TECHNOLOGY SOLUTIONS UK LTD
part of HID

ASCII PROTOCOL READ WRITE SAMPLE APPLICATION (.NET DESKTOP)

www.tsl.com
Design • Development • Manufacture

CONTENT

Overview

This document describes the ASCII Read Write Sample application provided as part of the ASCII 2 Desktop SDK

History

Version	Date	Modifications
1.0	05/11/2013	Document creation

INTRODUCTION

The ASCII Protocol Read Write Sample application was developed to provide developers with examples of using the .Net API to command devices that support the Technology Solutions ASCII 2 protocol. This sample builds on the Inventory and Switch sample applications and demonstrates how to read and write transponders and other functions.

This code sample includes the functionality of the other samples and adds commands to read and write transponders.

USING THE APPLICATION

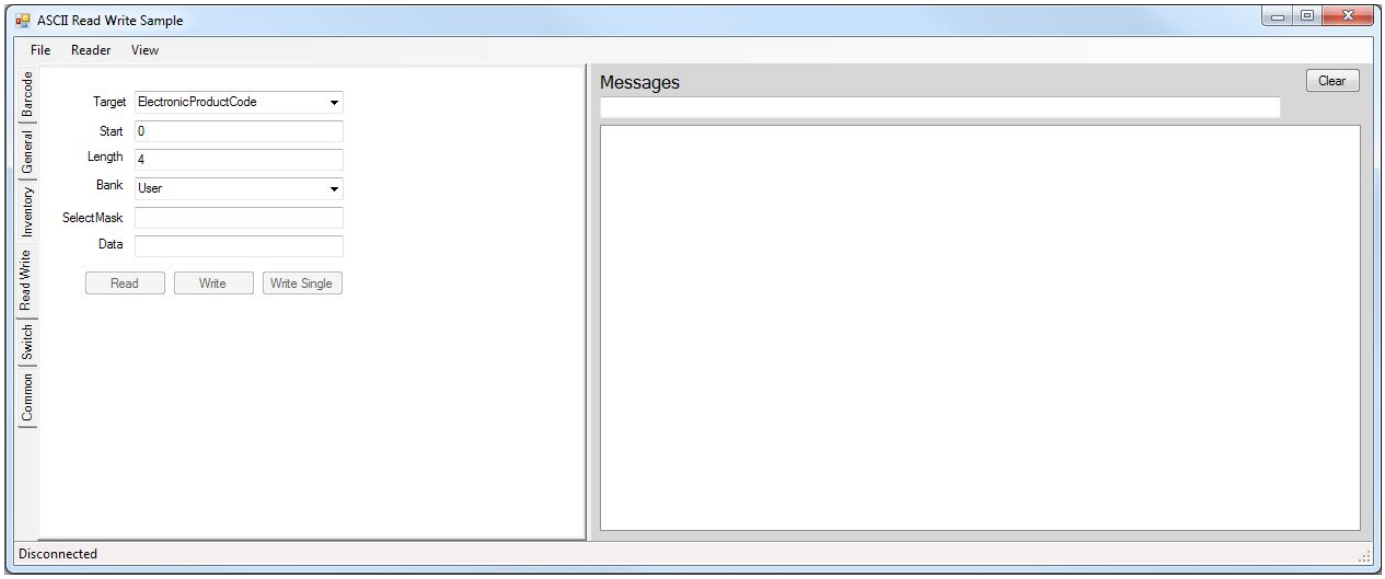

FIGURE 1: The main screen

The main screen (in its default state) is split into two main areas. The left side of the screen is split into tabs that group functionality. The right has a messages area to report command activity and responses.

You can exit the application by closing the window or with File>Exit.

