE1224, SE1524ER, and IT5180.
Code128=1,0	Code128=*enable,vin*

Parameter and Default Values	Description
	enable Enables/disables decoding of Code 128 bar codes. *vin* Enables Vehicle Identification Number (VIN) verification. If the decoded data matches the VIN format, the code type is changed from Code 128 to VIN. Reference Section 7.4. Verification is based on the following 1. Data must be 17 or 18 characters long. 2. Data must not contain any invalid characters (non-numeric and non-alphabetic). 3. Check digit calculation must match check digit character.
Code39=1,1,2,48,1,0,0,0,1,0,0,0	Code39=*enable,type,min,max,d1,d2,CdVerify,CdXmit,FullAscii, code32,code32prefix,vin* *enable* Enables/disables decoding of Code 39 bar codes. *type* 0=variable length 1=range 2=discrete *min* Range minimum. Valid range is 1 to 48. *type* must be set to "range" for this parameter to have significance otherwise it is ignored. *max* Range maximum. Valid range is 1 to 48. *type* must be set to "range" for this parameter to have significance otherwise it is ignored. *d1* Discrete length #1. Valid range is 0 to 48 where 0 means to ignore. *type* must be set to "discrete" for this parameter to have significance otherwise it is ignored. *d2* Discrete length #2. Valid range is 0 to 48 where 0 means to ignore. *type* must be set to "discrete" for this parameter to have significance otherwise it is ignored.

Parameter and Default Values	Description
	CdVerify When this option is enabled, the scanner checks the integrity of all Code 39 symbols. Only those Code 39 symbols which include a modulo 43 check digit are decoded. Only enable this feature if your Code 39 symbols contain a modulo 43 check digit. *CdXmit* This option selects whether or not to transmit the check digit. *FullAscii* Enables or disables Code 39 Full ASCII. Code 39 Full ASCII is a variant of Code 39 which pairs characters to encode the full ASCII character set. Refer to Section 7.1 for the mapping of Code 39 characters to ASCII values. Code 39 Full ASCII should not be enabled at the same time as Code 32 or Trioptic Code 39. *code32* Enables or disables converting Code 39 to Code 32. Code 32 is a variant of Code 39 used by the Italian pharmaceutical industry. Code 32 should not be enabled at the same time as Code 39 Full ASCII or Trioptic Code 39. *code32prefix* Enable this parameter to add the prefix character "A" to all Code 32 bar codes. *vin* Enables Vehicle Identification Number (VIN) verification. If the decoded data matches the VIN format, the code type is changed from Code 39 to VIN. Reference Section 7.4. Verification is based on the following 1. Data must be 17 or 18 characters long. 2. Data must not contain any invalid characters (non-numeric and non-alphabetic). 3. Check digit calculation must match check digit character.
Code93=1,1,4,55,1,0,0	Code93=*enable,type,min,max,d1,d2* *enable* Enables/disables decoding of Code 93 bar codes. *type* 0=variable length 1=range

Parameter and Default Values	Description
	2=discrete *min* Range minimum. Valid range is 1 to 64. *type* must be set to "range" for this parameter to have significance otherwise it is ignored. *max* Range maximum. Valid range is 1 to 64. *type* must be set to "range" for this parameter to have significance otherwise it is ignored. *d1* Discrete length #1. Valid range is 0 to 64 where 0 means to ignore. *type* must be set to "discrete" for this parameter to have significance otherwise it is ignored. *d2* Discrete length #2. Valid range is 0 to 64 where 0 means to ignore. *type* must be set to "discrete" for this parameter to have significance otherwise it is ignored.
Industrial2of5=0,2,4,48,12,0	Industrial2of5=*enable,type,min,max,d1,d2* *enable* Enables/disables decoding of Industrial 2 of 5 bar codes. *type* 0=variable length 1=range 2=discrete *min* Range minimum. Valid range is 1 to 48. *type* must be set to "range" for this parameter to have significance otherwise it is ignored. *max* Range maximum. Valid range is 1 to 48. *type* must be set to "range" for this parameter to have significance otherwise it is ignored. *d1* Discrete length #1. Valid range is 0 to 48 where 0 means to ignore. *type* must be set to "discrete" for this parameter to have significance otherwise it is ignored.

Parameter and Default Values	Description
	d2 Discrete length #2. Valid range is 0 to 48 where 0 means to ignore. *type* must be set to "discrete" for this parameter to have significance otherwise it is ignored. Also enables decoding of Identicode 2 of 5 labels.
Interleaved2of5=1,2,4,64,14,0,0,0,0 or I2of5=1,2,4,64,14,0,0,0,0	Interleaved2of5=*enable,type,min,max,d1,d2,CdVerify,CdXmit, ToEan13* *enable* Enables/disables decoding of Interleaved 2 of 5 bar codes. *type* 0=variable length 1=range 2=discrete *min* Range minimum. Valid range is 2 to 64. *type* must be set to "range" for this parameter to have significance otherwise it is ignored. *max* Range maximum. Valid range is 2 to 64. *type* must be set to "range" for this parameter to have significance otherwise it is ignored. *d1* Discrete length #1. Valid range is 0 to 64 where 0 means to ignore. *type* must be set to "discrete" for this parameter to have significance otherwise it is ignored. *d2* Discrete length #2. Valid range is 0 to 64 where 0 means to ignore. *type* must be set to "discrete" for this parameter to have significance otherwise it is ignored. *CdVerify* Specifies the check digit algorithm to use. 0=None 1=USS (Uniform Symbology Specification) 2=OPCC (Optical Product Code Council) *CdXmit*

Parameter and Default Values	Description
	This option selects whether or not to transmit the check digit. *ToEan13* This parameter converts a 14 character Interleaved 2 of 5 code into EAN-13, and transmits it as EAN-13. The code must have a leading zero and a valid EAN-13 check digit.
Isbt128=1	Enables/disables decoding of ISBT-128 bar codes. For the IT5180, enabling this parameter also enables ISBT-128 concatenation. The laser scan engines are not capable of concatenation.
MsiPlessey=0,0,4,48,6,0,1,0,0	MsiPlessey=*enable,type,min,max,d1,d2,NumCd,CdXmit,CdAlgor* *enable* Enables/disables decoding of MSI Plessey bar codes. *type* 0=variable length 1=range 2=discrete *min* Range minimum. Valid range is 4 to 48. *type* must be set to "range" for this parameter to have significance otherwise it is ignored. *max* Range maximum. Valid range is 4 to 48. *type* must be set to "range" for this parameter to have significance otherwise it is ignored. *d1* Discrete length #1. Valid range is 0 to 48 where 0 means to ignore. *type* must be set to "discrete" for this parameter to have significance otherwise it is ignored. *d2* Discrete length #2. Valid range is 0 to 48 where 0 means to ignore. *type* must be set to "discrete" for this parameter to have significance otherwise it is ignored. *NumCd* Specifies the number of check digits used by your labels. Valid values are 1 or 2.

Parameter and Default Values	Description
	CdXmit This option selects whether or not to transmit the check digit(s). *CdAlgor* Specifies the check digit algorithm to use for the 2nd check digit. 0=MOD10 1=MOD11
TriopticCode39=0	Enables/disables decoding of Trioptic Code 39 bar codes. Trioptic Code 39 should not be enabled at the same time as Code 39 Full ASCII or Code 32.
UccEan128=1	Enables/disables decoding of UCC/EAN-128 bar codes.

4.9 2D Codes Settings

Section [2DCODES]

Table 4-8: 2D Codes Settings

Parameter and Default Values[3]	Description
CompositeCode=0	Enables/disables decoding of Composite Code bar codes.[1] When Composite Codes are enabled, the ability to scan UPC and EAN labels without a 2D component will be adversely affected.
MicroPDF417=0	Enables/disables decoding of Micro PDF-417 bar codes.[1]
PDF417=1	Enables/disables decoding of PDF-417 bar codes.[1]
TLC39=0	Enables/disables decoding of TLC39 bar codes.[1]
Aztec=1	Enables/disables decoding of Aztec bar codes.[2]
Code16K=0	Enables/disables decoding of Code 16K bar codes.[2]
Code49=0	Enables/disables decoding of Code 49 bar codes.[2]
DataMatrix=1	Enables/disables decoding of Data Matrix bar codes.[2]
MaxiCode=1	Enables/disables decoding of MaxiCode bar codes.[2]

Notes

1. Supported by SE2223 and IT5180 only.
2. Supported by IT5180 only.
3. Default values shown are for supported scan engine types. These symbologies default to "0" (disabled) when an unsupported scan engine is utilized.

