Handbook for Communications Engineering I Laboratory

Preface

This set of lab activities centre around the design and measurement of the characteristics of a serial control bus. In performing the course work, you will learn: how the bus operates, how signals are processed and how to use a scope to make measurements on this communications bus.

The laboratory is divided into a series of activities.

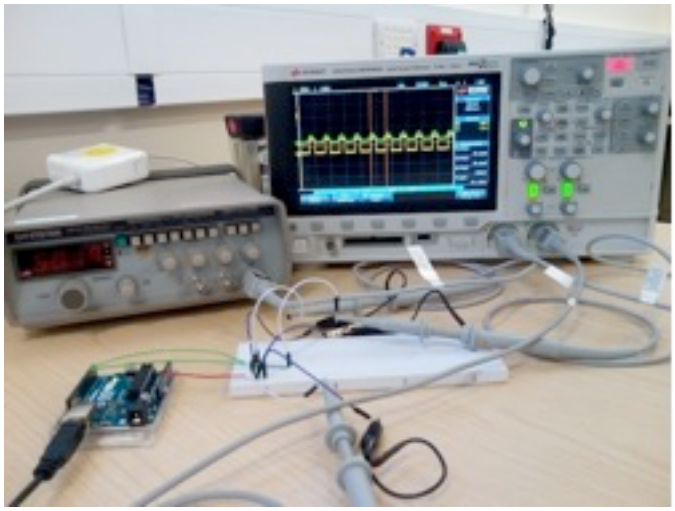
The first two activities develop understanding of the EIA-232 and EIA-485 busses. These labs focus on particular line driver chips.

The remaining activities focus on the Digital Multiplex bus, known as DMX-512A. Students are referred to a copy of the *Recommended Practice for DMX512*. This explains the DMX-512 specification and offers examples and professional advice on how to design and set up a successful system. People designing and making measurements on a DMX512-A compliant equipment must also refer to the *ANSI E1.11-2008 (USITT DMX512-A)*, *ANSI E1.20-2010 (RDM) Standards* and the *EIA-485 (RS-485)* standards. Both should be cited in any report produced for this lab work, along with any other related standards.

When you complete the work, you will need to provide a report for this laboratory. This will request you to provide a technical note with a set of captured waveforms showing the scope display for the set of tests that will be specified at the end of the lab. There must be at least one captured trace for each case. The diagrams must be captured from the scope using the **USB interface** (most usually saved to a USB memory stick). Each diagram must also include appropriate notes to label each part of the waveform measured. All diagrams must clearly show the timing and voltage.

Tip: Save each waveform from the scope so that you can access them if called upon in the continuous assessment reporting at the end of the lab activities. You also need to verify that you have saved output in a suitable format (e.g., PNG) so the day can be displayed by your computer.

Tip: If you use a USB drive with a scope, note that USB drives can become corrupted, so be sure to copy data to your computer or some other permanent storage..



The labs use a professional oscilloscope, with features common to many high-end industry scopes. On-screen cursers can measure any value or the difference between any two signals.

Waveforms can be saved to a USB storage device in PNG, Excel, Word, and MATLAB. The scopes are software-upgradable to provide remote operation across a control bus, advanced protocol decoding, digital sequence triggering, and to perform automated test measurements.

Figure: Keysight 2000 Scope

Activity A: The EIA-232 Bus

The goal of this activity is to understand the transmit interface of an EIA-232 transceiver.

The course web site resources contain additional information that is useful.

Equipment

BNC to BNC leads Qty 4.
BNC "T Piece".
9V Power Supply.
EIA-232 Line Driver on board with 4 BNC plugs.
Square wave generator, set to 5V TTL output.

Background

This activity will familiarise you with the signal waveform for transmission and reception using EIA-232 using an oscilloscope and to observe the operation of an open-ended transmission line. You will learn how to use the scope and capture appropriate waveforms to a USB stick. Ensure that any waveforms you observe are recorded and are transferred to your own computer for storage after each lab.

The MAX232 series of transceivers are designed to provide an EIA-232 differential transceiver interface (formerly known as RS-232). This is an electrical specification of a point-to-point serial connection. One conductor (A) sends data referenced to the ground level. A digital signal with value 1 will be sent using the EIA-232 interface as a voltage level between -3V and -12V and a signal of 0 will be sent as a level +3V to +12V. The receiver detects the value of a baud by measuring the voltage against a +/- 3V reference level (referenced to ground).

This transmission can provide protection from up to 6V of noise! In reality, the signal will be attenuated by any cable connected to the output, and this will also impact the level at the receiver.

*Tip: For each lab activity read **all** the notes for an activity before you arrive at the laboratory. The additional information will provide useful background to ensure you make the most of each activity.*

Equipment is connected to the cable using a line driver integrated circuit. This chip has seen faithful use in many pieces of equipment. Each transceiver contains two transmit circuits, and two receive circuits. In this laboratory, we will use one pair of circuits to convert the EIA-232 signals to and from a TTL signal (3.5-5V) from the TTL output of a waveform generator.

Since we will use a direct cable connection to the scope, reconfigure the input channel to select 1:1 attenuation input. This is done by pressing the channel button (above the BNC connector for channel), and then using the soft menu to select the input impedance of the channel.

Tip: Always check the channel summary information - in colour on the right of display to ensure you have the correct channel input impedance selected. The channel on the scope could be configured in 10:1 mode, to match the scope leads supplied. This would cause voltage displayed for the channel to be ten times larger than the actual value when using a directly connected piece of equipment. The course videos explain more about how to setup and choose probes for the scope.

Equipment Configuration:

Connect the data side of the driver to a TTL square waveform generator (set to 10 kHz):

- Connect signal generator to the BNC "T" piece using a BNC cable.
- Connect another cable from BNC "T" piece to the **channel** 1input of the scope.
- Check the scope display to ensure the waveform generator is sending a square wave.
- Use the scope to measure the frequency of the signal and confirm that this is 10 kHz.

We will now look at the line transmitter.

- Connect the BNC "T" piece to the **TTL input** of the line driver (shown on channel 1).
- Connect a second BNC cable from the **EIA-232 output** to channel 2 of the scope.
- Power-on the EIA-232 driver using the 9-12V power supply.

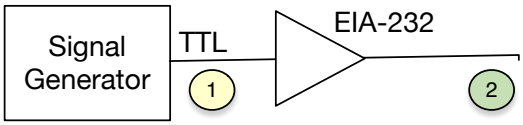


Figure: Diagram showing connections of the two scope channels to the TTL output and the EIA-485 line.

Tip: The EIA-232 board has a voltage regulator to produce 5V and an LED connected via a series resistor to this power supply. This can be used to confirm that the chip is receiving power.

A1: Observing the EIA-232 Signal

When you measure the voltage on the EIA-232 line, it may be less than the maximum voltage specified in the specification. This might be because of the choice of line transceiver, or the effect of the length of cable to which the transceiver is connected. In this experiment, we will use the Y cursors to accurately measure the voltage of the actual waveform on the line.

Experiment

- Measure the ***maximum and minimum digital voltages*** using the Y1 and Y2 cursors and measuring the TTL signal (channel 1) relative to the TTL ground.

  Activate the horizontal (X) cursors, by pressing the ***Cursor*** button.

  A set of soft buttons will now appear (at the bottom of the display). The ***Cursors*** soft button brings up a menu to select which cursors you will see on the display.

  Select the Y1 cursor by pressing the ***Cursor*** button, and a horizontal dashed line will appear. Use the "***Cursors***" control knob (or click on up/down when using the web interface) to position the Y1 cursor on the bottom of the waveform.

  Then, select the Y2 cursor by pressing the ***Cursor*** button, and a horizontal line with longer dashes will appear, position this, this time on the top of the waveform, again using the "***Cursors***" control knob (or click on up/down). The "cursors" display can show the difference (delta) between the values of the two cursors

- Examine the line signal at the output of the transceiver. Ensure you understand the output waveform and compare this with the input signal.

• Measure the ***maximum and minimum line voltages*** using the Y1 and Y2 cursors for the line, (channel 2) relative to the TTL ground. What is the peak-to-peak voltage of this signal?