CONNECTING TO THE READER

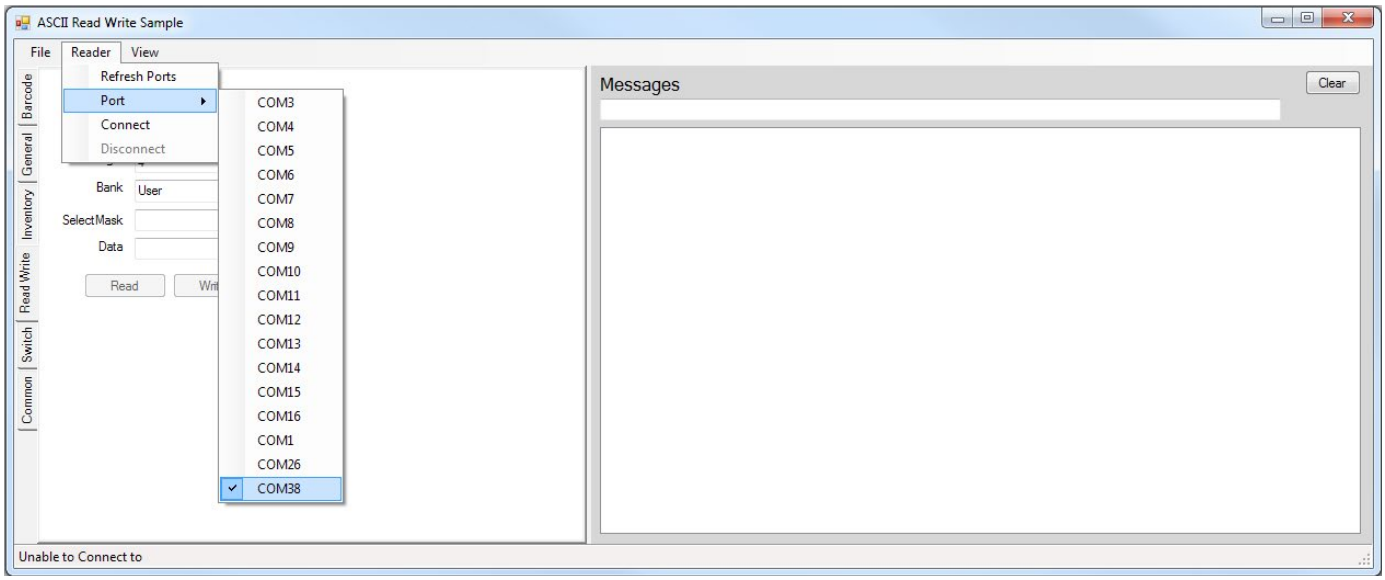

FIGURE 2: Connecting a reader

Readers are connected to the application using a standard serial port. This permits both the USB Desktop (e.g. 1126) and *Bluetooth*® (e.g. 1128) UHF Readers to be used. A list of available com ports are provided in Reader>Port menu. You will need to determine the com port that the reader is connected to. If the port required is not in the port list use Reader>Refresh Ports to refresh the list of available ports.

For *Bluetooth*® readers Windows will associate an incoming and outgoing com port when you pair a Technology Solutions UHF Reader. To establish a *Bluetooth*® connection to the reader you need to select and connect to the outgoing com port. For more information refer to the information in the reader user guide.

For USB readers Windows will associate a USB serial com port to the Technology Solutions UHF Reader as the reader is connected. Refer to the reader user guide for more information.

Once the com port has been selected use the Reader>Connect menu to connect to the com port and the reader. The connection status is shown in the status bar. Reader>Disconnect will disconnect from the reader. Once a reader is connected the controls enable