4.10 Virtual Key Translation

The virtual key translation settings are located in the [VK MAPPING] section of the *SuperWedge* configuration file. The virtual key map settings define the virtual key(s) that will be emulated by *SuperWedge* for any decoded ASCII character.

It is not necessary to define all of the key table entries. Only those that differ from default are necessary. Reference Section 7.1 for a complete list of the default settings.

Table 4-9: Virtual Key Translation Settings

Parameter and Default Values	Description
Key.1=20,0	Key.*n*=*vk*,*flags* *n* ASCII key code. Valid range is 1 to 254 excluding 21. This modifier is required to be present. *vk* Virtual key code. This is a decimal value. **Note:** Most references give virtual key codes in hex. Valid range is 0 to 255. This is an optional modifier. If this modifier is not specified, *vk* is set to the default. *flags* This is optional modifier. If this modifier is not specified, *flags* is set to the default. 0 If *flags* is set to 0, the virtual key is pressed and released without pressing any modifier keys 1 To press the Shift key in conjunction with a virtual key, set the "Shift" flag. 2 To press the Ctrl key in conjunction with a virtual key, set the "Ctrl" flag. 4 To press the Alt key in conjunction with a virtual key, set the "Alt" flag. 8 To press and hold a key without releasing it, set the "No Release" flag. 16 To emulate the release of a key without emulating a press, set the "No Press" flag.
Key=1,20,0	Alternate format

4.11 Data Validation and Manipulation

One of the most powerful features of *SuperWedge* is its ability to qualify data input and transform it into a different format for output. This data validation and manipulation is referred to as "editing" and the parameters that specify the formats are called "edits". Edits are defined in the *SuperWedge* configuration file under the section [EDITS].

Up to 99 individual and independent edits can be programmed. Edits are performed in numerical order. As soon as the input record is validated on data type, length and recognition field, the edit is performed and the edited record is returned. No further edits are performed on the data.

To understand how edits are specified, it is important to understand two edit concepts: Fields and Current Cursor Position.

The input data record consists of individual characters. A field can be an individual character or group of consecutive characters. For example, if a data record consists of the data "1234567" and we wish to group this into two fields -- one field with three characters and one with four -- the fields are assigned as:

Field 1			Field 2			
1	2	3	4	5	6	7

Once the input record is broken into fields, the output can act upon the fields in any order. The field concept permits data rearrangement, field repetition and entire field deletion.

The Current Cursor Position concept is used during input record validation as well as output record formation. For record validation, the current cursor position always begins at the leftmost character in the record and advances one character to the right for each character validated. During output record formation, the cursor position always starts at the leftmost character of the current field and advances one position to the right for each character added to the output record.

Table 4-10: Edit Settings

Parameter and Default Values	Description
Edit.*n*=0,0,0,,	Edit.*n*=*enable,DataType,InputLength,Recognition,Operation* The Edit Number, *n*, is a required modifier. All of the others are optional. If none of the others is specified, they are all set to the defaults. *n* Edit number. Valid range is 1 to 99. *enable* Enables (1) or disables (0) the edit specified by *n*. *DataType* Reference Section 7.4 for a complete list of data types. *InputLength* 0 means variable length

Parameter and Default Values	Description
	1 to 1024 fixed length *Recognition* The input data filter specifier string. *Operation* The output data format specifier string. A programmed edit can be temporarily disabled by issuing a command similar to "Edit.1=0". To re-enable the previously programmed edit parameters, issue "Edit.1=1".
Edit=*n,enable,DataType, InputLength,Recognition,Operation*	Alternate format

4.11.1 Edit Settings Details

Input validation is performed on the data type, data length and recognition specifier. The data type filter allows the edit to be selectively applied to specific bar code symbologies or to data sources. Reference Section 7.4 for a list of code types. Pay particular attention to the special edit code types SW_DATATYPE_ANY and SW_DATATYPE_SCANNER.

Recognition specifiers

"*string*"	Character(s) at current cursor position(s) must match *string*.
A	Character at current cursor position must be an Alpha character (A-Z or a-z).
N	Character at current cursor position must be a Numeric character (0-9).
L	Character at current cursor position must be an Alphanumeric character (A-Z, a-z or 0-9).
?	Character at current cursor position can be any character.
*	Ignores an unknown number of characters and moves the cursor to the character position of whatever comes next in the recognition specifier.
|	Inserts a Field separator. Up to 31 fields can be specified.

Operation specifiers

X	Deletes the character at current position.
S	Deletes the character at the current position and subsequent matching characters until a non-matching character is encountered.
*	Copies the character at the current cursor position as is.
@	Copies all data as is from current position through the end of the current field.
"*string*"	Inserts *string* at current position.
|*dd*	Moves cursor to beginning of Field *dd* where *dd* is 1 to 31 and begins output operations there. If a field specifier is not given, field 1 is assumed.
R	Repeats the previous character.

The ***** and the **"** operators have different meanings depending on whether they are used as recognition specifiers or operation specifiers.

Non-printable characters can be used in a specifier *string* through the use of the % operator.

%ddd Represents a non-printable character, e.g. %13 is a <CR>. Must also be used for embedding percents (%) and double quotes ("). Acceptable range is 1 to 255.

4.11.2 Edit Examples

Table 4-11: Edit Examples

Bar code	Size	Recognition	Operation	Result
A1B2C3	6	ANANAN	*X*X*X	ABC
PN-1245	7	AA"-"NNNN	"F"*******"%13"	FPN-1245<CR>
A1B2C3	6	LLLLLL	*RRX*RRX*RRX	AAABBBCCC
1A2B3C	6	N|A|N|A|N|A	|2*|1*|4*|3*|6*|5*	A1B2C3
TEST-SHEET	0	*|"-"|*	|3@","|1@	SHEET,TEST
0001234	0	"0"*	S@	1234

Suppose you have a Code 128 bar code label that contains the data "123ABCD" and you want to perform an edit on the record. Here is an example of what you can do

Edit.1=1,3,7,NNN|AAAA,"Easy as "|2***|1@

Data

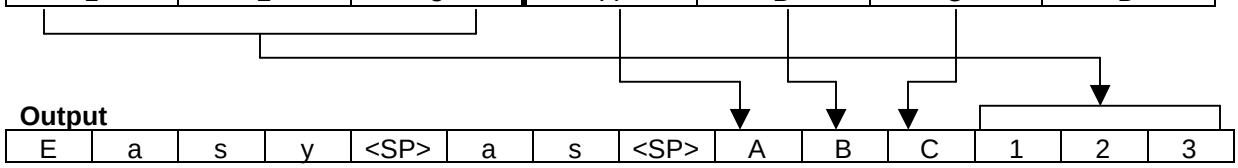
1	2	3	A	B	C	D

Input Recognition Specifier Meaning

Field 1			Field 2			
N	N	N	A	A	A	A

Output Operation Specifier Meaning

Field 1			Field 2			
1	2	3	A	B	C	D



Output

E	a	s	y	<SP>	a	s	<SP>	A	B	C	1	2	3

4.12 Special Commands

Allowed during initialization (reading an INI) or via an API command.

Command	Description
DefaultAll	Returns all parameters to factory defaults.
DefaultEdits	Returns all Edits to defaults.

5 Application Program Interface (API)

The *SuperWedge* Application Program Interface (API) provides a simple, convenient method for application programs to communicate with *SuperWedge*. Communication between the application program and *SuperWedge* is done via Windows messages and includes three features:

- Functionality to change configuration settings “on-the-fly” without launching the *Configuration Manager* or reading the configuration file.
- Provides a programmatic interface of the functionality currently done manually via the *SuperWedge* user interface.
- Delivers data provided by *SuperWedge* directly to a registered application.