*Tip: The **cursors** button, in the Measure area of the scope control, brings up a menu that can turn on a pair of horizontal (X1 and X2) or vertical (Y1 and Y2) dashed lines that you can move with the **Cursors** knob (or up/down controls with the web interface). The scope accurately displays the voltage difference (delta) between the two vertical cursor lines, or the time delta between the vertical cursors. This should be used to accurately **measure** the waveform.*

Tip: Save the display, so this can be used in your writeup. (The scope does not support large format sticks, so you may prefer to borrow a small capacity USB stick from the lab). Be sure to save this in PNG format to ensure you can later print a copy of the display.

Propagation delay is the time required for a digital signal to travel from the input(s) of a logic circuit to the output. In some designs, this is small enough to not be important, but it can be critical in some designs, such as the design a repeater for a half/full duplex signal.

• Use the scope to measure and record the ***propagation delay*** of the signal through the transmitter for EIA-232:

Set the X1 cursor on the centre of the TTL waveform.

You will need to adjust the horizontal time base to clearly see just the start of a baud. Now position the X2 cursor on the centre of the EIA-232 line level. Hence, measure the time it takes for the signal to propagate through the line transceiver.

This might be easier, if you enable a Y1 cursor, and set this to the centre of the voltage, making it easier to see when the X1 and X2 signals are aligned.

Experiment

Signals can also be measured in the time-domain using the X1 and X2 cursors. In these experiments, we will now use the X-cursors to make a measurement of the baud size.

•Next move the X1 and X2 cursors to measure the time period of one baud (one cycle Tb,) and hence determine the ***baud rate*****,** 1/Tb.

The slew rate determines the rise time. This is the time taken for a signal to cross a specified lower voltage threshold (usually 10% of the total change) followed by a specified upper voltage threshold (usually 90% of the total change). The output rise time (10% to 90%) is approximately equal to 2.2 RC, for a simple RC filter circuit. This is an important parameter in both digital and analog systems. In digital systems, the rise time describes how long a signal spends in the intermediate state between two valid logic levels.

• Use the scope to measure and record the ***rise time of the signal***:

Set the Y1 and Y2 cursors on the scope at voltages respectively 10% and 90% of the maximum EIA-232 line voltage that you have already measured.

Now, at the same time, you can set the X1 and X2 cursors to measure the time it takes for the signal levels to transition from 10% to 90% of the EIA-232 line.

•Confirm that this measured rise time is compatible with the required ***slew rate*** (1/rise-time) to support the baud rate of this signal.

A2: Frequency Response

This experiment explores the frequency response of the line transceiver using a function generator.

Experiment

Increase the input signal to 250 kHz and then 1 MHz. What happens as you increase the frequency? (Think about why the output changes and ask a demonstrator).

We will now look at the line receiver, using a loopback configuration of the EIA-232 signal.

- Disconnect all BNC cables and the BNC "T".
- Connect a BNC cable directly from the Waveform generator to the **TTL Input**.
- Connect the BNC "T" piece to the **EIA-232 output** of the line driver.
- Connect the "T" piece, using a BNC cable to channel 1 of the scope. This will show the signal on the EIA-232 interface.
- Connect another BNC cable from the "T" piece (EIA-232 output) to the **EIA-232 input**.
- Directly connect the **TTL output** of the line driver to scope channel 2 using a BNC cable.

The signal will now follow this loopback path:

Waveform Generator -> TTL-Input; EIA-232-Output (Ch1)-> EIA-232-Input; TTL-Output -> Ch2.

- Increase the wave generator input signal to 250 kHz and then 1 MHz. What happens as you increase the frequency? (Think about why the output changes).

Tip: Check you have recorded all the waveforms. You need to be sure to save your files in PNG format (or whatever format you require). Ensure you also note the axes and units for each of the waveforms. If you used a USB stick, copy the set of files from the USB stick to your computer.

Activity A Information: Using the Keysight 1000 Professional Scope

This shows the front panel of the smaller, more portable, scope. Each input is colour-coded and is enabled by pressing the numbered button above the corresponding input connector

Measurements

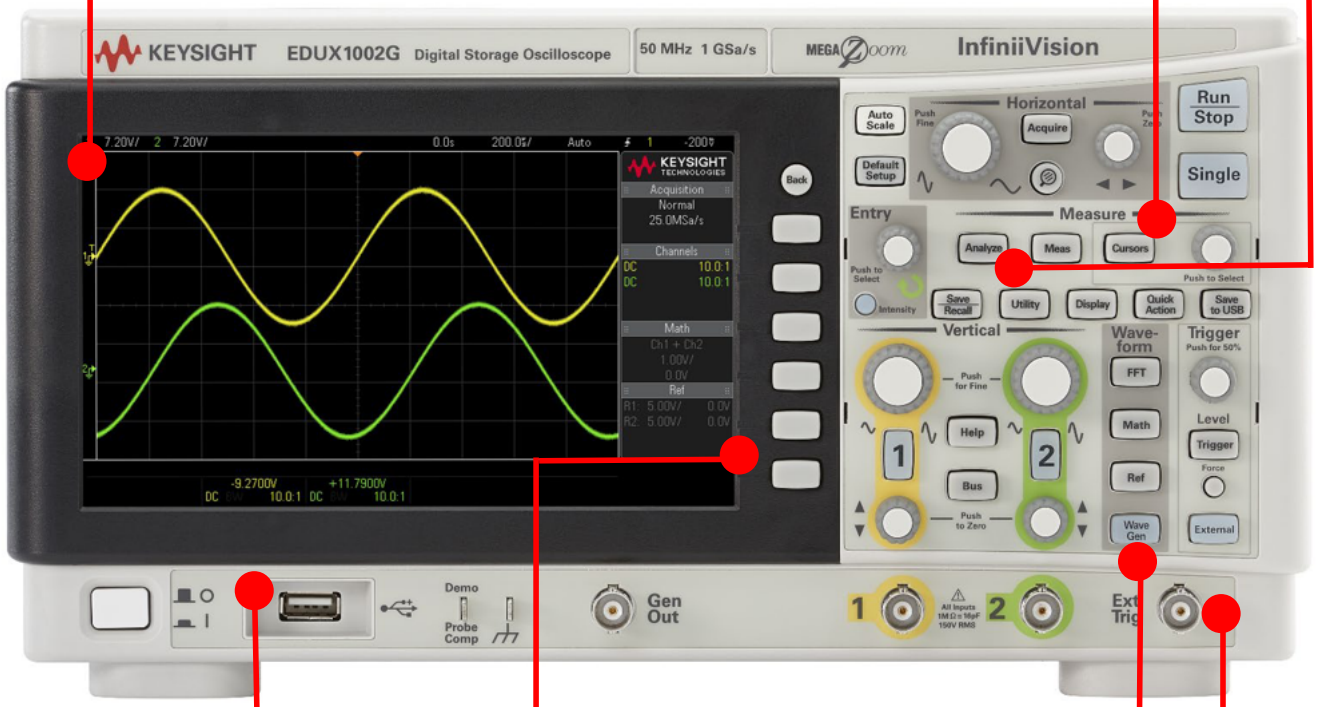
Press the measure key to access 24 built-in automatic measurements

Built-in localized-help

All buttons provide instant access to language-localized help by simply holding down the button you want explained

Cursors

Custom measurements are easily accomplished by cursors. Measure any value or the difference using four powerful cursors



USB save

Screenshots and date can be saved easily and fast with built-in USB port and your USB storage device.

Waveform Tools

Quick access to waveform math (+ − × ÷) and FFT. Reference Waveforms allow quick comparison of stored waveforms

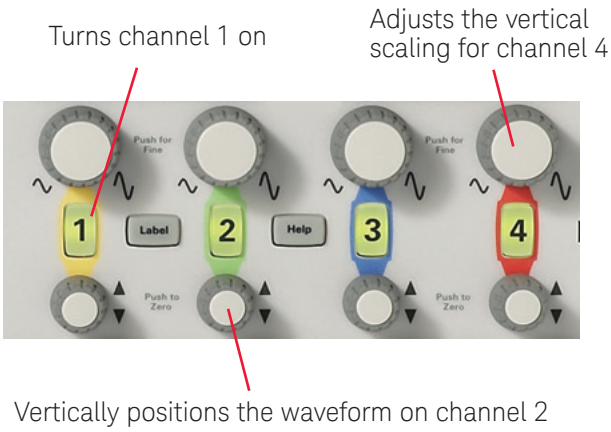
Industry leading user Interface

Fast and easy operation with the common oscilloscope controls right at your fingertips.

