DMX512 Protocol

The DMX protocol as implemented provides a unidirectional data transfer from a stage controller system to multiple controlled systems, including lighting and fog equipment. The protocol does not inherently provide data checking, and is not intended to control any safety critical equipment. Some controllers provide the means to send checksum and/or multiple copies of the data to provide some means of data checking.

The DMX protocol is defined by Entertainment Services and Technology Association (ESTA) in American National Standard E1.11 – 2004 as USITT DMX512-A "Asynchronous Serial Digital Data Transmission Standard for Controlling Lighting Equipment and Accessories."

The full specification may be obtained from http://www.esta.org/ or http://webstore.ansi.org

This document describes the operation of the SilverDust under the DMX512-A protocol. This document describes our best interpretation of the E1.11-2004 specification - these are only guide lines. The user should refer directly to the specification in verifying their own system implementation.

It is assumed the reader is familiar with DMX512 and programming SilverLode servos. See the SilverLode User Manual, chapter 1 for a tutorial.

Part Number

This protocol requires a specially configured SilverDust processor. This configuration requires the use of IO6 internally. IO6 should not be connected or used in DMX units. DMX is specified by adding a "D" to the part number "Options" field. See the price list for details. The following are examples of SilverDust with the DMX option:

QCI-D2-IG-D
QCI-D2-IG8-ED
QCI-D2-IG8-DEM
QCI-D2-IG8-DEMS
QCI-D2-IGF-D
QCI-D2-IGC-D
QCI-D2-MG-D

Hardware

The DMX Data Complement (Data-) should be connected to B/Tx, and the Data True (Data+) signal should be connected to A/Rx.

Configuring the SilverDust to Receive DMX Data

There are two steps to configuring the SilverDust to Receive DMX data. The first is configuring the serial port for DMX and the second is defining how the DMX data is to be mapped into the controller's registers.

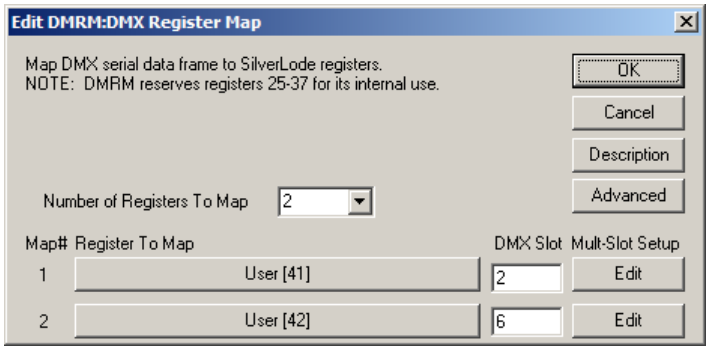
Configure Serial Port For DMX

The serial port must be configured for 250K Baud, RS-485, and DMX Protocol using the BRT, SIF, and PRO commands respectively. The code on the right is from the included example program "QCI-AN045 DMX512 Protocol-Simple.qcp".

2:REM		Configure for: 250K Baud Rate RS-485 Serial Interface DMX Protocol
3:BRT		Baud Rate Divisor = 2500
4:SIF		Serial Interface = RS485
5:PRO		Protocol = DMX 2 Stop Bits, No Parity

Mapping DMX Data To Registers Using QuickControl's DMRM Command

QuickControl's DMX Register Map (DMRM) command provides an easy way to map DMX data to 1-6 registers. The example below shows 2 Register Maps to registers 41 and 42.



Number of Registers To Map

Select the number of registers to map. Valid range 1 to 6. NOTE: The dialog box will expand as more Register Maps are selected.

Register To Map

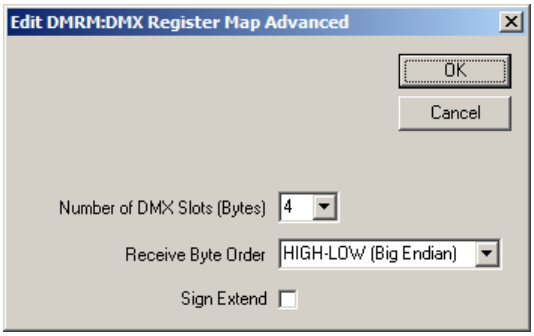
Select the register number you want the data mapped to. Valid range 40-199.