Because the API is implemented using Windows messages, only one header file, `swapi.h`, is required when using the C/C++ API. The "functions" described in the following sections are really macros defined in the API header file so there is no need to include a static library (LIB) or dynamic link library (DLL) when using the *SuperWedge* API. The header file is distributed as part of the minimum installation and can be found on the product support CD.

Although the header file is intended for C/C++ programs only, the concepts can be extended to other programming languages fairly easily. A class library for inclusion in Visual Basic or Visual C# is provided on the product support CD. The use of the class library is outside the scope of this document.

In order for an application to make on-the-fly changes or to issue other commands, the application must be registered with *SuperWedge*. The registration process is referred to as "connecting to" *SuperWedge*. When using the API, connection with *SuperWedge* is required.

An optional feature of the API, is "buffered mode". In normal operation, *SuperWedge* data is output via the keyboard interface as "wedged" data. In some programming instances, it is more desirable to receive the data directly into a buffer.

Regardless of the programming language or mode usage, the general procedure to follow when using the API is as follows:

At program start-up

1. Register application with *SuperWedge* via *SWRegisterClient()*. This also verifies that *SuperWedge* is running.
2. Call *SwGetVersion()* to verify *SuperWedge* version contains API support.
3. Put into buffered mode if desired via *SWSetBufferedMode()*.
4. Status check if desired.
5. Send default all command.
6. Send parameter changes.

During program execution

1. Receive and process Windows message (if in buffered mode).
2. Copy data from shared buffer (if in buffered mode).
3. Clear flag (if in buffered mode).

At program exit

Deregister using *SWDeRegisterClient()*.

5.1 Connection APIs

SWRegisterClient

Register client window handle with *SuperWedge*. Called to connect with *SuperWedge* for performing API calls.

```
SWRegisterClient(
HWND hClientWnd,
HWND hSuperWedgeWnd,
LRESULT lResult
);
```

Parameters

hClientWnd

[in] Handle to the window whose window procedure will receive messages from *SuperWedge*.

hSuperWedgeWnd

[in] Variable to receive *SuperWedge* window handle.

lResult

[in] Variable to receive API call result

Return Values

lResult

ERROR_SUCCESS indicates success
ERROR_FAILURE indicates failure

hSuperWedgeWnd

if ERROR_SUCCESS retrieved *SuperWedge* window

Remarks

SWDeRegisterClient

Deregister client window handle with *SuperWedge*. Called to disconnect from *SuperWedge*.

```
SWDeRegisterClient(
HWND hClientWnd,
HWND hSuperWedgeWnd,
LRESULT lResult
);
```

Parameters

hClientWnd
client window handle

hSuperWedgeWnd
SuperWedge window handle

lResult
variable to receive API call result

Return Values

lResult
ERROR_SUCCESS indicates success
ERROR_FAILURE indicates failure

Remarks

SWGetVersion

Retrieve *SuperWedge* version. *SuperWedge* versions starting at v2.000 support the API calls.

```
SWGetVersion(
HWND hSuperWedgeWnd,
DWORD dwVersion,
LRESULT lResult
);
```

Parameters

hSuperWedgeWnd
SuperWedge window handle

dwVersion
variable to receive version number

lResult
variable to receive API call result

Return Values

lResult
ERROR_SUCCESS indicates success
ERROR_FAILURE indicates failure

dwVersion
if ERROR_SUCCESS, retrieved *SuperWedge* version number

Remarks

The version number is returned as a hexadecimal number format: <*major version*>.<*minor version*>. For example, a return value of 0x2003 corresponds to major version = 2, minor version = 003

SWSetBufferedMode

Enable or disable *SuperWedge* buffered mode operation.

```
SWSetBufferedMode(
HWND hSuperWedgeWnd,
BOOL bBuffered,
LRESULT lResult
);
```

Parameters

hClientWnd

client window handle

bBuffered

[in] buffered mode setting;
TRUE=buffered mode,
FALSE=non-buffered (wedge) mode

lResult

variable to receive API call result

Return Values

lResult

ERROR_SUCCESS indicates success
ERROR_FAILURE indicates failure

Remarks

When Buffered Mode is enabled, a buffer is created in shared memory to house the *SuperWedge* data buffer. *SuperWedge* creates a file mapping for a file named “SWBuffer” with CreateFileMapping() and maps a view of the buffer with MapViewOfFile(). The shared memory “file” is allocated within the OS’s RAM File System. Following *SuperWedge* post processing of data, a message is posted with the global address of the *SuperWedge* data when a buffer is ready and buffered mode is enabled.

The *SuperWedge* data buffers contain the processed data, along with the symbology type, the data count and the data source. In addition to the *SuperWedge* data, the buffer structure contains a flag to show a read status of the data. The receiver must clear the flag once it has received the data marking the buffer as available for re-use. Upon data collection, *SuperWedge* stores the data in the next available buffer (if buffering is enabled) and posts the Windows message when the flag is cleared.. If there is no available buffer, *SuperWedge* tosses the data into the bit bucket and issues a buffer full error beep.

When "Buffered Mode" is disabled, *SuperWedge* operates in "Wedge Mode", i.e. output data are passed as emulated keystrokes. The use of Buffered Mode or Wedge Mode is dependent on the individual application need. Wedge Mode provides simple set-up and program interface; however, output data processing is limited. Buffered Mode provides virtually unlimited output data processing but with the added burden of a more complicated set-up and program interface.

Benefits of Wedge Mode

- Data input is programming language independent.
- Eliminates the use of application programming interface – scanned data appears to any application program as if it had been typed in.
- Pre-tested and proven interface – no coding hassles, no debugging.
- Provides out-of-the-box scan capabilities – no set-up required.
- Works with 3rd party applications that are unaware of laser scanner operation.

Benefits of Buffered Mode

- Application can tell the source of data input: scanner, Aux Port, etc.
- The number of decoded characters is known.
- Data transfer is fast.

SWInitialize

Initialize *SuperWedge*. Called to reset *SuperWedge* to the INI file settings.

```
SWInitialize(
HWND hSuperWedgeWnd,
LRESULT lResult
);
```

Parameters

hClientWnd

client window handle

lResult

variable to receive API call result

Return Values

lResult

ERROR_SUCCESS indicates success
ERROR_FAILURE indicates failure

Remarks'

This function duplicates the **Initialize** selection on the *SuperWedge* taskbar menu. It is not a required call for API operation.

SWExit

Call to direct *SuperWedge* to shutdown. This function will replicate the functionality provided by the *SuperWedge* Exit menu option.

```
SWExit(
HWND hSuperWedgeWnd,
LRESULT lResult
);
```

Parameters

hClientWnd

client window handle

lResult

variable to receive API call result

Return Values

lResult

ERROR_SUCCESS indicates success
ERROR_FAILURE indicates failure

Remarks

Failure may result from no registered client.

5.2 Command APIs

SWSetActive

Set *SuperWedge* active state. Called to activate or deactivate *SuperWedge* operation.

```
SWSetActive(
HWND hSuperWedgeWnd,
BOOL bActivate,
LRESULT lResult
);
```

Parameters

hClientWnd
client window handle

bActivate
active state setting;
TRUE=active,
FALSE=inactive

lResult
variable to receive API call result

Return Values

lResult
ERROR_SUCCESS indicates success
ERROR_FAILURE indicates failure

Remarks

This function duplicates the **Enable** selection on the *SuperWedge* taskbar menu.

SWSetLock

Set *SuperWedge* lock state. Called to lock or unlock *SuperWedge* operation.

```
SWSetLock(
HWND hSuperWedgeWnd,
BOOL bLock,
LRESULT lResult);
```

Parameters

hClientWnd
client window handle

bLock
lock state setting;
TRUE=locked,
FALSE=unlocked

lResult
variable to receive API call result

Return Values

lResult
ERROR_SUCCESS indicates success
ERROR_FAILURE indicates failure

Remarks

This function duplicates the **Lock** selection on the *SuperWedge* taskbar menu

SWSetTrigger

Set *SuperWedge* trigger state. Called to pull or release the *SuperWedge* trigger.

```
SWSetTrigger(
HWND hSuperWedgeWnd,
BOOL bOnOff,
LRESULT lResult
);
```

Parameters

hClientWnd
client window handle

bOnOff
trigger setting; TRUE=pull, FALSE=release

lResult
variable to receive API call result

Return Values

lResult
ERROR_SUCCESS indicates success
ERROR_FAILURE indicates failure

Remarks

For every software trigger pull (*bOnOff*=TRUE), there must be a corresponding software trigger release (*bOnOff*=FALSE).