BARCODE TAB

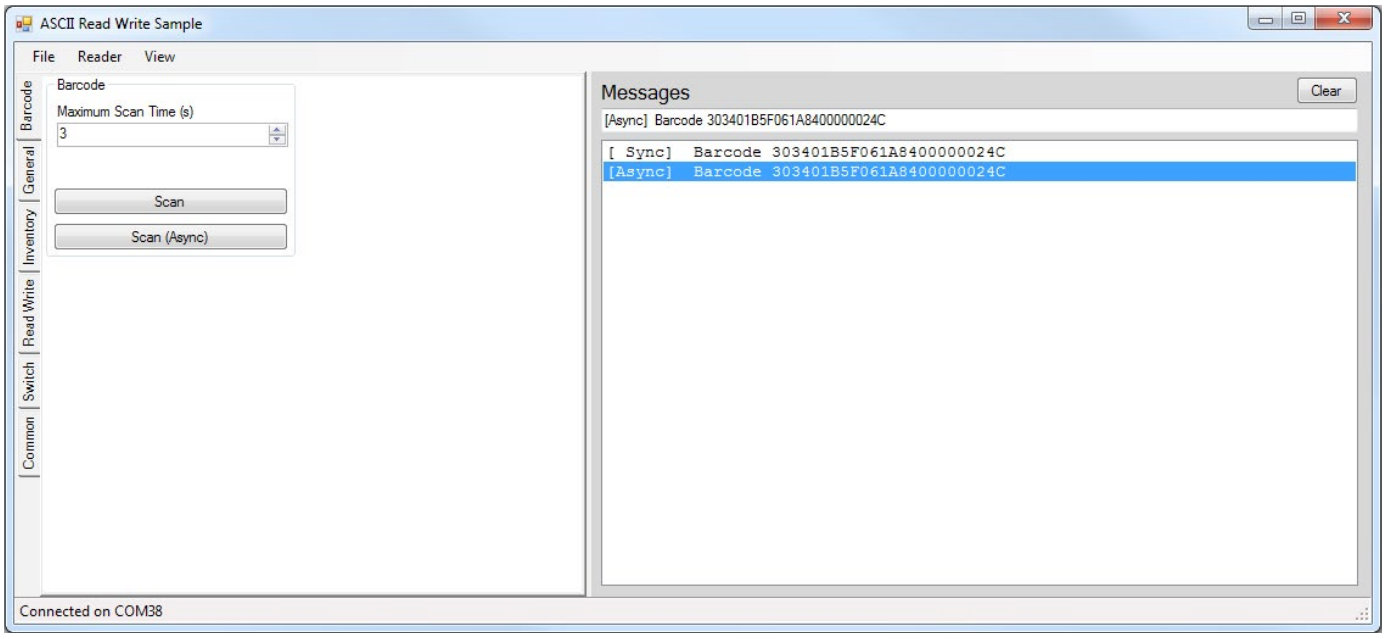

FIGURE 3: Barcode Tab

The barcode tab has controls to command a barcode. The barcode command is executed either synchronously or asynchronously depending on the button used. The maximum time in seconds that the scanner will wait for a barcode to be scanned is controlled with the up down control. Barcode output is reported in the Messages window. This feature is the sample as the Inventory sample.

GENERAL TAB

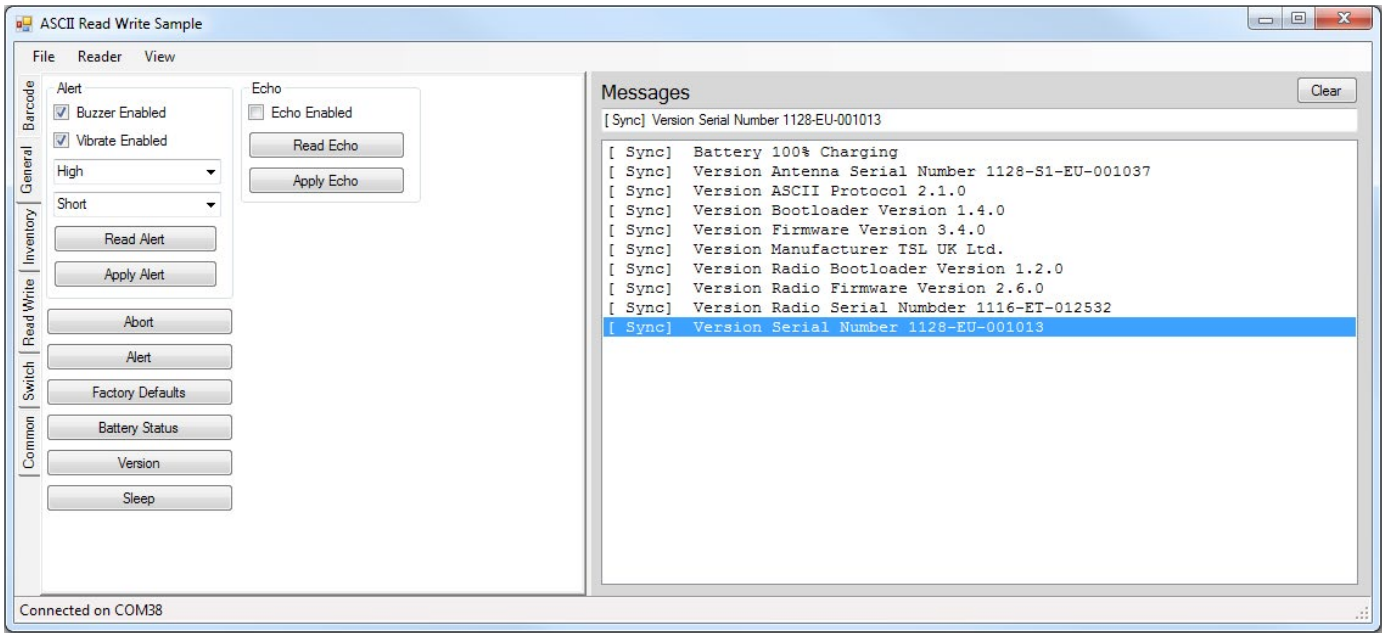

FIGURE 4: General Tab

The general tab demonstrates a number of different commands.