DMX Slot

Enter the starting DMX Slot number for this Register Map (see below for details on DMX Slot).

Mult-Slot Setup

Number of DMX Slots to use for this Register Map (1-4)

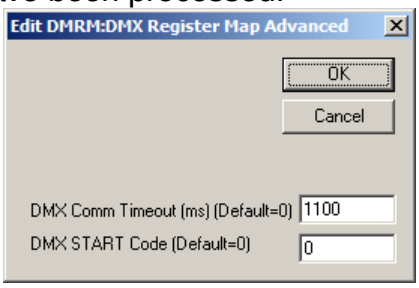


Receive Byte Order: 0= Big Endian (HIGH-LOW), 1=Little Endian (LOW-HIGH). Big Endian transmits the most significant byte first. Little Endian transmits the least significant byte first.

Sign Extend: 0=No Sign Extend, 1=Sign Extend
Sign extension only is applied for 1,2, or 3 byte values being written into a 32 bit register. 4 byte transfers have no sign extension.

Advanced

DMX Communication Timeout: If this value is not zero, it is written to the Delay Register (Register 5) whenever all requested Register Maps of a frame have been processed. Excessive time between frames may be detected either by checking the status bit Delay Count Active/Exhausted, or by enabling the Delay Counter Exhausted bit in the Kill Motor Conditions command, after first waiting for the first valid frame. This will cause the Kill Recovery routine to be executed if the DMX master stops sending data.



DMX START Code (Default 0)

QCI-AN045 DMX512 Protocol-Simple.qcp

This example program demonstrates how to both configure the serial port for DMX protocol (lines 3-5) and to map 4 slots of DMX data to register 41 (line 8). It is assumed these 4 Slots contain position data, which the program uses to move the servo.

The Profile Move Continuous (PMC) (line 26) command is used to move the servo to the position contained in the DMX data (Slots 2-5).

2:REM		Configure for: 250K Baud Rate RS-485 Serial Interface DMX Protocol
3:BRT		Baud Rate Divisor = 2500
4:SIF		Serial Interface = RS485
5:PRO		Protocol = DMX 2 Stop Bits, No Parity
6:REM		Map DMX Stots 2 Through 5 to Register 41
7:WRP		Write 0 to "User[41]" Register
8:DMRM		DMX Register Map: Number Reg Maps=1 Map 1: DMX Slot=2, Reg=User[41]
19:REM		Initialize Profile Move Registers
20:EMT		Enable Multi-Tasking
21:WRP		Write 0 to "Profile Move Pos[20]" Register
22:WRP		Write 16000 cps/s to "Profile Move Acc[21]" Register
23:WRP		Write 80000 cps to "Profile Move Vel[22]" Register
24:WRP		Write 16000 cps/s to "Profile Move Dec[23]" Register
25:WRP		Write 0 to "Profile Move Offset[24]" Register
26:PMC		Profile Move Continuous:
27:REM	LOOP	Main Loop
28:REM		Copy DMX mapped register to Profile Move register
29:CLX		Profile Move Pos[20] = User[41]
30:JMP		Jump to "LOOP"

Since the data is mapped to register 41, the Main Loop (lines 27-30) continually copies register 41 to PMC's position register (register 20).

QCI-AN045 DMX512 Protocol-Advanced.qcp

This is an advanced example using DMX to stream move and output commands to the servo. The DMX timeout feature is also used to detect a loss of DMX data. See the program line remarks more details.