5.3 Configuration APIs

SWSetParameter

Sets *SuperWedge* parameter setting(s). Called to update one or more parameter settings.

```
SWSetParameter(
HWND hSuperWedgeWnd,
LPCTSTR lpSetting,
LRESULT lResult
);
```

Parameters

hClientWnd
client window handle

lpSetting
ASCII string of parameter settings

lResult
variable to receive API call result

Return Values

lResult
ERROR_SUCCESS indicates success
Non-zero indicates error. The low word of *lResult* indicates the line number of the first line with an error. The high word of *lResult* indicates the argument that causes the error. Applications can then call GetLastError() to determine the type of error that occurred.

Remarks

lpSetting points to an ASCII string which contains a single or multiple parameter settings separated by carriage return and line feed characters. The ASCII string may contain any of the parameters described in Section 4. The ASCII string may also contain the "defaultall" command to direct *SuperWedge* to reset all settings to their factory defaults prior to any settings following in the string.

Example: CHAR ucStartingSetup[]="defaultall\r\nAim Dot=1,10\r\nAuxPort=1\r\n\0";

SWGetStatus

Retrieves the current status of all *SuperWedge* parameters. Parameters are returned via the wedge interface or previously set buffer. Syntax is identical to the initialization INI file.

```
SWGetStatus(
HWND hSuperWedgeWnd,
LRESULT lResult
);
```

Parameters

hClientWnd
client window handle

lResult
variable to receive API call result

Return Values

lResult
ERROR_SUCCESS indicates success
ERROR_FAILURE indicates failure

Remarks

Data type of SW_DATATYPE_MSG indicates the data is a status message from *SuperWedge*.

5.4 Data Buffer

When an application uses the API and puts *SuperWedge* in Buffered Mode, data are returned from *SuperWedge* to the application via a data buffer.

The data buffer has the following structure:

```
typedef struct BUFFERED_SUPERWEDGE_DATA_S
{
   WORD   wFlag;          // TRUE = unread data, FALSE = read data
   WORD   wDataType;      // Symbology/Data source
   WORD   wCount;         // Count of characters placed in data buffer
   WORD   wData[4096];    // SuperWedge data buffer
} BUFFERED_SUPERWEDGE_DATA, *PSWBUF;
```

When *SuperWedge* has data ready for the application, *SuperWedge* sends the application the WM_SUPERWEDGE_BUFFER_READY Windows Message. *SuperWedge* data will be located at the global address sent as wParam. The address is the shared memory created by *SuperWedge* during the SWSetMode call which enabled buffered mode.

In addition to *SuperWedge* data, the buffer structure contains a flag, *wFlag,* set to TRUE indicating that the data is unread. Upon reading, the client must clear the flag so that *SuperWedge* knows the buffer is free for re-use.

Reference Section 7.4 for a list of data types returned by *wDataType*.

5.5 Sample Code

The following code is an eVC++ snippet demonstrating the use of the *SuperWedge* API. For a full example, reference `SwSimpleApiTest.cpp` found on the product support CD. A C# example, `SW2testNET`, VisualBasic samples and class libraries are on the product support CD also.

```
//---------------------------------------------------------------------
// Code snippet taken from SwSimpleApiTest.cpp
//
BOOL HookupWithSuperWedge()
{
    LRESULT lResult;
    DWORD   dwSWVersion;
    CHAR    ucSetting1[] = "Code ID Type=0\r\n"\
                           "Scanner PreAmble=060,060\r\n"\
                           "Scanner PostAmble=062,062\r\n"\
                           "AuxPort=1,115200\r\n\0";
    CHAR    ucSetting2[] = "Code ID Type=1\r\n"\
                           "Scanner PostAmble=041,041\r\n"\
                           "Scanner PreAmble=040,040\r\n"\
                           "AuxPort=1\r\n\0";
    CHAR    ucStartingSetup[] = "defaultall\r\n"\
                                "Aim Dot=1,10\r\n"\
                                "AuxPort=1\r\n\0";
    SWBUF   SwMem;
    PUCHAR  pMem=(PUCHAR)&SwMem;
    int     iMsg, cnt;
    PCHAR   pSetting;

    // Register this app. with SuperWedge and get its window handle
    SWRegisterClient(hMainWnd, hSwWnd, lResult);

    if ( !hSwWnd )
    {
        MessageBox(hMainWnd,
                   L"FindWindow failed to find the SuperWedge!",
                   L"SuperWedge failed!",
                   MB_OK);
        lResult = FALSE;
    }
    else if ( lResult != ERROR_SUCCESS )
    {
        MessageBox(hMainWnd,
                   L"Failed to register with SuperWedge!",
                   L"SuperWedge failed!",
                   MB_OK);
        lResult = FALSE;
    }
    else
    {
        // verify SuperWedge version
        SWGetVersion(hSwWnd, dwSWVersion, lResult);

        if (dwSWVersion < 0x2000)
        {
            MessageBox(hMainWnd,
                       L"Not SuperWedge v2!",
                       L"Version Error",
                       MB_OK);
            lResult = FALSE;
        }
        else
```

```
        {
            // turn on buffered data mode
            SWSetBufferedMode(hSwWnd, TRUE, lResult);

            // start us off at a known state
            SWSetParameter(hSwWnd, ucStartingSetup, lResult);

            if (lResult != ERROR_SUCCESS)
            {
                TCHAR   tcMsg[30];
                swprintf(tcMsg,
                         L"SWSetParameter failed! bad param #%d\r\n",
                         lResult);
                MessageBox(hMainWnd, tcMsg, L"API Fail", MB_OK);
            }

            cnt = 0;
            do
            {
                // toggle the ambles to demonstrate "on-the-fly"
                // configuration
                cnt++;
                pSetting = cnt & 1 ? ucSetting1 : ucSetting2;

                SWSetParameter(hSwWnd, pSetting, lResult);

                if (lResult != ERROR_SUCCESS)
                {
                    TCHAR   tcMsg[30];
                    swprintf(tcMsg,
                             L"SWSetParameter failed! bad param #%d\r\n",
                             lResult);
                    MessageBox(hMainWnd, tcMsg, L"API Fail", MB_OK);
                }

                // see if they want to trigger or exit
                iMsg = MessageBox(hMainWnd,
                                  L"Click OK to Pull Trigger, "\
                                    "Cancel to end the test!",
                                  L"Trigger Test",
                                  MB_OKCANCEL);

                if (iMsg != IDCANCEL)
                {
                    SWSetTrigger(hSwWnd, TRUE, lResult);
                }
            } while ( iMsg != IDCANCEL );
        }
    }

    // detach
    SWDeRegisterClient(hMainWnd, hSwWnd, lResult);

    return lResult;
}
```

6 Troubleshooting

Use the following list to aid troubleshooting of some common *SuperWedge* problems.

Icon does not appear in task tray

Has *SuperWedge* been loaded?
Did an error occur during initialization? If so, *SuperWedge* does not remain resident.
Is the "TaskTrayIcon" setting a 0. If so, change it to 1 and reinitialize.

Scanner does not come on when scan button is pressed

Check the unit model number and verify a scan engine is installed.
Is *SuperWedge* loaded? -- can't always go by presence of taskbar icon
Is *SuperWedge* enabled?
Is scanner port enabled?
Is *SuperWedge* configured to output directly to a window? If so, window must be open.
Is main battery critically low? Laser won't come on if it is.
Are scan buttons enabled?
Could be a hardware problem.

Scanner comes on, but does not scan bar code (beam times out after 3 to 5 seconds)

Make sure bar code isn't damaged.
Make sure scanner window is clean.
Is this symbology supported by the installed scan engine?
Make sure symbology is enabled.
Make sure symbology parameters match those of the label being read. For example, the length may not match or *SuperWedge* may be looking for a check digit but the label doesn't have one.

Scanner comes on, goes off immediately when reading a bar code, but data doesn't appear

In buffered mode?
Output edits only?
Check programmed edits.
Make sure symbology parameters match those of the label being read.
Is the bar code a menu set-up label?