- The ‘Abort’ command is sent asynchronously to abort any currently executing long command (for example a switch press or barcode command).
- The ‘Alert’ box configures the alert command. Read will read the current settings and Apply will update the alert command. The ‘Alert’ button performs an alert as currently configured.
- ‘Battery Status’ will read back the battery percentage charge and charge status.
- The ‘Echo’ group reads or applies whether command echo is turned on. When enabled the command sent to the reader is echoed to the host before sending the response.
- ‘Factory Defaults’ performs the factory defaults command to revert the reader to its defaults.
- ‘Sleep’ commands the reader to sleep (this is mainly provided to allow an iOS application to forcibly disconnect from a reader).
- ‘Version’ gets the version response of the reader containing many values.

INVENTORY TAB

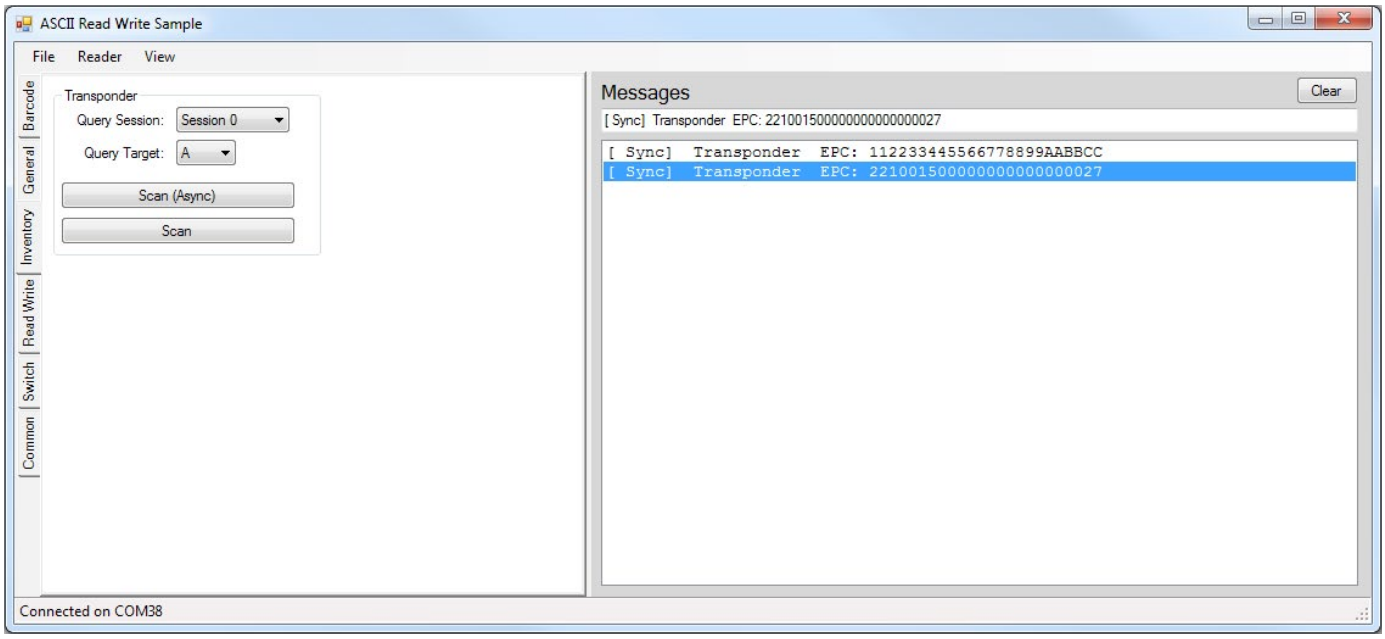

FIGURE 5: Inventory Tab

The Inventory tab contains the rest of the functionality of the original Inventory sample (the rest being on the barcode tab). The inventory can be performed synchronously or asynchronously and the basic parameters of the inventory can be set as required.

The inventory command is customised by the parameters on the Common tab which permit control of which fields are output for each transponder.