External Trigger

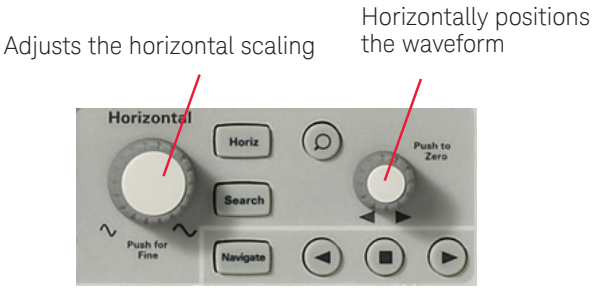
It can be used for triggering or displayed as a 3rd channel for a digital signal. It can also be used to create a 3-channal bus-type display

Vertical controls



Vertical controls on an oscilloscope typically are grouped in a section marked Vertical. These controls allow you to adjust the vertical aspects of the display. For example, a control designates the number of volts per division (scale) on the y-axis of the display grid. You can zoom in on a waveform by decreasing the volts per division or you can zoom out by increasing this quantity. There also is a control for the vertical offset of the waveform. This control simply translates the entire waveform up or down on the display.

Horizontal controls



An oscilloscope's horizontal controls are grouped in a front-panel section marked Horizontal. These enable you to make adjustments to the horizontal scale of the display. A control designates the time per division on the x-axis. Decreasing the time per division enables you to zoom in on a narrower range of time. There will also be a control for the horizontal delay (offset). This enables you to scan through a range of time.

Trigger controls

Triggering on your signal helps provide a stable, usable display and allows you to synchronize the scope's acquisition on a part of the waveform. The trigger controls let you pick your vertical trigger level (e.g., voltage at which your oscilloscope will trigger) and choose between various triggering capabilities. The Trigger Mode can select other ways to trigger scope, such as the width of a pulse.

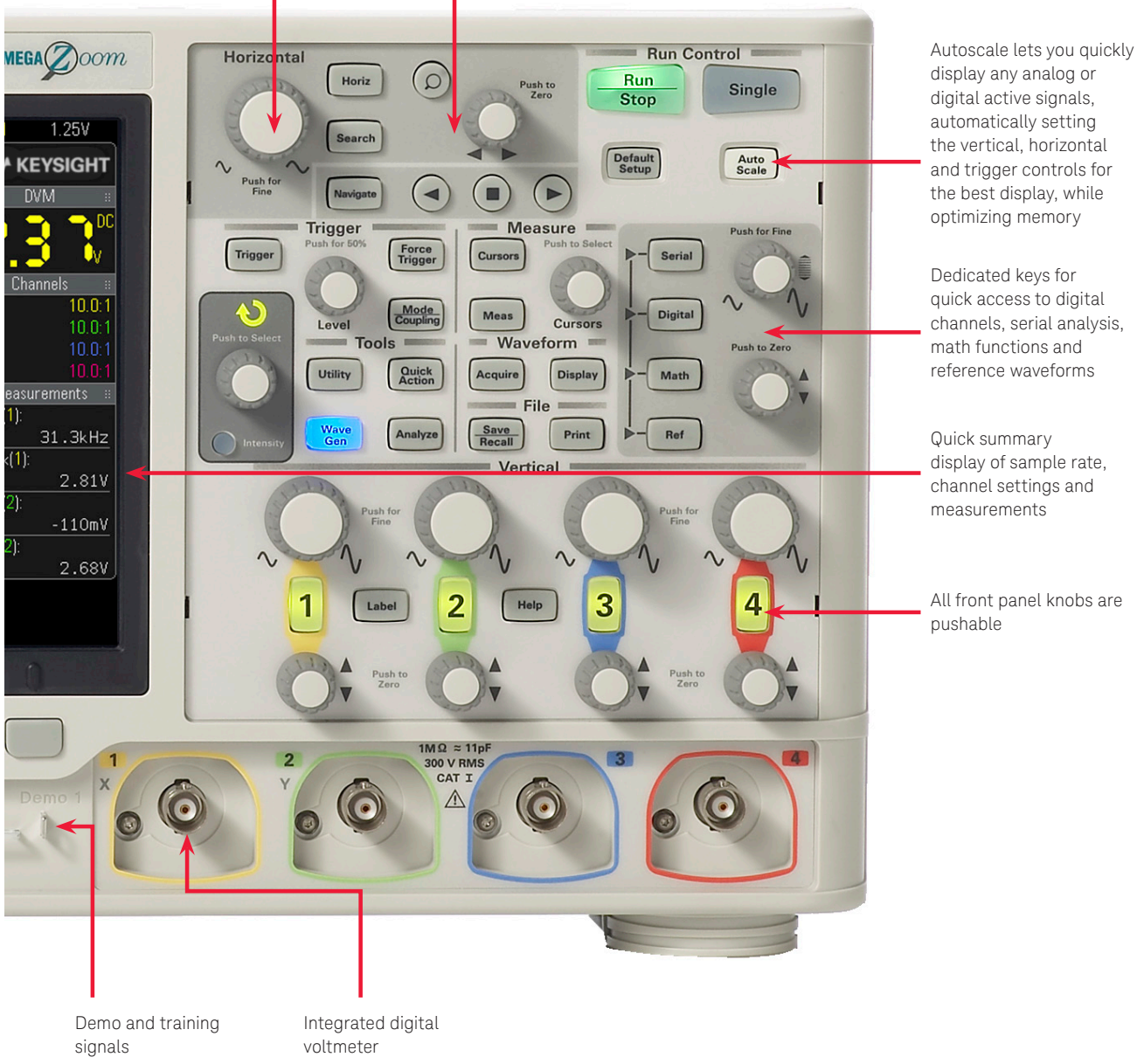
*Tip: The **autoscale** button quickly sets up the scope for you - but be aware that the scope does not know what part fo the waveform is of interest, so be ready to then adjust (1) the voltage (above the channel) (2) the timebase (marked horizontal) and (3) the trigger to to see the signal you require.*

Tip: Some oscilloscopes have a wavegen feature. If enabled, pressing the wavegen button will activate this output on the left of the scope. This in-built waveform generator could alternatively be used to generate the waveforms used in this experiment. One significant advantage of the inbuilt generator is that it can be remotely controlled for a web-enabled scope, enabling remote testing.

Activity A Information: Using the Keysight 2000 Professional Scope

This shows the front panel of the larger display scope. For this scope, the soft buttons are positioned in a row beneath the display.

*Tip: The **cursors** button in the Measure area to turn on a pair of horizontal or vertical lines that you can move with the Cursors knob. The scope accurately displays the voltage between the two vertical cursor lines, or the time between the vertical cursors. This can accurately **measure** the waveform.*



*Note: The remote web interface uses **up** and **down** buttons to replace the rotary knobs.*

Activity B: Asynchronous Frames

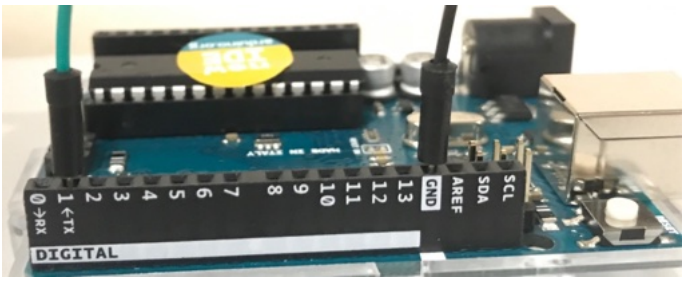
This activity will explore the waveform for an asynchronous frame using a programmable scope to decode the frame contents and to trigger the waveform on a specific character in the decoded frame.

Equipment

An Arduino emulating a GPS NMEA receiver & Arduino power supply
A serial TTL/EIA-232 Interface & power supply
Red and Black jumper leads, 1 BNC/crocodile, 3x BNC/BNC cable, 1 BNC T-piece.
1 Oscilloscope with a USB stick.

Background

The NMEA GPS specification defines a multiplex format based on frames of asynchronous bytes. Each frame is preceded by an ASCII dollar character. A receiver uses the character to start reception of a series of frames each containing characters representing the NMEA string/sentence. We will use EIA-232, sent at a baud rate of 4800 bps.from an Arduino that replays the NMEA stream generated from a previously-captured GPS receiver log. The signal from Arduino TX data pin (pin 1) is at TTL voltage level, and is converted to an EIA-232 signal, and then back to a TTL voltage.