Data does not appear when input through the Aux Port

Is *SuperWedge* loaded?
Is *SuperWedge* enabled?
Check the "AuxPort=" parameter and make sure the com port is set correctly.
Check the "AuxPort=" parameter and verify all communication parameters match the connected device.
Is *SuperWedge* configured to output directly to a window? If so, window must be open.
Is main battery critically low?
In buffered mode?
Output edits only?
Check programmed edits.
Could be a hardware problem. Check cables.

7 Reference

7.1 Default Key Translation

Table 7-1: Default Key Translation

Hex	Dec	Code 39	Definition	Virtual Key[1,2,3]
00	000	%U	(reserved)	N/A
01	001	$A	Caps Lock	VK_CAPITAL
02	002	$B	Left Alt On	VK_LMENU (press)
03	003	$C	Left Ctrl On	VK_LCONTROL (press)
04	004	$D	Left Shift On	VK_LSHIFT (press)
05	005	$E	{Solicit Aux Port}	Note 4
06	006	$F	Left Alt Off	VK_LMENU (release)
07	007	$G	Left Ctrl Off	VK_LCONTROL (release)
08	008	$H	Left Shift Off	VK_LSHIFT (release)
09	009	$I	Tab	VK_TAB
0A	010	$J	Line Feed	VK_LCONTROL + VK_RETURN
0B	011	$K	Home	VK_HOME
0C	012	$L	End	VK_END
0D	013	$M	Enter	VK_RETURN
0E	014	$N	Page Up	VK_PRIOR
0F	015	$O	Page Down	VK_NEXT
10	016	$P	Delete	VK_DELETE
11	017	$Q	Insert	VK_INSERT
12	018	$R	Print Screen	VK_SNAPSHOT
13	019	$S	Scroll Lock	VK_SCROLL
14	020	$T	Pause key	VK_PAUSE
15	021	$U	{Pause}	Note 4
16	022	$V	Sys. Req.	VK_PRINT
17	023	$W	Break	VK_LCONTROL + VK_SCROLL
18	024	$X	Backspace	VK_BACK
19	025	$Y	{Date}	Note 5
1A	026	$Z	{Time}	Note 5
1B	027	%A	Escape	VK_ESCAPE
1C	028	%B	Up Arrow	VK_UP
1D	029	%C	Down Arrow	VK_DOWN
1E	030	%D	Right Arrow	VK_RIGHT
1F	031	%E	Left Arrow	VK_LEFT

Notes

1. Refer to <winuser.h> or other Microsoft documentation for the definitions of the virtual key codes.
2. Keystrokes are press and release unless otherwise noted.
3. Virtual keystrokes only occur when *SuperWedge* is in "wedge" mode, not "buffered" mode.
4. This entry is interpreted as a command function. No keystrokes are emulated.
5. This entry is a command function and produces multiple keystrokes.
6. Function Codes are 0x00 through 0x1F and 0x80 through 0xFF.

Hex	Dec	Code 39	Definition	Virtual Key
20	032	(SP)	Space	VK_SPACE
21	033	/A	!	VK_LSHIFT + '1'
22	034	/B	"	VK_LSHIFT + VK_APOSTROPHE
23	035	/C	#	VK_LSHIFT + '3'
24	036	/D	$	VK_LSHIFT + '4'
25	037	/E	%	VK_LSHIFT + '5'
26	038	/F	&	VK_LSHIFT + '7'
27	039	/G	'	VK_APOSTROPHE
28	040	/H	(	VK_LSHIFT + '9'
29	041	/I	)	VK_LSHIFT + '0'
2A	042	/J	*	VK_LSHIFT + '8'
2B	043	/K	+	VK_LSHIFT + VK_EQUAL
2C	044	/L	,	VK_COMMA
2D	045	-	-	VK_HYPHEN
2E	046	.	.	VK_PERIOD
2F	047	/O	/	VK_SLASH
30	048	0	0	'0'
31	049	1	1	'1'
32	050	2	2	'2'
33	051	3	3	'3'
34	052	4	4	'4'
35	053	5	5	'5'
36	054	6	6	'6'
37	055	7	7	'7'
38	056	8	8	'8'
39	057	9	9	'9'
3A	058	/Z	:	VK_LSHIFT + VK_SEMICOLON
3B	059	%F	;	VK_SEMICOLON
3C	060	%G	<	VK_COMMA
3D	061	%H	=	VK_EQUAL
3E	062	%I	>	VK_PERIOD
3F	063	%J	?	VK_SLASH
40	064	%V	@	VK_LSHIFT + '2'
41	065	A	A	VK_LSHIFT + 'A'
42	066	B	B	VK_LSHIFT + 'B'
43	067	C	C	VK_LSHIFT + 'C'
44	068	D	D	VK_LSHIFT + 'D'
45	069	E	E	VK_LSHIFT + 'E'
46	070	F	F	VK_LSHIFT + 'F'
47	071	G	G	VK_LSHIFT + 'G'
48	072	H	H	VK_LSHIFT + 'H'
49	073	I	I	VK_LSHIFT + 'I'
4A	074	J	J	VK_LSHIFT + 'J'
4B	075	K	K	VK_LSHIFT + 'K'
4C	076	L	L	VK_LSHIFT + 'L'
4D	077	M	M	VK_LSHIFT + 'M'
4E	078	N	N	VK_LSHIFT + 'N'
4F	079	O	O	VK_LSHIFT + 'O'

Hex	Dec	Code 39	Definition	Virtual Key
50	080	P	P	VK_LSHIFT + 'P'
51	081	Q	Q	VK_LSHIFT + 'Q'
52	082	R	R	VK_LSHIFT + 'R'
53	083	S	S	VK_LSHIFT + 'S'
54	084	T	T	VK_LSHIFT + 'T'
55	085	U	U	VK_LSHIFT + 'U'
56	086	V	V	VK_LSHIFT + 'V'
57	087	W	W	VK_LSHIFT + 'W'
58	088	X	X	VK_LSHIFT + 'X'
59	089	Y	Y	VK_LSHIFT + 'Y'
5A	090	Z	Z	VK_LSHIFT + 'Z'
5B	091	%K	[	VK_LBRACKET
5C	092	%L	\	VK_BACKSLASH
5D	093	%M	]	VK_RBRACKET
5E	094	%N	^	VK_LSHIFT + '6'
5F	095	%O	_	VK_SHIFT + VK_HYPHEN
60	096	%W	`	VK_BACKQUOTE
61	097	+A	a	'A'
62	098	+B	b	'B'
63	099	+C	c	'C'
64	100	+D	d	'D'
65	101	+E	e	'E'
66	102	+F	f	'F'
67	103	+G	g	'G'
68	104	+H	h	'H'
69	105	+I	i	'I'
6A	106	+J	j	'J'
6B	107	+K	k	'K'
6C	108	+L	l	'L'
6D	109	+M	m	'M'
6E	110	+N	n	'N'
6F	111	+O	o	'O'
70	112	+P	p	'P'
71	113	+Q	q	'Q'
72	114	+R	r	'R'
73	115	+S	s	'S'
74	116	+T	t	'T'
75	117	+U	u	'U'
76	118	+V	v	'V'
77	119	+W	w	'W'
78	120	+X	x	'X'
79	121	+Y	y	'Y'
7A	122	+Z	z	'Z'
7B	123	%P	{	VK_LSHIFT + VK_LBRACKET
7C	124	%Q	|	VK_LSHIFT + VK_BACKSLASH
7D	125	%R	}	VK_LSHIFT + VK_RBRACKET
7E	126	%S	~	VK_LSHIFT + VK_BACKQUOTE
7F	127	%T		