READ WRITE TAB

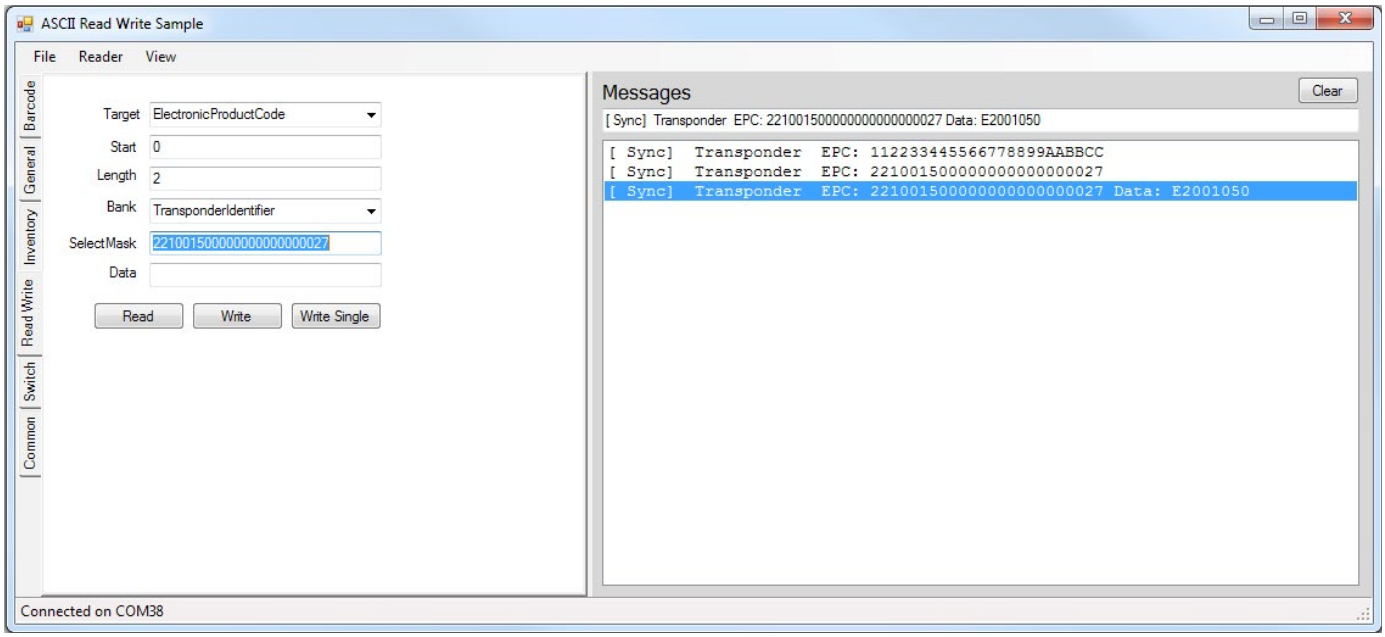

FIGURE 6: Read Write Tab

The read write tab contains functionality read or write a transponder or transponders. Currently the demo is setup to target transponders in one of three ways:

1. Target None – Read every transponder possible in range (in session 1 A)
2. Target Electronic Product Code – Assume the value in the select mask is an EPC. Perform a select and read or write transponders that have been selected by that EPC.
3. Target TID – Assume the value in the select mask is a TID. Perform a select and read or write transponders that have been selected by that TID

There is a text box above the messages list that displays the value of the selected list item. This provides mechanism to select a line in the messages list and copy values (e.g. EPC or TID) from the text box into the select mask field.

The ‘Read’ and ‘Write’ commands perform the general read and write commands as provided by the API. ‘Write Single’ is an implementation of the WriteSingle command that removes some of the options provided by the write to provide a targeted write of a single transponder with retries.
The parameters on the common tab are applied to the read and write commands to control the output power and values output for each transponder

