V+ Custom Socket I/O Driver

SimPhonics

Table of Contents

Table of Figures

Table of Tables

1 Revision History

Ensure you have the latest release of this document before relying on this information. This document was authored using Microsoft Word 2007 and is maintained at the SimPhonics web site in .DOC format and HTML format. For your convenience this document may contain references to web addresses. If you are reading this document on a computer using Microsoft Word or compatible software, clicking these references will point your browser to a location on the web. If you are reading a hard copy, these references will appear as underlined blue text. See SimPhonics home page at www.simphonics.com for more information.

Table 1, Revision History

Version	Revision	Date
1.0	Initial Release	18May00
1.1	Added Notices for Blank Fields	February, 28 2002
1.2	Corrected miscellaneous	July 24, 2002
1.3	Added byte ordering notes	July 25, 2003
1.4	Added multicast and byte ordering selection along with various performance improvements.	March 23, 2007
1.5	Modified the dialog for the Destination/Multicast addressing.	December 12, 2007
1.6	Added a note about disconnected networked cards.	February 15, 2007
1.7	Added support for host names.	February 18, 2009
1.8	Document Cleanup and Switch to corporate template format.	2011-03-05
1.9	Ported to Windows 7. This version is the same as the one in Windows XP.	2014-03-28 - SWJ

2 Overview

The Custom Socket I/O driver is a UDP/IP socket interface that is customizable down to the bit level. Customization is done through a standard database format accessible through Open Database Connectivity (ODBC). The structure of the interface is defined using an ODBC-compatible database in the format given in the DATABASE FORMAT section of this document. When the driver is loaded, the user specifies one or more ODBC Data Source Names (DSN) that will be used to define the individual I/O devices. Each device, then, is available in V+ as a set of I/O ports as defined in the database. While the Custom Socket I/O Driver can be used with any ODBC-compatible database, this document is written assuming Microsoft Access nomenclature and capabilities.

2.1 Important Notes

1. Using this feature, interesting configurations are possible. For example a device can input data on a given network into V+ and output that data in a different format on a different network interface effectively creating a network bridge.
2. Even though all devices share the same Ethernet settings, each device creates a separate Ethernet socket. When modifying the database, ensure that all field names have valid text. Any fields left blank will cause errors in the I/O device. For example, the host buffer table cannot have a HostName field left blank, or the I/O device will generate an error, and the I/O device will only load V+ ports up until the blank field in encountered.
3. The most efficient way to implement the host side of the Ethernet driver is to use the same database for generating an Ethernet UDP packet for transmission to the V+ device. This ensures that both sides of the interface use the same data description.
4. Do not change the filename of the database since this is usually setup as a system DSN under windows ODBC.
5. If fields are added to the database, add them to the right of the last column, since the V+ I/O device uses indexes to fields, and adding a field on the left will cause V+ to be confused as the layout of the database.
6. The Driver can have as many devices as the user wishes to configure. Each device has a separate network configuration. It is possible for several devices to operate on different Ethernet network cards since they each have a separate bind address field in the configuration.

2.2 INPUT/OUTPUT

If the database specifies an input device, the Destination Address and Destination Address check box will be disabled.

2.3 BYTE ORDERING

This device can be programmed to use network or host byte ordering. SimPhonics UDP/IP device uses host byte ordering. Network byte order is the most common byte-ordering scheme for transferring data via Ethernet networks.