Hex	Dec	Code 39	Definition	Virtual Key
Numeric Keypad				
80	128	%U%U	.	VK_DECIMAL
81	129	%U$A	/	VK_DIVIDE
82	130	%U$B	*	VK_MULTIPLY
83	131	%U$C	-	VK_SUBTRACT
84	132	%U$D	+	VK_ADD
85	133	%U$E	Enter	VK_RETURN
86	134	%U$F	Num Lock	VK_NUMLOCK
Extended Function Keys				
87	135	%U$G	Right Alt On	VK_RMENU (press)
88	136	%U$H	Right Ctrl On	VK_RCONTROL (press)
89	137	%U$I	Right Shift On	VK_RSHIFT (press)
8A	138	%U$J	Right Alt Off	VK_RMENU (release)
8B	139	%U$K	Right Ctrl Off	VK_RCONTROL (release)
8C	140	%U$L	Right Shift Off	VK_RSHIFT (release)
8D	141	%U$M		
8E	142	%U$N		
8F	143	%U$O		
90	144	%U$P		
91	145	%U$Q		
92	146	%U$R	Stylus Tap	VK_LBUTTON
93	147	%U$S	Mouse right button	VK_RBUTTON
94	148	%U$T	Control-break processing	VK_CANCEL
95	149	%U$U	Mouse middle button	VK_MBUTTON
96	150	%U$V	Help key	VK_HELP
97	151	%U$W	Left Windows key	VK_LWIN
98	152	%U$X	Right Windows key	VK_RWIN
99	153	%U$Y	Context Menu key	VK_APPS
9A	154	%U$Z	Computer Sleep key	VK_SLEEP
9B	155	%U%A	Clear key	VK_CLEAR
9C	156	%U%B	Select key	VK_SELECT
9D	157	%U%C	Execute key	VK_EXECUTE
9E	158	%U%D		
9F	159	%U%E		
Numeric Keypad				
A0	160	%U(SP)	0	VK_NUMPAD0
A1	161	%U/A	1	VK_NUMPAD1
A2	162	%U/B	2	VK_NUMPAD2
A3	163	%U/C	3	VK_NUMPAD3
A4	164	%U/D	4	VK_NUMPAD4
A5	165	%U/E	5	VK_NUMPAD5
A6	166	%U/F	6	VK_NUMPAD6
A7	167	%U/G	7	VK_NUMPAD7
A8	168	%U/H	8	VK_NUMPAD8
A9	169	%U/I	9	VK_NUMPAD9
Extended Function Keys				
AA	170	%U/J		
AB	171	%U/K		
AC	172	%U/L		
AD	173	%U/M		
AE	174	%U/N		
AF	175	%U/O		

Hex	Dec	Code 39	Definition	Virtual Key
Extended Function Keys				
B0	176	%U/P		
B1	177	%U/Q		
B2	178	%U/R		
B3	179	%U/S		
B4	180	%U/T		
B5	181	%U/U		
B6	182	%U/V		
B7	183	%U/W		
B8	184	%U/X		
B9	185	%U/Y		
BA	186	%U/Z		
BB	187	%U%F		
BC	188	%U%G		
BD	189	%U%H		
BE	190	%U%I		
BF	191	%U%J		
C0	192	%U%V	F1 key	VK_F1
C1	193	%UA	F2 key	VK_F2
C2	194	%UB	F3 key	VK_F3
C3	195	%UC	F4 key	VK_F4
C4	196	%UD	F5 key	VK_F5
C5	197	%UE	F6 key	VK_F6
C6	198	%UF	F7 key	VK_F7
C7	199	%UG	F8 key	VK_F8
C8	200	%UH	F9 key	VK_F9
C9	201	%UI	F10 key	VK_F10
CA	202	%UJ	F11 key	VK_F11
CB	203	%UK	F12 key	VK_F12
CC	204	%UL	F13 key	VK_F13
CD	205	%UM	F14 key	VK_F14
CE	206	%UN	F15 key	VK_F15
CF	207	%UO	F16 key	VK_F16
D0	208	%UP	F17 key	VK_F17
D1	209	%UQ	F18 key	VK_F18
D2	210	%UR	F19 key	VK_F19
D3	211	%US	F20 key	VK_F20
D4	212	%UT	F21 key	VK_F21
D5	213	%UU	F22 key	VK_F22
D6	214	%UV	F23 key	VK_F23
D7	215	%UW	F24 key	VK_F24
D8	216	%UX		VK_PROCESSKEY
D9	217	%UY	Attn key	VK_ATTN
DA	218	%UZ	CrSel key	VK_CRSEL
DB	219	%U%K	ExSel key	VK_EXSEL
DC	220	%U%L	Erase EOF key	VK_EREOF
DD	221	%U%M	Play key	VK_PLAY
DE	222	%U%N	Zoom key	VK_ZOOM
DF	223	%U%O		

Hex	Dec	Code 39	Definition	Virtual Key
Extended Function Keys				
E0	224	%U%W	PA1 key	VK_PA1
E1	225	%U+A	Clear key	VK_OEM_CLEAR
E2	226	%U+B		VK_BROWSER_BACK
E3	227	%U+C		VK_BROWSER_FORWARD
E4	228	%U+D		VK_BROWSER_REFRESH
E5	229	%U+E		VK_BROWSER_STOP
E6	230	%U+F		VK_BROWSER_SEARCH
E7	231	%U+G		VK_BROWSER_FAVORITES
E8	232	%U+H		VK_BROWSER_HOME
E9	233	%U+I		VK_VOLUME_MUTE
EA	234	%U+J		VK_VOLUME_DOWN
EB	235	%U+K		VK_VOLUME_UP
EC	236	%U+L		VK_MEDIA_NEXT_TRACK
ED	237	%U+M		VK_MEDIA_PREV_TRACK
EE	238	%U+N		VK_MEDIA_STOP
EF	239	%U+O		VK_MEDIA_PLAY_PAUSE
F0	240	%U+P		VK_LAUNCH_MAIL
F1	241	%U+Q		VK_LAUNCH_MEDIA_SELECT
F2	242	%U+R	Start Application 1 key	VK_LAUNCH_APP1
F3	243	%U+S	Start Application 2 key	VK_LAUNCH_APP2
F4	244	%U+T		
F5	245	%U+U		
F6	246	%U+V		
F7	247	%U+W		
F8	248	%U+X		
F9	249	%U+Y		
FA	250	%U+Z		
FB	251	%U%P		
FC	252	%U%Q		
FD	253	%U%R		
FE	254	%U%S		
FF	255		(reserved)	

7.2 Code 39 Full ASCII Conversion

If Code 39 Full ASCII conversion is enabled, certain two character sequences made up of one of the Code 39 symbols ($, +, %, /) followed by one of the 26 letters are interpreted as a single character.

Table 7-2: Code 39 Full ASCII Conversion

NUL	%U	DLE	$P	SP	SP	0	0	@	%V	P	P	`	%W	p	+P
SOH	$A	DC1	$Q	!	/A	1	1	A	A	Q	Q	a	+A	q	+Q
STX	$B	DC2	$R	"	/B	2	2	B	B	R	R	b	+B	r	+R
ETX	$C	DC3	$S	#	/C	3	3	C	C	S	S	c	+C	s	+S
EOT	$D	DC4	$T	$	/D	4	4	D	D	T	T	d	+D	t	+T
ENQ	$E	NAK	$U	%	/E	5	5	E	E	U	U	e	+E	u	+U
ACK	$F	SYN	$V	&	/F	6	6	F	F	V	V	f	+F	v	+V
BEL	$G	ETB	$W	'	/G	7	7	G	G	W	W	g	+G	w	+W
BS	$H	CAN	$X	(	/H	8	8	H	H	X	X	h	+H	x	+X
HT	$I	EM	$Y	)	/I	9	9	I	I	Y	Y	i	+I	y	+Y
LF	$J	SUB	$Z	*	/J	:	/Z	J	J	Z	Z	j	+J	z	+Z
VT	$K	ESC	%A	+	/K	;	%F	K	K	[	%K	k	+K	{	%P
FF	$L	FS	%B	,	/L	<	%G	L	L	\	%L	l	+L	|	%Q
CR	$M	GS	%C	-	-	=	%H	M	M	]	%M	m	+M	}	%R
SO	$N	RS	%D	.	.	>	%I	N	N	^	%N	n	+N	~	%S
SI	$O	US	%E	/	/O	?	%J	O	O	_	%O	o	+O	DEL	%T

Notes

1. Character sequences /M and /N are interpreted as "-" and "." respectively.
2. Character sequences /P through /Y are interpreted as "0" through "9" respectively.
3. Character sequences %X through %Z are undefined per the Code 39 specification but are interpreted as DEL.