Connect the Arduino transmit data pin (marked Tx -> 1) to a red jumper lead, and the Arduino signal ground (marked GND) to a second, black, jumper lead. Connect the BNC end of the a BNC to crocodile cable to the TTL BNC INPUT of an-EIA 232 Interface, and attach the crocodile clips at the other end to receive the corresponding signals from the Arduino.

Connect the EIA-232 signal: Connect a BNC T-piece to the EIA 232 Interface output to the input of the scope, and connect one side of the T-piece to the scope using a BNC/BNC cable (see (1) below). Connect a second BNC/BNC cable to the EIA-232 input of the EIA 232 Interface.

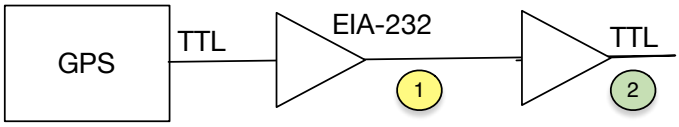


Figure: Diagram showing connections of the scope channels to the EIA-232 line and the TTL output of the transceiver.

Connect the TTL receive signal: Connect the TTL OUTPUT of the EIA 232 Interface to scope channel 2, using a BNC cable (see (2)). Add power supplies for the Arduino, and EIA-232 interface.

B1: Examining the serial bus data

The ***run*** button can be used to freeze the waveform, so you can see a sample of a changing display. You may need to also just the timebase to select an appropriate time/interval for the X-axis.

- Look at waveforms and look to see if you can see the idle time between characters.

Tip: The GPS receiver sends a signal that is nominal 4800 baud.

Decode one of the characters:

- Identify the idle (mark) before the start baud (this is one baud in duration).
- Identify the 8 bits of data within a character (this is eight bauds in duration).
- Confirm that the line returns to the idle state after the final baud of data (the stop baud).

Tip: Using an ASCII chart, you can decode the value of the 8 bits of data encoded lsb-first.

B2: Using a Scope as a serial bus protocol decoder

You may have observed that some of the characters, are a '$' character, but this is very hard to split when simply viewing the voltage level of the line. To look further, we need more help. Some advanced scopes (e.g. the 1000 & 2000 series scopes) provide a hardware serial bus decoder. This can decode the bus signal for a 4800 bps EIA-232 bus. Configure the scope to decode channel 1 as a serial bus at 4800 bps, asynchronous mode using EIA-232 (see additional information).

Tip: Check the on-line video tutorials have been provided by Keysight.

Configuring Serial Decode on the 2000 Series Scope:

You will now configure the scope to decode an NMEA GPS signal.

Press the ***Mode*** softbutton (below the display) and set to ***UART/RS232***.
Press the ***Bus config*** softbutton
The the ***#bits*** softbutton should be set to ***8*** bits/slot.
The ***parity*** softbutton should be set to ***None.***
Press the ***Baud rate*** softbutton and choose ***4800 bps*** from the menu.
Press the ***Polarity*** soft button
Set the polarity to ***Idle Low*** (The EIA-232 signal is inverted).)
The ***Bit Order*** soft button should be ***LSB bit order***
Press the ***signals*** softbutton.
Press the ***Rx*** softbutton and select ***channel 1***.
Press the ***Threshold*** softbutton and adjust to a value ***3.6V*** using the rotary knob.
(The channel needs to be set to 1:1 mode)
Press the "***back***" button once.
Press the ***settings*** softbutton.
The ***Base*** soft button can be set to ***binary, hex, or ASCII***
. The ***Framing*** soft button could be set to **0x24** (the hex value for an ASCII ' $').

Note: See appendix to this lab for a description for the 1000series.

*Note: On the remote on-line interface, the rotary knob is replaced by **up** and **down** buttons.*
*Press the **Serial Button** to activate the bus decoder.*

Tip: Important buttons are: (Back) [Analyze] [Mode] [Signals] [Bus Config] [Settings]

*Tip: The **settings** softkey allows this decode to be viewed in **hexadecimal** or **ASCII.** If the decode shown does not correspond to the expected character, check the polarity of the signal.*

B2: Experiment

After configuration, press the *run* button and observe the oscilloscope display. You should see that some slots/characters carry a '$' character. This is the frame delimiter for the start of a NMEA GPS string. Below the waveform, the scope will now also show a decode each byte seen on the EIA-232 line. The dollar characters are now highlighted.

- What is duration of one character sent using this format?

B3: Triggering on the waveform

The oscilloscope allows a user to set a trigger value to start the capture based on the decoded data, for example to look for a particular character in the data being displayed. Set the scope to trigger on the value at the start of frame, using the ***single*** run-mode display.

Press the ***Trigger*** button. (This activates the trigger soft menu)
Set the (left) ***trigger-type*** soft button to "***UART/RS232***".
Set the (next) ***trigger-setup*** soft button. (This changes the soft menu to trigger type).
Use the (left) softbutton ***Trigger*** to see the scope to ***Trigger Tx Data***
Press the (next) softbutton ***Data =***
Use ***the up/down*** controls to set a **hexadecimal value of 0x24**/ASCII '$'
The next button sets the display to ***ASCII*** or ***hexadecimal***.
Press ***back*** twice to return to the scope main display.
Set the scope to ***single*** mode (in the run-control area).

Tip: Additional information about other trigger modes is supplied in the Keysight tips (see Tip 5)

B4: Decoding the NMEA Sentences

Set the **horizontal timebase** to show many characters on the screen. To do this, decrease the ***horizontal*** time-base value, and press the ***single mode. You may need to reduce several times,*** until it shows a frame (NMEA sentence) each triggered run.

As you move the ***horizontal scroll knob*** (to the right of the horizontal timebase control), this allows you to move left and right through the decode the characters to read the NMEA sentence.

- Be sure you understand the format of the NMEA Sentences, and capture frames of characters.

*Tip: If you **press** the horizontal scroll button it resets the **position to zero.***
*Tip: The **Zoom to Selection** softbutton moves the X-axis to display the currently selected data.*

You should check the format of the strings against NMEA Sentences that have been standardised.

Lister - Supported on a Keysight 2000 scope

*Tip: The **Scroll Lister** softbutton helps scroll the display in **single** mode.*

Press ***serial*** button. (This activates the serial soft menu)
Press the ***lister*** softbutton (right). (This activates the lister).
Press the ***scroll lister*** softbutton.

In the lister window, upper half of the display shows a compact decode of the received characters.

Activity B Information: Setting up a scope to decode a serial bus

Tip: On line video tutorials have been provided by Keysight, and you are advised to check these before you arrive at the laboratory.

Some advanced scopes (e.g. the 1000 and 2000 series scopes) provide an integrated serial bus decoder. The following notes describe how to decode the bus signal for a 250 kbps EIA-485 bus.

Configuring Serial Decode on 1000 Series Scopes:

Press the ***Bus button*** *(on 1200 scopes press the* ***Analyse button*** *to bring-up the* ***Bus*** *menu)*
Change ***Bus type*** to ***Serial bus***.
Press the top button, then select ***serial bus*** (using the rotary knob).
Set ***Mode*** to ***UART/RS232*** (2nd button).

Configuring Serial Decode on 2000 Series Scopes:

Press the ***Serial Button*** to activate the bus decoder.
Press the ***Mode*** softbutton (below the display) and set to ***UART/RS232***.

Setup Serial Decode (all scopes):

Press the ***signals*** softbutton.
Press the ***Rx*** softbutton and select ***channel 1*** (using the rotary knob)*.
Press the ***Threshold*** softbutton and adjust to a value ***3.6V*** (using the rotary knob)*
(Remember to set the channel 1:1 mode , set using the **channel** button)
(You could also setup channel 2 also if you like, but will not use this.)
Press the "***back***" button
Press the ***Bus config*** softbutton
Press the ***#bits*** softbutton
Set the ***bits*** to ***8*** bits/slot.
Press the ***parity*** should be se set to ***None.***
Press the ***Baud rate*** softbutton and choose ***the required baud rate.***
If none of the rate are suitable, select *User Defined* from the menu.
Press the second soft button to configure the ***Baud Rate***
Press the ***Polarity*** soft button
Set the polarity to ***Idle High*** (for the A-line)
Press the ***Bit Order*** soft button
Set the bit order to ***LSB bit order***

You have now configured the scope to decode a serial signal.
Connect the BNC cable to input 1 (yellow) and press the ***run*** button.
Below the waveform, the scope now decodes each byte seen on the bus.