SWITCH TAB

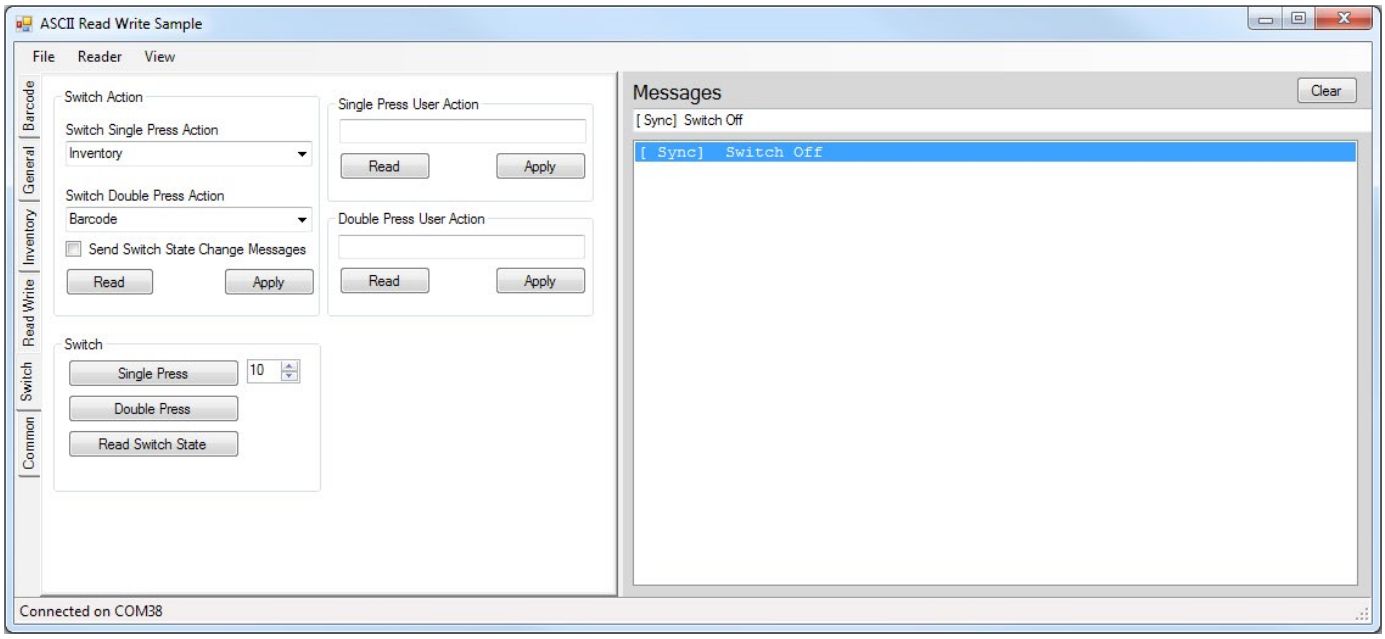

FIGURE 7: Switch Tab

The switch holds the controls added by the Switch sample code. The switch actions can be customised and configured or even turned off and the asynchronous notification of the change in switch state can be enabled.

There are also functions to perform a ‘soft’ switch, that is to command the reader to behave as if the single or double action has been triggered for a period in seconds. This can be aborted with the ‘Abort’ button on the commands tab.

COMMON TAB

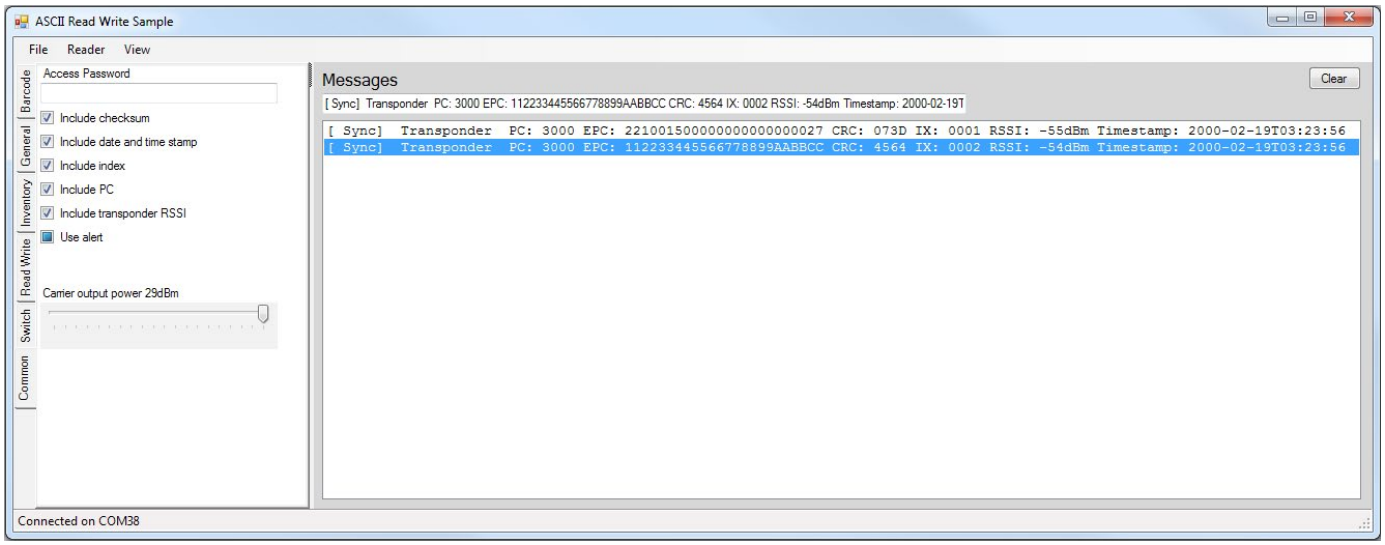

FIGURE 8: Common Tab

The common tab has been provided to provide a central location for a number of parameters that are common to many commands.