7.3 Keyboard Function Records

Keyboard Function Records are used to manipulate the cursor on the display and to cause the terminal to enter modes of operation that ordinarily require a keystroke.

Keyboard Function Records are defined as three-byte ASCII character records that when scanned will emulate the keyboard function to the terminal. To use Keyboard Function Records, Code 39 should be set as follows:

- Code 39 must be enabled
- Full ASCII must be enabled
- Length settings must allow for 3 characters

For example, "Code39=1,1,1,55,1,0,0,0,1,0,0,0"

#01 - Caps Lock

#02 - Left Alt On

#03 - Left Ctrl On

#04 - Left Shift On

#05 - {Solicit Aux Port}

#06 - Left Alt Off

#07 - Left Ctrl Off

#08 - Left Shift Off

#09 - Tab

#10 - Line Feed

#11 - Home

#12 - End

#14 - Page Up

#16 - Delete

#18 - Print Screen

#20 - Pause key

#22 - SysReq

#24 - Backspace

#26 - {Time}

#28 - Up Arrow

#30 - Right Arrow

#13 - Enter

#15 - Page Down

#17 - Insert

#19 - Scroll Lock

#21 - {Pause}

#23 - Break

#25 - {Date}

#27 - Esc

#29 - Down Arrow

#31 - Left Arrow

7.4 Data Types

This table is provided for reference only. Refer to `swapi.h` for the official list.

Table 7-3: Data Types

Code Type	Symbolic Constant	Hex	Dec
No Code	SW_DATATYPE_NONE	0x00	0
Code 39	SW_DATATYPE_CODE39	0x01	1
Codabar	SW_DATATYPE_CODABAR	0x02	2
Code 128	SW_DATATYPE_CODE128	0x03	3
Discrete 2 of 5 (Industrial 2 of 5)	SW_DATATYPE_D2OF5	0x04	4
IATA 2 of 5 (Identicode 2 of 5)	SW_DATATYPE_IATA25	0x05	5
Interleaved 2 of 5	SW_DATATYPE_I2OF5	0x06	6
Code 93	SW_DATATYPE_CODE93	0x07	7
UPC-A	SW_DATATYPE_UPCA	0x08	8
UPC-E	SW_DATATYPE_UPCE	0x09	9
EAN-8	SW_DATATYPE_EAN8	0x0A	10
EAN-13	SW_DATATYPE_EAN13	0x0B	11
Code 11	SW_DATATYPE_CODE11	0x0C	12
MSI Plessey	SW_DATATYPE_MSI_PLESSEY	0x0E	14
EAN-128	SW_DATATYPE_EAN128	0x0F	15
UPC-E1	SW_DATATYPE_UPCE1	0x10	16
PDF-417	SW_DATATYPE_PDF417	0x11	17
Code 39 Full ASCII	SW_DATATYPE_CODE39_FULL_ASCII	0x13	19
Trioptic Code 39	SW_DATATYPE_TRIOPTIC	0x15	21
Bookland EAN	SW_DATATYPE_BOOKLAND	0x16	22
Coupon Code	SW_DATATYPE_COUPONCODE	0x17	23
ISBT-128	SW_DATATYPE_ISBT128	0x19	25
Micro PDF-417	SW_DATATYPE_MICRO_PDF417	0x1A	26
Data Matrix	SW_DATATYPE_DATA_MATRIX	0x1B	27
Code 32	SW_DATATYPE_CODE32	0x20	32
ISBT-128 Concatenate	SW_DATATYPE_ISBT128_CONCAT	0x21	33
Maxicode	SW_DATATYPE_MAXICODE	0x25	35
RSS-14	SW_DATATYPE_RSS_14	0x30	48
RSS Limited	SW_DATATYPE_RSS_LIMITED	0x31	49
RSS Expanded	SW_DATATYPE_RSS_EXPANDED	0x32	50
UPC-A with 2 supps.	SW_DATATYPE_UPCA_2	0x48	72
UPC-E with 2 supps.	SW_DATATYPE_UPCE_2	0x49	73
EAN-8 with 2 supps.	SW_DATATYPE_EAN8_2	0x4A	74
EAN-13 with 2 supps.	SW_DATATYPE_EAN13_2	0x4B	75
UPC-E1 with 2 supps.	SW_DATATYPE_UPCE1_2	0x50	80
	SW_DATATYPE_CCA_EAN128	0x51	81
	SW_DATATYPE_CCA_EAN13	0x52	82
	SW_DATATYPE_CCA_EAN8	0x53	83
	SW_DATATYPE_CCA_RSS_EXPANDED	0x54	84
	SW_DATATYPE_CCA_RSS_LIMITED	0x55	85
	SW_DATATYPE_CCA_RSS_14	0x56	86
	SW_DATATYPE_CCA_UPCA	0x57	87
	SW_DATATYPE_CCA_UPCE	0x58	88

Code Type	Symbolic Constant	Hex	Dec
	SW_DATATYPE_CCC_EAN128	0x59	89
	SW_DATATYPE_TLC39	0x5A	90
	SW_DATATYPE_CCB_EAN128	0x61	97
	SW_DATATYPE_CCB_EAN13	0x62	98
	SW_DATATYPE_CCB_EAN8	0x63	99
	SW_DATATYPE_CCB_RSS_EXPANDED	0x64	100
	SW_DATATYPE_CCB_RSS_LIMITED	0x65	101
	SW_DATATYPE_CCB_RSS_14	0x66	102
	SW_DATATYPE_CCB_UPCA	0x67	103
	SW_DATATYPE_CCB_UPCE	0x68	104
UPC-A with 5 supps.	SW_DATATYPE_UPCA_5	0x88	136
UPC-E with 5 supps.	SW_DATATYPE_UPCE_5	0x89	137
EAN-8 with 5 supps.	SW_DATATYPE_EAN8_5	0x8A	138
EAN-13 with 5 supps.	SW_DATATYPE_EAN13_5	0x8B	139
UPC-E1 with 5 supps.	SW_DATATYPE_UPCE1_5	0x90	144
Vehicle ID Number	SW_DATATYPE_VIN	0xC0	192
Aztec Code	SW_DATATYPE_AZTEC	0xC1	193
Code 16K	SW_DATATYPE_CODE16K	0xC2	194
Code 49	SW_DATATYPE_CODE49	0xC3	195
Auxiliary Port	SW_DATATYPE_AUXPORT	0x100	256
Used in Edit declaration to indicate the edit applies to data from any input source	SW_DATATYPE_ANY	0xA000	40960
Used in Edit declaration to indicate the edit applies to data of any decoded symbology	SW_DATATYPE_SCANNER	0xA001	40961
Indicates data is a message from *SuperWedge* rather than data from an input source	SW_DATATYPE_MSG	0xF000	61440

7.5 Code Identification

Table 7-4: Code Identification

Code Type	AIM[1]	Compsee	Symbol	HHP
No Code		Z		
Aztec	z	N		z
Bookland	X	M	L	d
Codabar	F	C	C	a
Code 11	H	F	H	h
Code 16K	K	N		o
Code 32	A	A	B	<
Code 39	A	A	B	b
Code 39 Full ASCII	A	A	B	b
Code 49	T	N		l
Code 93	G	E	E	i
Code 128	C	D	D	j
CCA+EAN128	e	T	T	y
CCA+EAN13	e	T	T	y
CCA+EAN8	e	T	T	y
CCA+RSS Expanded	e	T	T	y
CCA+RSS Limited	e	T	T	y
CCA+RSS14	e	T	T	y
CCA+UPCA	e	T	T	y
CCA+UPCE	e	T	T	y
CCC+EAN128	e	T	T	y
CCB+EAN128	e	T	T	y
CCB+EAN13	e	T	T	y
CCB+EAN8	e	T	T	y
CCB+RSS Expanded	e	T	T	y
CCB+RSS Limited	e	T	T	y
CCB+RSS14	e	T	T	y
CCB+UPCA	e	T	T	y
CCB+UPCE	e	T	T	y
Coupon Code	X	S	N	c
Data Matrix	d	N	P	w