Tip: Important buttons are: (Back) [Mode] [Signals] [Bus Config] [Settings]

Tip: For a Keysight 2000 scope, the ***lister*** *softbutton displays the contents of the decoded data in summary, together with timing information (this can be displayed relative to the time of trigger, or slot-by-slot).*
Note: On the remote on-line interface, the rotary knob is replaced by ***up*** *and* ***down*** *buttons.*

Tip: In experiment D, you will need to decode a DMX waveform using a 250 k baud signal:
Press the ***Baud rate*** softbutton and choose ***User Defined*** from the menu.
Press the second soft button to configure a ***User Baud Rate***
Use the rotary knob* to set the User Baud Rate to ***250 kbps*** for DMX-512.

Activity C: EIA-485 Interface

Equipment

Application Note for the 75176 Line Driver[1].
Scope and 2 x scope probes
1 x BNC to crocodile lead
Breadboard with:
- Set of jumper wires
- Multimeter
- 5V Power Supply (confirm voltage is correct using multimeter before start)
- EIA-485 Line Driver: 75176
- 2 x120 ohm resistor
- Square wave generator, TTL

Background

This activity will familiarise you with the differential signal waveform using an oscilloscope and observe the operation of differential transmission. You will continue to use the scope and capture more waveforms for future reference. Ensure that any waveforms you observe are recorded and are transferred to your own computer for storage after each lab.

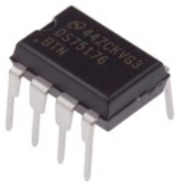


Figure : 75176 Differential Bus Transceiver IC

The 75176 transceiver Integrated Circuit uses an EIA-485 differential transceiver interface[2] (formerly RS-485). This is an electrical specification of a multipoint serial connection using differential signalling. Two tightly twisted conductors (A,B) form the transmission line.

A digital bit with value 1 will be sent using the twisted pair of wires when the potential of the positive wire (A) is greater than the potential of the negative wire (B) and a signal of 0 will be sent when the situation is reversed. The difference of potential between A and B must be different by at least 0.2 volts at a receiver for valid operation, but the sender uses a differential 5V voltage. This voltage is not referenced to ground and the ground level may be between +12V and -7V.

Differential transmission has two important benefits: Any interference that impacts the signal on the cable will have the same phase and magnitude in both wires that form the balanced line (A,B). Therefore the detected signal at the receiver (the difference in voltage between A and B) will be unaltered by the presence of such common mode interference. The differential signal also results in no net radiated signal from the cable. This reduces interference that could impact other nearby cables. The shield of the cable consists of conducting foil surrounded by wire stands that

---

[1] See: SNx5176B Differential Bus Transceivers, http://www.ti.com/lit/ds/symlink/sn75176b.pdf

[2] EIA-485 is a separate specification: Electrical Characteristics of Generators and Receivers for Use in Balanced Multipoint Systems. Electronic Industries Association 1985. Some guidance is provided in pages 68ff of the Recommended Practice for DMX512, but please also check the original specification.

completely enclose the A and B conductors. This electrical shield forms a Faraday cage around the enclosed twisted pair cable, particularly effective at higher frequencies.

*Tip: Always read **all** the notes for an activity before you arrive at the laboratory. The additional information will provide useful background to ensure you make the most of each activity.*

Sections of cable are typically joined together to form a single bus[3]. The end of each bus must have a 120 (or 110) ohm termination resistor (the same as the cable's characteristic impedance). Without this termination, reflections of the baud edges in the signal can cause data corruption.

Equipment is connected to the balanced cable using an EIA-485 line driver. Familiarise yourself with the specification of the chip. A separate sheet describes the frequency response and slew rate of a transceiver. This will be useful in explaining the shape of the waveforms that you observe when using the line driver.

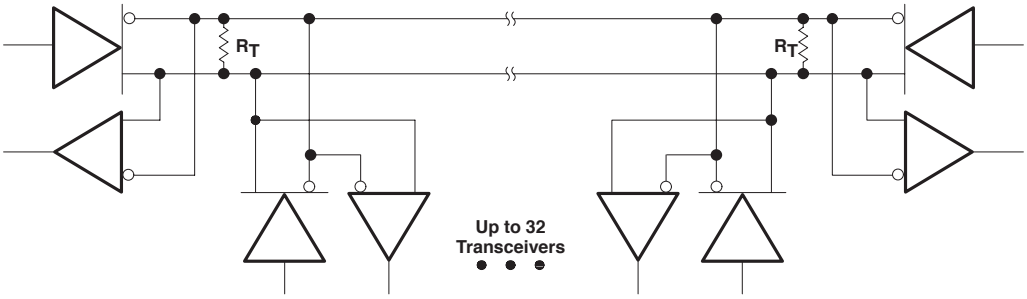


Figure: Typical application of a transceiver chip to drive the two lines of a differential bus.

The EIA-485 bus supports either one or two-way transmission, so in this experiment you need to ensure the tri-state control inputs are set to the **transmit** mode.

C1: Observing the EIA-485 output

In this experiment, we will examine the EIA-485 signal.

Equipment Configuration

- Connect the data side of the driver chip to a TTL square wave generator, using the BNC/ crocodile clip cable.
- Set the square wave generator to produce a 100 kHz square wave.
- Enable and disable the transmit interface by setting an appropriate voltage on the DE pin.

Note: The DMX 512 baud rate is 250 kbaud. The fastest changing data signal results when sending 101010 etc - i.e. alternating the polarity of each baud.

[3] [3] Page 18 of the Recommended Practice for DMX512 describes the pinout of the XLR connector.

Experiment

- Observe the resulting effect on the differential output.

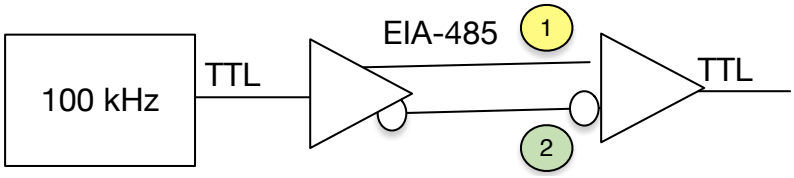


Figure: Diagram showing connections of the two scope channels to the two lines of the EIA-485 bus.=

The goal of this activity is to understand the signal from an EIA-485 transceiver. Attach the scope probes to channel 1 and channel 2 of the scope.

Observe the resulting waveforms:

- What is the maximum and minimum value of the A-line on channel 1?
- What is the maximum and minimum value of the B-line on channel 2?

Is this what you expect for a EIA-485 signal?

Tip: The channels on the scope should be configured in 10:1 mode, to match the scope leads supplied. (i.e. 1:10). If you have a switched probe then the voltage displayed for the channel to be ten times larger/smaller than the actual value. Watch the course video on configuring probes.

Tip: Remember to use the oscilloscope cursors to make accurate measurements.

*Tip: Set the run control to "**Stop**" and use the "**Single**" button to make this easier.*

Examine the transmit part of the EIA-485 driver.

- Use the scope to measure and record the ***rise time of this signal***:

  Set the Y1 and Y2 cursors on the scope at voltages respectively 10% and 90% of the maximum EIA-485 line voltage that you have already measured.

  Now, at the same time, you can set the X1 and X2 cursors to measure the time it takes for the signal levels to transition from 10% to 90% of the EIA-485 line.

- Confirm that this measured rise time is compatible with the required ***slew rate*** (1/rise-time) to support the baud rate for a DMX-512 signal.

Check also the additional information in the annexe to this lab.

C2: Observing the EIA-485 in Loopback

Experiment

- •Increase the signal to 250 kHz and then 1 MHz. What happens as you increase the frequency? (Consult the driver data sheet and think about why the output changes).
- •Place a 120 ohm resistor across the balance line, does this change the output waveform?
- •Place a second 120 ohm resistor also across the balance line, does this change the output?

This observes the two signals from the differential output using the two input channels of the oscilloscope.

This part of the activity uses the line driver chip in a configuration commonly known as *loopback*, in which the output of the transmit part of the line driver is fed to the input of the receive part of the same line driver. This configuration checks operation of both parts of the chip.