DMX512-A Protocol - Details

The DMX512-A protocol includes both hardware and software components:

Hardware:

- RS-485-A Compliant Transmitters And Receivers
- Proper Termination Of The RS-485 Style Bus
- Bus Wiring
- Connector Styles And Wiring
- System Grounding And Ground Interconnecting Resistances
- Distribution Networks For 485 Signal
- Unidirectional Vs. Bidirectional Signaling
- 250k Baud Rate
- 8 Data Bits 2 Stop Bits, No Parity

Software (see below for details):

- Start Of Frame By Means Of Break Signal
- First DMX Slot (Byte) – Slot 0 - Is “Start Code”
- Up To 512 Additional Data DMX Slots – Numbered As 1 To 512
- Multiple Successive DMX Slots May Be Combined

Data is sent in “Frames”. A Start Of Frame is denoted by forcing a “Break” condition – “Space” or Low level onto the 485 serial connection for a minimum of 88 microseconds, followed by a “Marking” or high level for a minimum of 11 microseconds. Data is then sent as 8 bit data with 2 stop bits and no parity at a 250 kbaud rate. The Start Of Frame Break is followed with data. Each successive data byte is called a “DMX Slot” or Slot, starting with Slot 0 and going to a maximum of Slot 512. The first data, Slot 0, is special as it designates the Start Code. The default Start Code is zero (0), but other Start Codes have assigned uses in the specification. A device is configured to respond only to its defined Start Code, with other special Start Codes optionally implemented. Data in frames following Start Codes other than the selected Start Code are ignored. If a Slot is received with improper stop bits (loss of framing), then that Slot and and all following are ignored until the next start of frame (break).

The frame to frame (break to break) time, as transmitted, should be less than 1 second to avoid timeout (if configured). The receiver should be configured to respond to a loss of data if the frame to frame time exceeds 1.25 seconds as a maximum. A shorter timeout time may be implemented by the user.

Update rate for a maximum packet (513 slots) is 44 updates per second – 22.7ms. Smaller packets may update at higher rates – shortest allowable packet is 1.204 ms.

The data in the various Slots from 1 to 512 are user configurable, at the master show controller as well as at the receiver (SilverDust). Data may be sent as 1, 2, 3, or 4 Slot (byte) (8, 16, 24, or 32 bit) values, and may be implemented as either Big-Endian (most significant byte first) or Little-Endian (least significant byte first).

Mapping DMX Data To Registers - Native

If more than 6 Register Maps are required you will have to initialize the DMX registers without the help of QuickControl's DMRM Combo-Command. This is done by configuring registers 25-32+ as follows and is demonstrated in the included example program "QCI-AN045 DMX512 Protocol-Native.qcp".

Reg#	Word	Format	Description
25	Both	U32	**Last Slot Processed Time (uSec)** Value representing the value of the microsecond counter when the last Slot was processed. Difference calculations may be used to determine frame time to time, if desired by the user.
26	Upper	U16	**Register Map Number Processed** Set to 0 when a new frame is detected, and is incremented after each data Register Map has been processed. It may be used to determine when data is available for use without waiting for the entire frame to complete.
	Lower	U16	**Checksum** Checksum from Slot 0 to last slot of last Register Map.
27	Both	U32	**Number Of Bad Characters** Register 27 is incremented each time a frame is lost due to a bad character or a start of frame (break) with slots not yet processed. The count will lock at maximum (4,294,967,295) rather than wrapping.
28	Upper	U16	**Parsing Error Code** 1: No Register Maps Requested 2: Register Map Register Number Not In Valid Range Of 40 To 199 3: More Than 100 Register Maps Requested. 4: First Slot Of This Register Map Has Already Been. Register Maps must be ordered so as to consume Slots in ascending order. No Slot may be used more than once.
	Lower	U16	**Register Number Of Register Map With Error**
29	Upper	U16	**Frame Received** Set to a 1 whenever the last requested Register Map has been processed. The user should set this word back to zero after detection if used to hand-shake with the DMX protocol. Alternately the upper word of Register 26 may be compared against the needed Register Map to see if it has been processed.
	Lower	U16	**Number Of Register Maps** 0=Disable DMX Protocol: Stop processing incoming frames.
30	Lower	U16	**Start Code**
31	Both	U32	**DMX Communications Timeout (120 uSec Ticks)** 0=Default (no timeout). See DMRM above for details.