The majority of parameters for commands are optional. Where the parameter is not specified the reader uses the last specified value or the default if the value has never been specified. The checkboxes on this view are tri-state, they can be set to yes, no or not-specified. When set to not-specified the reader uses the last specified or the default value. When set they command the reader to that value and also specify the value for future commands. Note the reader maintains a separate cache for each value per command.

The Access Password is used when performing tag access commands like read and write. Leave the access password blank for it not to be specified. When specifying an access password this should be an eight character (four byte) hex value.

The ‘date and time stamp’ and ‘use alert’ properties are implemented by a number of commands. Where supported by a command they will inherit the values specified here when executed (e.g. inventory, read, write).

The checksum, index, PC and transponder RSSI are optional fields that are output when a transponder is reported by the API. Commands that return one or more transponders inherit these values.

VIEWING THE ASCII PROTOCOL RESPONSES

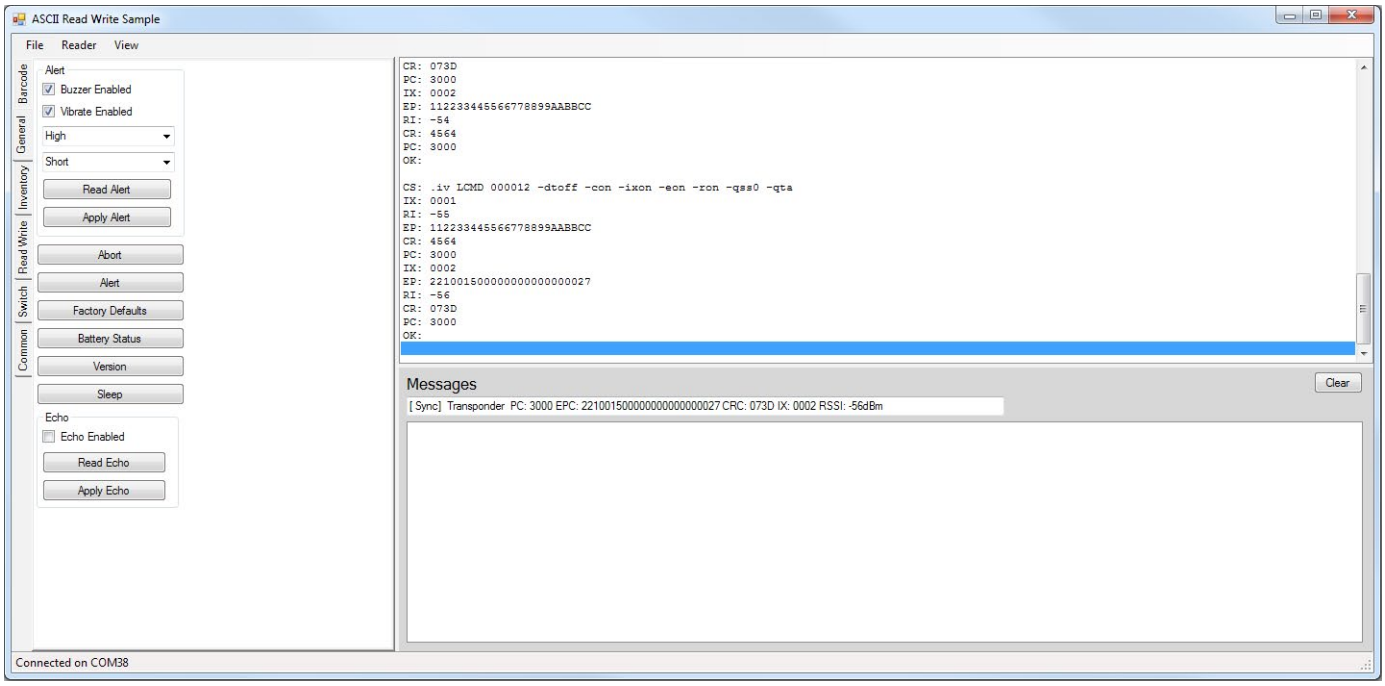

FIGURE 9: The Protocol Response Display (top)

All reader commands in the .Net API provide a convenient method of issuing ASCII commands to the device. To see the raw ASCII command responses for the operations in the sample application use the Protocol Response Display. This can be toggled using the View>Show Protocol Responses menu option.