- Use the signal generator to send a 100 kHz TTL square wave into the Data Input (D) pin.
- This will generate a balanced differential output signal at the cable interface (A,B).
- Enable and disable the transmit interface by setting an appropriate voltage on the DE pin. Observe the resulting effect on the differential output.

The chip ***internally*** connects the differential output and input signals together. You do not need to do this, because the transceiver is already able to receive the signal you are sending.

- Change the connection to channel 2 of the scope so that it shows the received data output (R).
- With transmit interface enabled, enable and disable the receive interface using the RE pin and observe the resulting effect on the differential output.
- Repeat using a higher frequency square wave to illustrate slew-rate limiting by the driver chip. Note the resulting waveforms. Use the oscilloscopes cursors to make appropriate measurements.
- Measure the receive propagation delay through the receiver. How long is the propagation delay? (Check the data sheet for the line driver).

Tip: Save the display to a USB stick so this can be used in your writeup. (The scope does not support large format sticks, so you may prefer to borrow a small capacity USB stick from the lab). Be sure to save this in PNG format to ensure you can later print a copy of the display.

Tip: Check you have recorded all the waveforms for both pairs of balanced outputs for a 100 kHz signal and measured the chip propagation delay by the driver chip. You need to be sure to save your files in PNG format (or whatever format you require). Ensure you also note the axes and units for each of the waveforms. Copy the set of files from the USB stick to your computer.

Activity C Information: EIA-485 Signals

The differential transmission used by EIA-485 tries to minimize radiated emissions by using twisted-pair cabling and balanced transmitters. This has the effect of canceling each other out, and ideally results in no net radiated emissions. This tends to work fairly well, but, like everything in engineering, it isn't perfect. Inevitably, some radiated emissions will usually leak out. As a general rule, the higher the frequency components in the signal and the longer the cable, the worse the situation becomes, resulting in this being especially noticeable at higher frequencies.

The design of a EIA-485 physical layer must set a maximum baud rate appropriate for the link to select a driver circuit that is capable of running at the required speed. A circuit may be selected that is rated at a speed equal to or greater than the required baud rate. Knowing this, you might wonder if there are any disadvantages in choosing the fastest devices available, if they might be overkill. The answer is yes!

Although it is true that fast driver circuits can be used for both high and low baud rates, there are drawbacks in using a driver that supports a baud rate higher than needed. Figure 1 shows the signal (left) and the (right) Fourier transform for a driver specified for high-speed operation at many Mbaud. The frequency domain signal shows frequency components well past 2 MHz. These high-frequency components are necessary to produce nice square edges. (Note also the propagation delay through the driver circuit visible in (a).

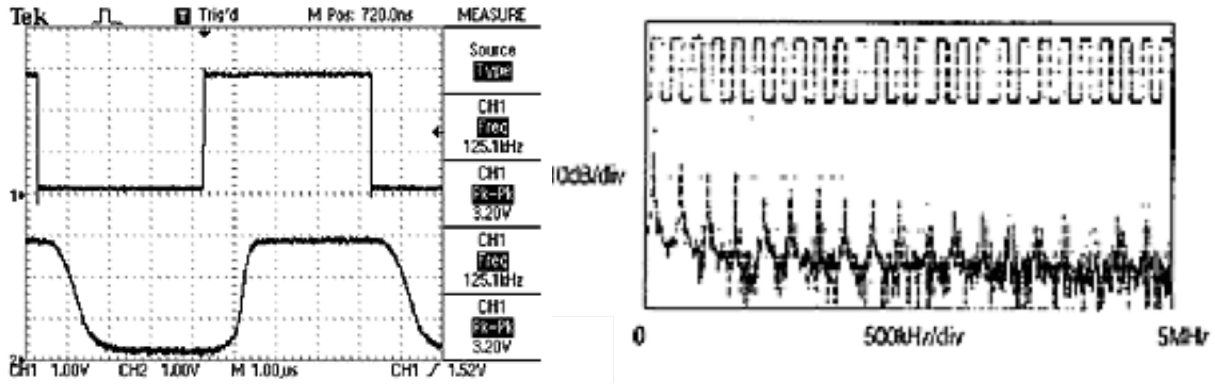


Figure: Plots for a 125kHz (250kbaud) test signal, without limiting slew-rate
Left: driver input (TTL) and non-inverted output waveform,
Right: driver output waveform and corresponding frequency domain signal

Cable terminations are also important[4]. Both ends of an EIA-485 cable should be properly terminated in the characteristic impedance of the cable to prevent reflections. Resistor and cable tolerances, among other things, can result in mismatches between these two impedances. This will result in reflections that increase the noise and can ultimately lead to corruption of data. Similar to radiated emissions, the higher the frequency components and the longer the cable, the more likely it is that reflections will affect the performance.

Most driver circuits include an output filter to control the slew-rate of the signal. This shapes the signal to reduce signal harmonics at frequencies high above the baud rate. Without this the signal will have extend to high frequencies and spikes in the frequency domain that make it much more likely to induce interference to other signals.

[4] Page 21 of the Recommended Practice for DMX512 describes the requirements for termination.

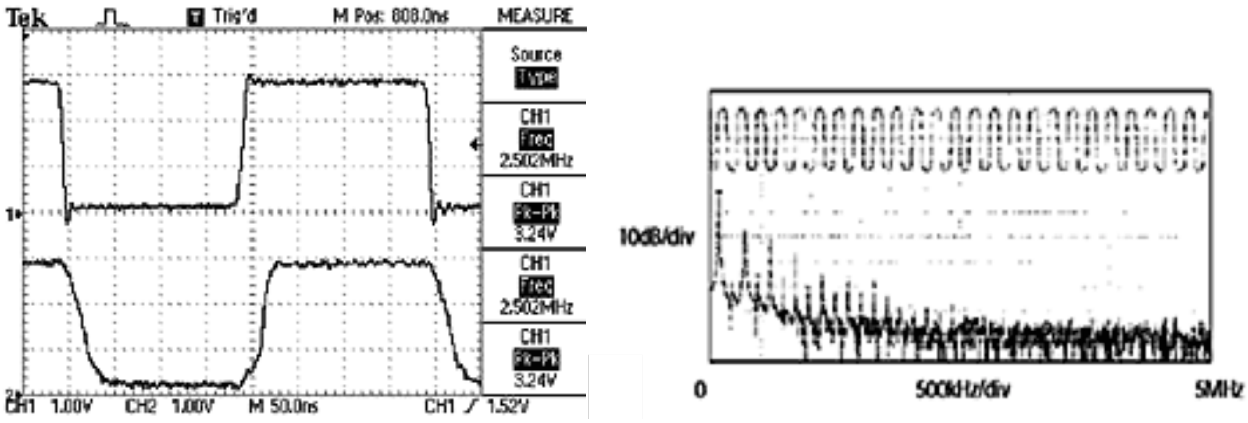


Figure : Plots for a 125kHz (250kbaud) test signal, with an appropriate limiting slew-rate.
(left) driver input (TTL) and non-inverted output waveform
(right) driver output waveform and corresponding frequency domain signal

Slew-rate limiting at the transmitter works by slowing the edges of the EIA-485 signal down and therefore reducing the signal's high-frequency components The Fourier transform of the slew-rate-limited signal shows that the frequency components above 2 MHz are virtually eliminated. An appropriate choice of driver/receiver circuit at the transmitter/receiver hence reduces radiated emissions and reduces susceptibility to noise and improper termination. This results in distortion of the signal (clearly visible in figure 3).

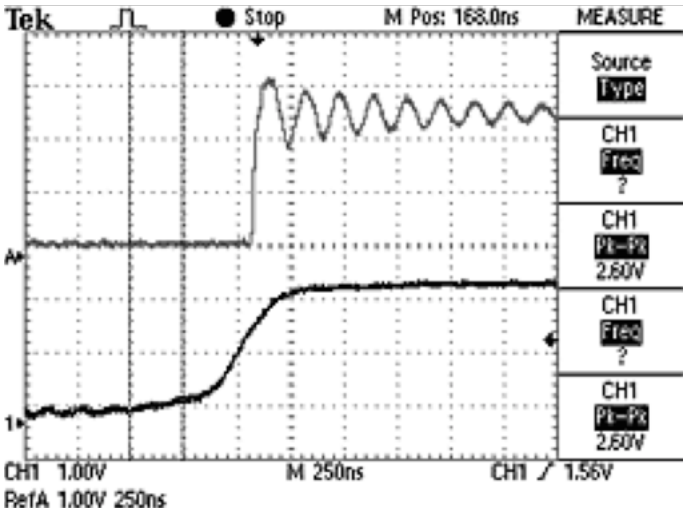


Figure : Time domain plots from an oscilloscope showing the start of a pulse sent with slew-rate limiting at 2.5 Mbaud (above) and one shaped to 250 kbaud (below).