An additional register, starting at register 32 is required for each Register Map.

Reg#	Word	Format	Description
32+	Upper		**Not Used**
		Bits:15:14	Sset to zero
			Number Of Slots
		Bits:13:12	0-3 represent 1-4 Slots (bytes)
			Sign Extend
			0=No Sign Extend, 1=Sign Extend
			Sign extension only is applied for 1,2, or 3 byte values being written into a 32 bit
		Bit 11	register. 4 byte transfers have no sign extension.
			Receive Byte Order
			0= Big Endian (HIGH-LOW), 1=Little Endian (LOW-HIGH)
			Big Endian transmits the most significant byte first. Little endian transmits the least
		Bit 10	significant byte first.
			Starting Slot
			This should be in the range of 1 to 512-(number of bytes-1).
		Bits:9:0	0 indicates no more elements to scan until next start byte.
	Lower		**Register To Map**
		U16	Register Map's register number. Valid range 40-199.

QCI-AN045 DMX512 Protocol-Native.qcp

This example shows how to directly use the DMX registers 25-32+ without QuickControl's Combo-Command DMRM. This is the same as the example "QCI-AN045 DMX512 Protocol-Simple.qcp" except the native WRP commands are being used instead of the DMRM Combo-Command. Note Combo-Commands consist of many native SilverLode native commands. See Combo-Command in Command Reference for details.

Hardware Configuration Details

For A2 Compliance (preferred for non-isolated compliance):
User to supply Chassis to Data Link Common (0v power supply) Resistor. It must be 2W, 100 Ohm +/- 20% only one per user chassis (may be shared by one or more SilverDust units. Capacitor bypass permitted (A4). (Power supply ground should not connect directly to chassis for compliance – only through above resistor, one per unit.

User supplied wiring ***must not*** cause Resistance between Data Link Common and 0v supply to exceed 100 ohms. Similarly, user supplied wiring ***must not*** cause the Resistance between any Data Link Common In and Data Link Common Out (Pass though connector) to exceed .2 ohm.

Port must be marked NON-ISO or Non-Isolated per section 10.5

For A3 Compliance (less preferred):
Power supply may be directly connected to the chassis (no 2W 100 ohm resistor). However, port must be marked Per section 10.5 of Specification (warning marker with earth and chassis grounds connected) – See specification.

For both A2 and A3 Compliance:
Communications connector: User should mount 5-Pin XLR connectors for input and output to system, if possible. Male connector required on receiving device, with optional loop-through output connector. If no such connector is implemented, must be marked as NCCDMX512-A (Not Connector Compatible). See Section 7 of specification.
If a 5 pin XLR connector is used, it must be wired as

PIN 1 - Signal Common (Shield) (Connect to GND directly, and to Chassis with/without above 100 ohm resistor, according to wanted classification)
PIN 2 - Data (Dimmer) Drive Complement (Data 1 -) (connect to B/Tx)
PIN 3 - Data (Dimmer) Drive True (Data 1 +) (connect to A/Rx)
PIN 4 - Optional Second Data Link Complement (Data 2 -)
PIN 5 - Optional Second Data Link True (Data 2 +)

Bus topology RS-485 Termination
The DMX signal is a differential 485 signal, and should be run as a linear bus topology with a controlled impedance shielded twisted pair, with minimal stub lengths. It should terminated at both ends with a 120 ohm resistance. See DMX 512 specification.

Enabling DMX On a Non-DMX SilverDust (Engineering Work Around)
In as small, well controlled environments, standard SilverDust units may be configured for DMX by connecting IO6 to the B/Tx signal. This configuration does ***not*** provide the full RS-485 compliance range, however, and should not be marked as DMX compliant. This mode maybe useful for software testing with a standard unit in engineering.