Code Type	AIM[1]	Compsee	Symbol	HHP
Discrete 2 of 5	S	L	G	f
EAN8	E	I	A	D
EAN8+2	E	I	A	D
EAN8+5	E	I	A	D
EAN13	E	J	A	d
EAN13+2	E	J	A	d
EAN13+5	E	J	A	d
UCC/EAN128	C	P	K	I
IATA25 (Identicode 2 of 5)	R	L	G	f
Interleaved 2 of 5	I	B	F	e
ISBT128	C	D	D	
ISBT128 concatenate	C	D	D	
MaxiCode	U	N	P	x
MicroPDF-417	L	N	X	R
MSI Plessey	M	K	J	g
PDF417	L	N	X	r
RSS-14	e	R	R	y
RSS Limited	e	R	R	y
RSS Expanded	e	R	R	y
TLC39	L	N	X	T
Trioptic Code 39	X	O	M	=
UPCA	E	G	A	c
UPCA+2	E	G	A	c
UPCA+5	E	G	A	c
UPCE	E	H	A	E
UPCE+2	E	H	A	E
UPCE+5	E	H	A	E
UPCE1	E	H	A	E
UPCE1+2	E	H	A	E
UPCE1+5	E	H	A	E
VIN	A	V	B	b

Notes

1. The complete AIM code identification consists of a flag character ("]"), the code character listed in this table, and a modifier character. Description of the modifier character is outside the scope of this document. Refer to ITS 98-002 from AIM International, Inc. for a complete description.

7.6 Supported Symbologies

The following table lists the symbologies supported by the various scan engines.

Table 7-5: Supported Symbologies

Code Type	SE824	SE923	SE1223	SE2223	SE1524ER	SE955	SE1224	IT5180
Aztec	×	×	×	×	×	×	×	✓
Bookland	✓	✓	✓	✓	✓	✓	✓	✓
Codabar	✓	✓	✓	✓	✓	✓	✓	✓
Code 11	✓	×	×	×	✓	✓	✓	✓
Code 16K	×	×	×	×	×	×	×	✓
Code 32	✓	✓	✓	✓	✓	✓	✓	✓
Code 39	✓	✓	✓	✓	✓	✓	✓	✓
Code 39 Full ASCII	✓	✓	✓	✓	✓	✓	✓	✓
Code 49	×	×	×	×	×	×	×	✓
Code 93	✓	✓	✓	✓	✓	✓	✓	✓
Code 128	✓	✓	✓	✓	✓	✓	✓	✓
Composite Code A	×	×	×	✓	×	×	×	✓
Composite Code B	×	×	×	✓	×	×	×	✓
Composite Code C	×	×	×	✓	×	×	×	✓
Coupon Code	✓	✓	✓	✓	✓	✓	✓	✓
Data Matrix	×	×	×	×	×	×	×	✓
Discrete 2 of 5	✓	✓	✓	✓	✓	✓	✓	✓
EAN8	✓	✓	✓	✓	✓	✓	✓	✓
EAN8+2	✓	✓	✓	✓	✓	✓	✓	✓
EAN8+5	✓	✓	✓	✓	✓	✓	✓	✓
EAN13	✓	✓	✓	✓	✓	✓	✓	✓
EAN13+2	✓	✓	✓	✓	✓	✓	✓	✓
EAN13+5	✓	✓	✓	✓	✓	✓	✓	✓
UCC/EAN128	✓	✓	✓	✓	✓	✓	✓	✓
IATA25 (Identicode 2 of 5)	✓	✓	✓	✓	✓	✓	✓	✓
Interleaved 2 of 5	✓	✓	✓	✓	✓	✓	✓	✓
ISBT128	✓	✓	✓	✓	✓	✓	✓	✓
ISBT128 Concatenate	×	×	×	×	×	×	×	✓

Code Type	SE824	SE923	SE1223	SE2223	SE1524ER	SE955	SE1224	IT5180
MaxiCode	×	×	×	×	×	×	×	✓
MicroPDF-417	×	×	×	✓	×	×	×	✓
MSI Plessey	✓	✓	✓	✓	✓	✓	✓	✓
PDF417	×	×	×	✓	×	×	×	✓
RSS-14	✓	×	×	✓	×	✓	✓	✓
RSS Limited	✓	×	×	✓	×	✓	✓	✓
RSS Expanded	✓	×	×	✓	×	✓	✓	✓
TLC39	×	×	×	✓	×	×	×	✓
Trioptic Code 39	✓	✓	✓	✓	✓	✓	✓	✓
UPCA	✓	✓	✓	✓	✓	✓	✓	✓
UPCA+2	✓	✓	✓	✓	✓	✓	✓	✓
UPCA+5	✓	✓	✓	✓	✓	✓	✓	✓
UPCE	✓	✓	✓	✓	✓	✓	✓	✓
UPCE+2	✓	✓	✓	✓	✓	✓	✓	✓
UPCE+5	✓	✓	✓	✓	✓	✓	✓	✓
UPCE1	✓	✓	✓	✓	✓	✓	✓	✓
UPCE1+2	✓	✓	✓	✓	✓	✓	✓	✓
UPCE1+5	✓	✓	✓	✓	✓	✓	✓	✓
VIN	✓	✓	✓	✓	✓	✓	✓	✓

7.7 Menu Labels

When the *SuperWedge* parameter "Menu Label Setup" is enabled, the following Code 128 labels will be interpreted as commands.

Description	Data
Set to Factory Defaults	C$$$-6D
Toggle Lock Mode	C$$$-9D
Enter Status Check Menu	C---$-C
Alphas "A" through "J"	A*x*C where *x* is 0 through 9
Digits "0" through "9"	A*x*B where *x* is 0 through 9
"ON" label	A3D
"OFF" label	A2D
"Exit" label	D0D

Scan this bar code to restore the factory default values listed in Table 4-1 through Table 4-10.

Set to Factory Defaults

Scan this bar code to lock *SuperWedge* if it is unlocked. If this label is scanned while *SuperWedge* is locked, you will be prompted with the password entry screen. If the correct password is entered, *SuperWedge* will unlock.

Toggle Lock Mode

Scan this series of labels to display the current status of the parameter selections that have been programmed.

❶ Enter Status Check

Enter Status Check

❷ Send status check to the terminal

To Terminal

❸ Select which status items to output

A - General/Auxiliary/Scanner
B - Bar Code Configuration
C - Edit Parameters
D - Keyboard Remap
E - Custom Code ID
F - Configuration Properties

alphas

A

B

C

D

E

F

G

H

I

J

digits

0

1

2

3

4

5

6

7

8

9

others

ON

OFF

EXIT

7.8 Sample Labels

Code 39 (Standard)

Code 39 (Full ASCII)

Code 39 (with check digit)

Code 32

Codabar

Code 128

Code 11 (with 1 check digit)

MSI Plessey (with 1 check digit)

Code 11 (with 2 check digits)

MSI Plessey (with two MOD10 check digits)

Code 93

MSI Plessey (with MOD11 2nd check digit)

Interleaved 2 of 5 (without check digit)

Interleaved 2 of 5
(with USS/MOD10 check digit)

Interleaved 2 of 5 (OPCC version)

Interleaved 2 of 5 (EAN-13 format)

Identicode 2 of 5 (Identicon 2 of 5) (without check digit)

UPC-A (no supplemental)

UPC-A (2 character supplemental)

UPC-A (5 character supplemental)

UPC-E (no supplemental)

UPC-E (2 character supplemental)

UPC-E (5 character supplemental)

EAN/JAN-13 (no supplemental)

Industrial 2 of 5 (Standard 2 of 5) (without check digit)

UCC Coupon Code 1

UCC Coupon Code 2

UCC Coupon Code 3

UPC-E1 (no supplemental)

UPC-E1 (2 character supplemental)

UPC-E1 (5 character supplemental)

EAN/JAN-8 (no supplemental)

EAN/JAN-13 (2 character supplemental)

EAN/JAN-8 (2 character supplemental)

EAN/JAN-13 (5 character supplemental)

EAN/JAN-8 (5 character supplemental)

ISBN (Bookland EAN)

Compsee, Inc.

www.compsee.com

Email: info@compsee.com

Corporate Headquarters
400 N. Main St.
Mt. Gilead, NC 27306

(800) 828-3888 or (321) 724-4321