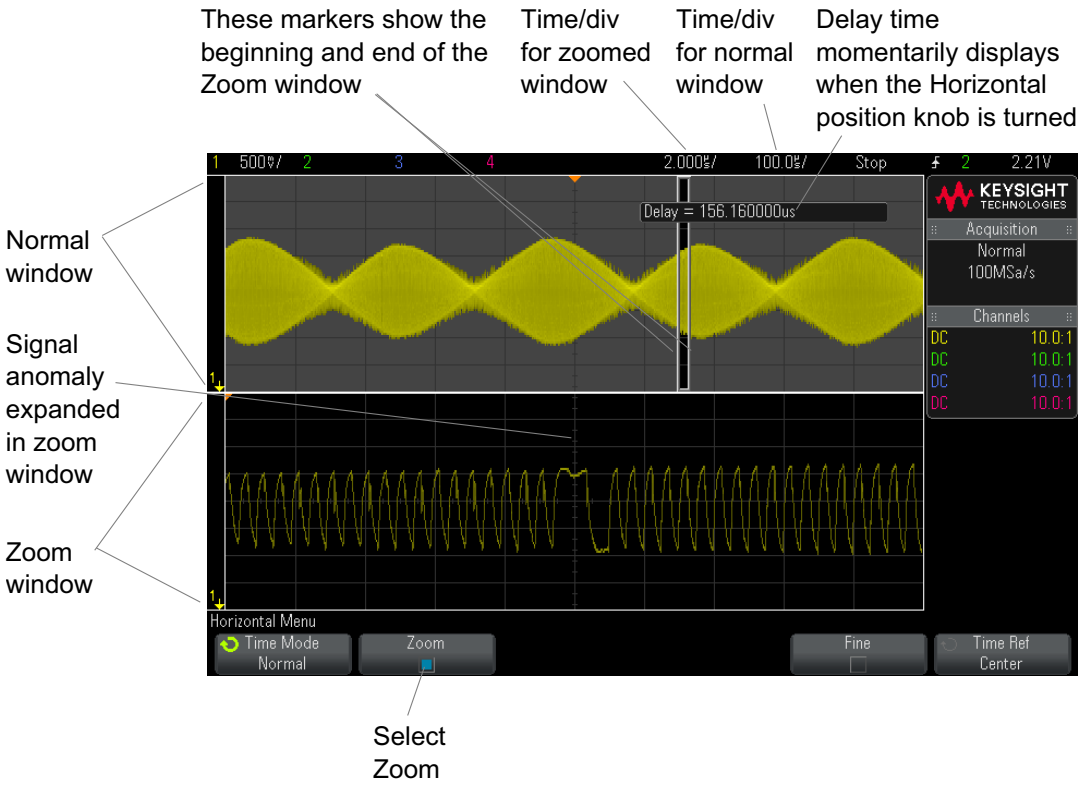
A driver circuit normally includes both a transmitter and a receiver. A receiver circuit that supports a higher baud rate will be designed to accept a wider bandwidth input signal, this makes the receiver more susceptible to interference at higher frequencies and more vulnerable to noise.

Tip: Selecting an appropriate driver chip for your design therefore really does impact the performance of a serial communications system!

Tip: Using the zoomed time base

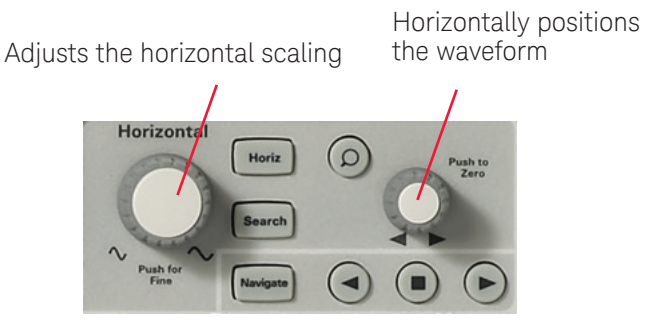
You can use Zoom button to locate and horizontally expand part of the normal window for a more detailed (higher-resolution) analysis of signals. This makes it really easy to find a particular part of a complex signal. You should try this to magnify the start of a signal transition. In the next lab this can be used again, e.g., to locate the break signal before the start of a DMX frame.

The Zoom button displays a horizontally expanded version of the normal display. When selected, the display is divided in half. The top half of the display shows the normal time/div window and the bottom displays a magnified portion of the time/div window.



The area of the normal display that is expanded is outlined with a box and the rest of the normal display is ghosted. The box shows the portion of the display that is expanded in the lower display.

This can be used with a live or stopped display. The display is easiest to read if you press STOP to capture the waveform, and then use this to view the stored waveform.



The horizontal position knob (left knob)) sets the left-to-right position of the zoomed display.

The horizontal scale knob (right knob) sets the time/div. As you turn the knob, the zoomed window time/div is highlighted in the status line above the waveform display area. The horizontal scale knob controls the size of the zoomed box.

Activity C: Information: Useful tips on triggering using a Keysight scope

Two tips on advanced triggering are part of a series of tips available on line to illustrate how to best use a professional scope. See also notes on the next page describing the steps to decode a serial bus.

TIP 5

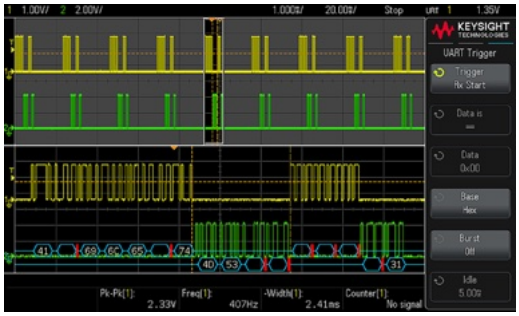
See More Detail Using Advanced Triggering

In TIP 1 we talked about basic triggering, but there are many more triggering options to choose from.

Rise/fall time trigger	Helpful if there's an impedance mismatch or loading on your system that's causing your edges to be too slow.
Setup and hold time trigger	Usually used to trigger on setup and hold timing violations.
Built-in protocol triggers	Extremely useful if you are working with serial buses.

Rise/fall time trigger

The rise/fall time trigger looks for a rising or falling edge transition from one level to another level in greater than or less than a certain amount of time. It triggers on signals that change state either too fast or too slow. This trigger is helpful to see if there's an impedance mismatch or some extra loading on your system that is causing your edges to be too slow.



Setup and hold time trigger

A setup and hold time trigger is used for any data and clock signal. One oscilloscope channel probes the clock signal and another channel probes the data signal. Setup time is the time a data signal level must be present before the clock edge. Hold time is the time a data signal level must remain after a clock edge.

This is an important trigger because digital designs require that the data line's state be setup (0 or 1) for a certain amount of time before the clock edge occurs. Set the trigger conditions to your specified setup and hold requirements to check for violations in your design.

Protocol triggers

Many oscilloscopes today have built-in protocol triggers. These are extremely useful if you are working with serial buses. For each of these different buses, there is a series of different triggers (Start condition, Stop condition, Missing Ack, Address with no Ack and more).

Aerospace/Defense	ARINC 429, MIL-STD 1553, etc.
Automotive	CAN, I²C, SPI, etc.
Computer	USB, etc.

You can begin your debugging by triggering on a start condition, which will give you a stable view of the packets coming through and insight into how your system is operating. If you're getting system errors or want to prove that everything's functional, you can even **trigger exclusively on errors**. This will allow you to focus only on the areas causing problems and not waste time wading through hundreds of error-free packets. If your oscilloscope has segmented memory, you can turn it on and exclusively capture errors over very long periods of time.