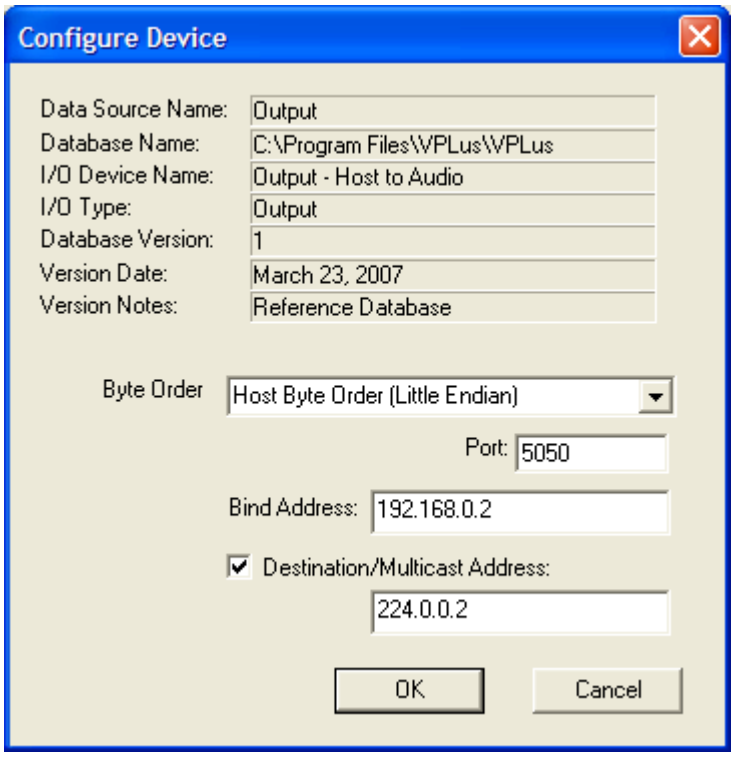
2.4 DATABASE FILE I/O

The database file is opened for reading only during initialization. Once the information is read into internal buffers, the database file is closed so that other devices or users may use the file.

2.5 NETWORK CONFIGURATION

This device is designed to use a single network adapter for communications. If there is only one network adapter then that adapter will be used by default. If more than one network card is available, then all adapters may be used unless the user has specifically bound to a given network adapter's IP address resulting in multiple packets being transmitted. In order to use a given network adapter, enter either an IP address or host name in the Bind Address field (see **Error! Reference source not found.**).

Figure 1, Device Configuration Dialog



2.5.1 Destination/Multicast Address Selection

The **Destination/Multicast Address** is a multipurpose edit box. If the checkbox is un-checked, then the value of the destination address will be 255.255.255.255, which is a broadcast address. This field can also be set to either a Unicast or Multicast address to send packets to a specific computer or group of computers. Host names may be entered in this field as well.

Note: *If you bind to an address and that address does not exist for a Network Interface Card (NIC) in the system, it will generate an error. It will also generate an error if the NIC is disconnected.*

2.5.2 MULTICAST ADDRESSES

IP addresses in the range of 224.0.0.0 through 239.255.255.255 are reserved for multicast. The range of addresses between 224.0.0.0 and 224.0.0.255, inclusive, is reserved for the use of routing protocols and other low-level topology discovery or maintenance protocols, such as gateway discovery and group membership reporting. Multicast routers usually do not forward any multicast datagram with destination addresses in this range, regardless of its TTL. Therefore use a multicast address in the range of 224.0.1.0 and 239.255.255.255.

If a valid multicast address is entered, then multicast operation will become active. The Multicast TTL (Time-To-Live) is hard coded at 5. In theory, time to live is measured in seconds. In practice, the TTL is reduced by one on every hop. That is why this field is named "hop limit" in IPV6.

2.6 Database Format

In order to be compatible with the Custom Socket I/O Driver, a database must contain the following tables:

DataTypes
Directions
Masks
Info
Version
HostBuffer
VplusPorts
ByteOrder

Each table must be formatted in the manner as described below. Some tables are used as static reference tables within the program and must contain certain data. Others are to be filled in depending on the particular use of the application. In many cases, data from one table is cross-referenced (or related) to data in another table. For more information on creating cross-references, or relationships, refer to your database's User Manual. Additional data may be added to the database for informational or other purposes, but must at least contain the following data:

2.6.1 DataTypes

A static table used as a cross reference for the **HostBuffer** table. The table must have the following format and contain the following data:

Table 2, Data Types

ID	DataType	Description
1	char	8-bit signed integer
2	uchar	8-bit unsigned integer
3	short	16-bit signed integer
4	word	16-bit unsigned integer
5	long	32-bit signed integer
6	dword	32-bit unsigned integer
7	float	32-bit IEEE single precision float
8	double	64-bit IEEE double precision float

where:

ID = long integer value
DataType = text string
Description = text string

2.6.2 Directions

A static table used as a cross reference for the **Info** table. The table must have the following format and contain the following data:

Table 3, Direction Types

ID	Direction
0	Input
1	Output

where:

ID = long integer value
Direction = text string

2.6.3 Masks

A static table used as a cross reference for the **VplusPorts** table. This table is not absolutely necessary for proper operation, but provides a convenient way to store the bit masks that may be used. A recommended format and data values for the table is shown below:

Table 4, Bit Masks

MaskValue	Mask
1	Bit 0
2	Bit 1
4	Bit 2
8	Bit 3
16	Bit 4
32	Bit 5
64	Bit 6
128	Bit 7
256	Bit 8
512	Bit 9
1024	Bit 10
2048	Bit 11
4096	Bit 12
8192	Bit 13
16384	Bit 14
32768	Bit 15

where:

MaskValue = long integer value
Mask = text string

Notice that the **Masks** table shown above could be expanded to include Bits 16 through 31, or could include other bit combinations. The important thing to remember is that the MaskValue parameter will be used by the I/O driver to extract one or more bits from a Host value by ANDing the value from MaskValue with the Host value (this is further explained in the **VplusPorts** table description).

2.6.4 Info

A user-configured table that must contain the following columns:

Table 5, Info Record

ID	DeviceName	Direction

where

ID = long integer value
DeviceName = text string
Direction = long integer value (cross-referenced to **Directions** table)

This table contains information about the I/O device and it should only contain one record. The ID parameter can be an arbitrary value. The DeviceName parameter is the name of the I/O device as it will be labeled in V+. The Direction parameter is a cross-reference to the **Directions** table that specifies the I/O direction of the I/O device. A device specified as an Input expects data to be sent from the Host to V+, and a device specified as an output expects data to be sent from V+ to the Host.

2.6.5 Version

The version table must contain only one record and is user defined. It contains information only but must contain the following columns:

Table 6, Version Table

ID	Version	Date	Author	Company	ProjectCode	ProjectName	Notes

where

ID = long integer value
Version = long integer value (must contain an integer value)
Date = text string
Author = text string
Company = text string
ProjectCode = text string
ProjectName = text string
Notes = text string

All of these fields can be left blank with the exception of the version field.

The information in this table is shown to the user when the database is first loaded into the V+ Platform Shell. The table may contain more than one record but only the last record is shown when the device is loaded, so the most recent change information should be placed in the last record of the table.

2.7 Version Compatibility Issues

The version parameter is actually a schema version. The Version parameter is important because the I/O driver checks this value to make sure it is compatible with the database format. Currently, the Custom Socket I/O Driver is compatible with database version 1 or lower. Future database schemas that are not backward compatible with older versions, should increment this version number in order to stop the loading of the database file. An error message is generated when this occurs, "This database has a version of n, This I/O Driver is compatible with version m only." where n is the schema version of the loading database, and m is the schema version that the driver wants to see.

2.7.1 HostBuffer

This record describes the structure of the network packet to be transferred by the socket and must contain the following fields:

Table 7, Host Buffer Table

HostRef	HostName	DataType	Notes

where:

HostRef = long integer value
HostName = text string
DataType = long integer value (cross-referenced to **DataTypes** table)
Notes = text string

The records in this table describe the structure of the packet that will be sent between the Host and V+. Each record in this table represents a single element of the packet. The **DataTypes** table shows the valid data types for each element. The smallest element that can be represented in the packet is an 8-bit integer, although individual bits can be masked out by setting appropriate parameters in the **VplusPorts** table.

The HostRef parameter is an arbitrary number that uniquely identifies each element in the packet. This value can be a Host offset that has some significance or simply a count from 1 to n. In any case, the elements should be numbered so that the lowest values are the first elements in the buffer and in ascending order to the last elements. The HostName parameter is used only as reference and is not read by the I/O driver. It is used, however, by the **VplusPorts** table as a cross-reference label and is useful whenever printing the table to create an Interface Control Document. The DataType parameter is a cross-reference to the **DataTypes** table. This parameter tells the I/O driver how many bytes to allocate and how to handle the data in the buffer. The Notes column is a place to add notes to clarify any element in the packet. It can be left empty. This part of the record does not show up in V+ and is only used for informational purposes.