The Technology Solutions ASCII API has the options to identify a command as a library command and also to index each command sent. This is not required by the reader which has loose parsing requirements but as the reader can echo the command in the response the library uses this feature to determine responses to library commands and specifically indexed commands from responses generated from trigger presses and other means. This can be seen using the Protocol Response Display and comparing the lines prefixed by CS: for commands issued using the on-screen buttons and those arising from trigger operations.

CODE DESCRIPTION

This code sample builds on the simpler Inventory and switch code samples. The functionality of the application is split into view models. The view models are mostly data bound using *System.ComponentModel.INotifyPropertyChanged* to reflect changes in values between the view model and the user interface (View / Form). Actions are implemented using *ICommand*. This simple interface allows an action to be defined and a method to determine whether it can execute. *Controls* on the views can then be bound to the *ICommands* presented by the view model using the appropriate *CommandBinder* instance to bind a *Control* to an *ICommand*. The *ReaderCommand* class handles reflecting the *ICommand.CanExecute* to the *Enabled* property of the bound control based on the state of the reader (connected, idle). To determine how a function of a button or menu item is implemented you can determine the *ICommand* the control is bound to in the View (Form) and find the execute delegate for that *ICommand* in the view model.

The *ReaderService* handles setting up the connection and communication with the reader via a com port. This class is inherited by *CommandService* which performs the setup of the responder chain to report reader responses back to the user interface. All messages to the user interface are now routed through the *MessageService* providing a common channel for messages from the responder chain and output after completing a synchronous command. The *DisplayResponder* is a custom responder to capture all responses from the reader to show on the UI routed through the *MessagesService*. The *SwitchAsynchronousResponder* is a custom responder that is added to the responder chain to capture the asynchronous changes in switch state when enabled.

The tabbed view hosts a user control per tab that encapsulates the user interface for that tab. The view in turn communicates with an appropriately named view model. The view model provides the properties and *ICommands* that are bound to the user interface in the view.

FURTHER INFORMATION

More information can be found on the Technology Solutions website. The product downloads section of each product requires a free, one time, registration. See “Product Downloads” of the following products to download the document describing the ASCII Protocol and also user manuals for the products.

http://www.tsl.com/products/1128-bluetooth-handheld-uhf-rfid-reader/

http://www.tsl.com/products/1126-desktop-uhf-rfid-reader-with-usb/

If you have any questions please contact support@tsl.com

ABOUT

ABOUT TSL®

Technology Solutions UK Ltd (TSL®), part of HID Global, is a leading manufacturer of high performance mobile RFID readers used to identify and track products, assets, data or personnel.
For over two decades, TSL® has delivered innovative data capture solutions to Fortune 500 companies around the world using a global network of distributors and system integrators. Specialist in-house teams design all aspects of the finished products and software ecosystems, including electronics, firmware, application development tools, RF design and injection mould tooling.
TSL® is an ISO 9001:2015 certified company.

ISO 9001: 2015

CONTACT

Address:	Technology Solutions (UK) Ltd, Suite A, Loughborough Technology Centre, Epinal Way, Loughborough, Leicestershire, LE11 3GE, United Kingdom.
Telephone:	+44 1509 238248
Fax:	+44 1509 214144
Email:	enquiries@tsl.com
Website:	www.tsl.com

ABOUT HID GLOBAL

HID Global powers the trusted identities of the world's people, places and things. We make it possible for people to transact safely, work productively and travel freely. Our trusted identity solutions give **people** convenient access to physical and digital **places** and connect **things** that can be identified, verified and tracked digitally. Millions of people around the world use HID products and services to navigate their everyday lives, and billions of things are connected through HID technology. We work with governments, educational institutions, hospitals, financial institutions, industrial businesses and some of the most innovative companies on the planet. Headquartered in Austin, Texas, HID Global has over 4,000 employees worldwide and operates international offices that support more than 100 countries. HID Global is an ASSA ABLOY Group brand.
For more information, visit www.hidglobal.com.