2.7.2 VplusPorts

A user-configurable buffer that describes the way the Host buffer is translated into V+. The table must contain the following columns:

Table 8, V+ Ports Table

VplusPortNo	VplusPortName	HostRef	Mask	Notes

where:

VplusPortNo = long integer value
VplusPortName = text string
HostRef = long integer value (cross-referenced to the **HostBuffer** table)
Mask = long integer value (cross-referenced to the **Masks** table)
Notes = text string

The records in this table describe the way the Host buffer is translated into V+ data ports. Since V+ ports are always 32-bit IEEE floating point values, this table defines how the elements in the Host buffer appear in V+. The VplusPortNo parameter is the identifier of the port as it will be in V+. The ports will be shown in ascending order in V+. The ports do not need to be in the same order as the Host buffer. More than one port can point to the same Host buffer element. The VplusPortName parameter is the text name that will be shown in the V+ design. The HostRef parameter is a long integer value that is a cross-reference to the **HostBuffer** table. It defines which Host element that the V+ port is attached. The Mask parameter allows the V+ port to mask out a bit from the Host element so that the V+ port can access any Host element down to the bit level. For example, assume that HostElement1 has been defined as a 16-bit word in the **HostBuffer** table. Then VplusPort1 can refer to that element with a Mask equal to 4 to extract Bit 2. VplusPort2 can also refer to the same Host element but with a Mask equal to 32 to extract Bit 5. If the Mask field is empty, then no masking is applied to the Host element. Also, masks do not have any effect on "float" or "double" data types.

2.7.3 ByteOrder

A static table used as a cross reference for the **Info** table. The table must have the following format and contain the following data:

Table 9, Byte order

ID	**Byte order**
0	Network Byte Order
1	Host Byte Order

where:

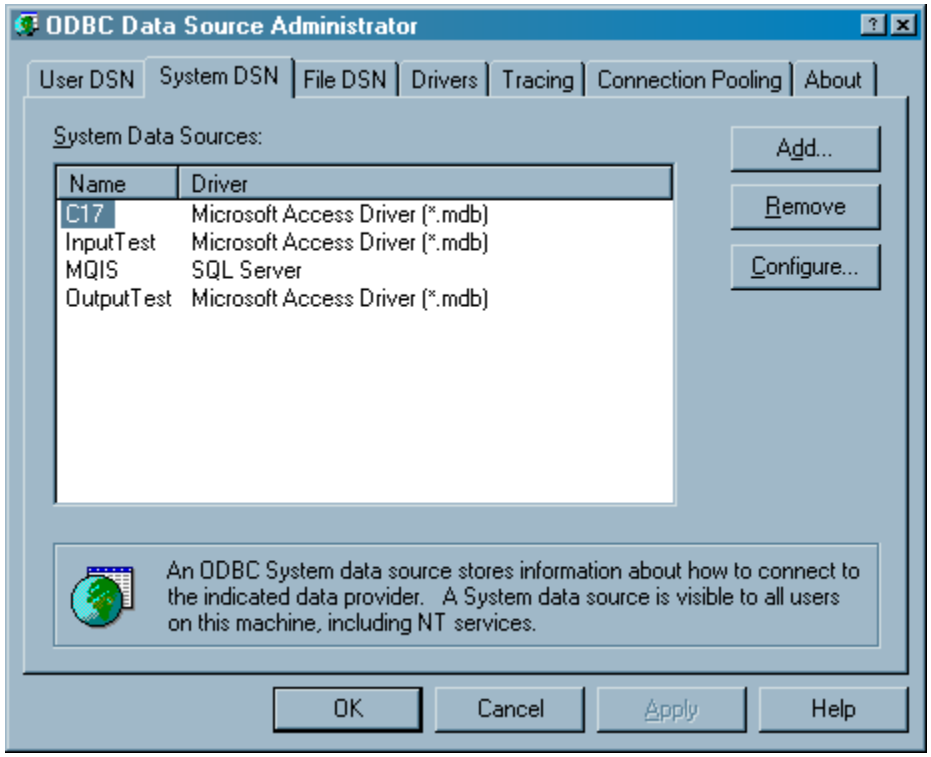
ID = long integer value
Byte order = text string

2.8 Configuring Data Source Names

In order for the Custom Socket I/O Driver to open a database, an ODBC Data Source Name (DSN) must first be configured for the system. Setting up a DSN is a fairly simple process consisting of the following steps:

1. Open the Control Panel by clicking on: Start/Settings/Control Panel
2. Open the ODBC Data Source Administrator by double-clicking on the ODBC Data Sources (32bit) icon
3. Click on the System DSN tab
4. Click on the Add button to insert a new DSN
5. In the next window, select the driver for the database type (i.e. Microsoft Access Driver for Access databases) and click Finish
6. Enter a Data Source Name in the space provided. The DSN should be a descriptive name for the database since this is part of the name that will appear in V+ when the device is loaded (the I/O device name in V+ will be "DSN - DeviceName" where DeviceName is read from the **Info** table of the database)
7. Enter a description of the DSN (not required)
8. Click the Select button and select the actual database file that this DSN represents and click OK
9. Click OK to add the new DSN to the System Data Sources list
10. Click OK to close the ODBC Data Source Administrator and close the Control Panel

Figure 2, ODBC Configuration Dialog



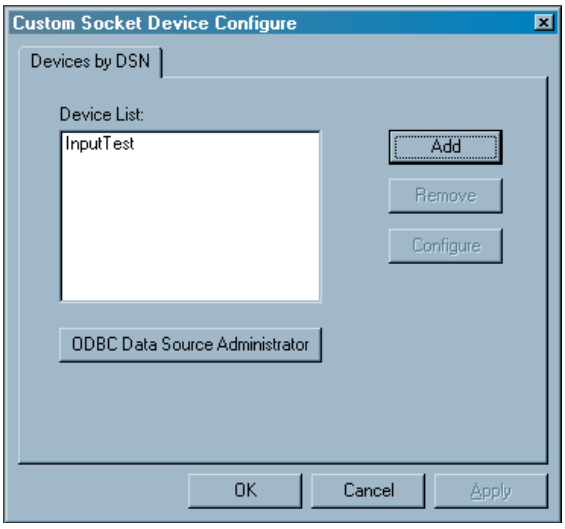
The figure above shows an example of an ODBC Data Source Administrator with four DSNs assigned. The name in left-hand column is the Data Source Name, the name that will be used later in V+. The ODBC Data Source Administrator can also be used to remove DSNs and to reconfigure DSNs that already exist. When you load the Custom Socket I/O Driver in V+ and go to its Configure page, there is a shortcut button to the ODBC Data Source Administrator. You can click this button to go directly to the Administrator without having to do steps 1 & 2 from above.

2.9 Loading a Custom Socket I/O Device in V+

Once a database has been created and a Data Source Name assigned, it can be loaded into V+ as an I/O device. The following steps outline an example procedure for loading a Custom Socket I/O Device:

1. Launch V+ and go to the V+ Run Time System window
2. Click on Configure to open the Platform Configure window
3. A list of available I/O devices should be shown, and among them should be the Configurable Network Socket driver
4. Double-click Configurable Network Socket from the driver list to load the driver (the icon to the left of the text should turn green to indicate a successful load)
5. Click the Configure button to open the configuration page for the driver
6. Click Add to insert a new I/O device
7. Enter the name of the Data Source Name that was assigned in the previous section (if no DSN has been assigned, use the shortcut button at the bottom of the configuration window to launch the ODBC Data Source Administrator and follow the instructions in the previous section)
8. When you click OK from the Add Device window, the driver will attempt to load the database from the DSN; any errors in configuration will be reported here
9. If the database loads successfully, a Device Configure window will be shown that displays some of the data from the database (name, version, etc.); from here, you can also customize the instance data such as network port number and broadcast address (outputs only)
10. Click OK to add the device to the device list for this driver
11. Click OK to close the Configure window for the driver; you are now ready to use the I/O driver.

Figure 3, Device Selection



2.9.1 Data Source Names and Driver Names

The name that will be used by V+ to identify the I/O device(s) that you have created is a concatenation of the Data Source Name followed by the name of the device assigned in the database. For example, if you have a database that defines an I/O device named "ARC-190 Radio Panel" and you have configured a DSN for that database named "HF1", then the V+ I/O device name will be: "HF1 - ARC-190 Radio Panel".

2.9.1.1 Multiple Devices

It is possible to create more than one DSN that refers to the same database. This enables the creation of a single database that may have more than one instance (I/O device) in V+. Continuing from the example above, suppose we created a second DSN pointing to the same database named "HF2", then the V+ I/O device name will be "HF2 - ARC-190 Radio Panel". Both I/O devices will be available in V+ and behave as separate I/O devices. Therefore, it is possible and sometimes necessary to create several Ethernet interfaces on a single V+ machine.