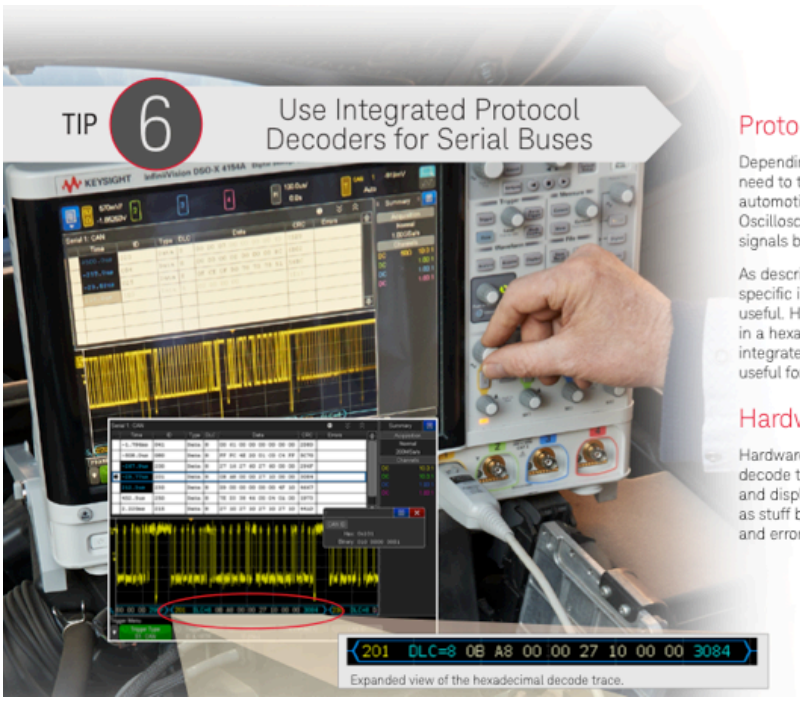
Learn more

Blog: Advanced Triggering

Webcast: Advanced Triggering (14:30)

TIP 6

Use Integrated Protocol Decoders for Serial Buses



Expanded view of the hexadecimal decode trace.

Protocol decode

Depending on what type of device you're testing, you might need to test certain serial buses (such as CAN and LIN for automotive and I²C and RS-232 for embedded designs). Oscilloscopes can characterize the analog quality of these signals by making physical layer measurements.

As described in Tip 5, a protocol trigger can help capture a specific instance or event on the bus, which is tremendously useful. However, many of the serial buses used today are encoded in a hexadecimal format and can be difficult to understand. An integrated protocol decoder translates those events into a more useful format.

Hardware-based decoding

Hardware-based decoding provides a real-time update of the decode trace. This enhances the scope's probability of capturing and displaying infrequent serial bus communication errors, such as stuff bit errors, form errors, acknowledge errors, CRC errors, and error frames.

Activity D: DMX Frame Transmission

The goal of this activity is to explore the DMX frame waveform using a programmable scope to decode the frame contents and to also trigger the waveform on the start of a DMX frame.

Equipment

1 Arduino-based DMX board with a 9V power supply
1 Oscilloscope with 2 scope leads, 1 BNC/BNC cable, 1 USB stick
1 1m Male to female screened XLR cable and Breakout cable (black, grey and white wires)
1 10 or 100 K Ohm potentiometer and a Male to Male jumper wire

Background

The Digital Multiplex (DMX) specification defines a multiplex format that is based on frames of bytes sent using asynchronous communications. Each frame is preceded by a Mark Time Between Frames (MTBF), with a high voltage level. This can be up to 1 second. The start of a frame is indicated by a Break of 88μS, or greater followed, by a Mark After Break (MAB) of 8μS or greater. A receiver uses the break to start reception of a series of DMX slots. The first slot contains a start code (byte) to inform the receiver what sort of data follows. Simple data uses a start code of zero. Other values denote other formats. The remaining frame payload can include up to 512 slots. You will interpret a DMX signal sent from an Arduino AT Mega micro controller programmed to generate a sequence of DMX frames. The Arduino software is configured using a DIP switch.

Equipment Configuration

Connect a DMX cable to the output of the board, and to a breakout cable (with black, white, and grey wires).

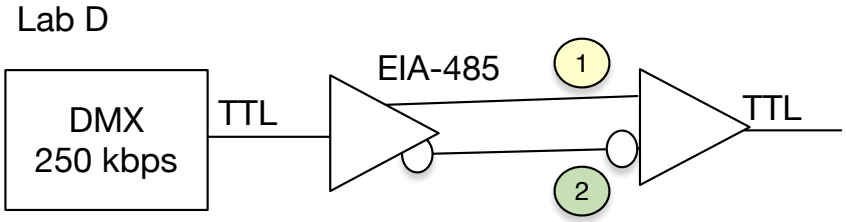


Figure : A way to observe the transmit output of the line driver connected to the DMX port.

When powered-on, the DMX controller will output a series of frames with preset contents onto the XLR connector for DMX bus (A,B).

Connect the channel 1 scope input to the A-line (white wire), channel 2 scope input to the B-line(grey wire). Connect the ground (black) wire to the scope probes to complete the circuit.

Use the correct probe setting for the setup of the probe that you use (1:1 or 1:10 input mode).

Later in the laboratory, you may wish to connect a BNC/BNC cable to the lower BNC connector on the board (using a 1:1 input mode). This connector can be used to display the TTL-level transmit signal from the Arduino, before it is sent to the line driver. (The upper connector shows the TTL-receive signal).

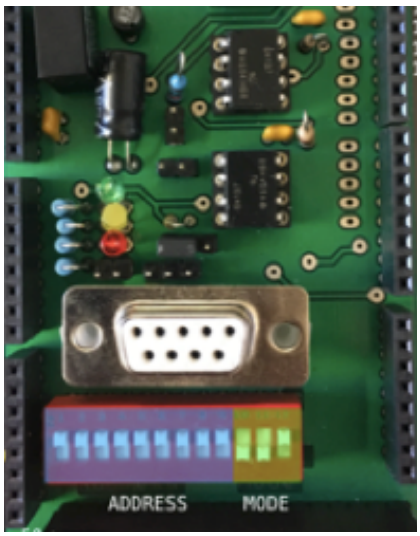


Figure: Arduino DMX Board

The board is powered by an external 9V dc power supply. There are a set of 12 DIP switches. These correspond to a 9-bit value and a 3-bit mode. This experiment will use two different DIP switch settings. To illustrate use of the switches, the figure shows mode 6, and address 0000 0000 110. The '1' position is when the switch is closest to the outside of the board.

The BNC output can be used to observe the TTL Rx and Tx signals using an oscilloscope. The voltages on the bus can be observed using an XLR breakout cable presenting signals on the black, grey and white wires. The 3.3V, GND and A3 input can connect an external potentiometer

D1 Setting the DMX Shield mode and observing g the output

On the Arduino DMX shield, find and set the DIP switch to binary 0000 0000 0 000 (see Figure). Power on the Arduino power supply.

Check that you understand the waveform presented on the scope:

- Look at the A-line, can you identify the Start of the Frame (the Break and the MAB)[5]?

Tip: It may help to use the start/stop button of a digital scope to freeze the waveform. Be sure to adjust the timebase so that you can see the required number of frames on the display.

Press the ***run*** button and observe the frozen scope display.

- Identify the break followed by the control slot with the start code. Use the X-cursors to measure the length of an entire frame, and then the frame rate[6].
- What is the ***length of*** the DMX Frame?
- What is the ***refresh rate*** of the DMX Frame?[7]

D2 Decoding the bus data

Setup the scope to decode a serial bus (see the additional information, for help, to do this for the DMX waveform). This configuration is different to that used for EIA-232!

*Tip: Set the **baud rate** to **user-defined** and then set the **rate** to 250 kbps.*

Observe the waveform again on the scope:

- Observe the ***mark after break (MAB)*** interval using the X cursors.
- Measure the length the period of the ***Break*** period using the X cursors.
- Check and record the TTL output waveform corresponding to the EIA-485 bus signal.

Tip: The break is now decoded in red, because the decoder interprets this as an illegal slot.

---

[5] The frame format is defined in the DMX Specification.

[6] Page 71 of the Recommended Practice for DMX512 describes the refresh rate of a DMX Frame.

D3: Triggering on the waveform

Can you work out how to set a trigger value to start the scope display at the start of each frame? To do this, the scope needs to be programmed to trigger on the break that preceded each frame:

To setup the trigger function, press ***trigger*** button.
Select ***trigger type*** – set using the rotary knob.
Select ***Pulse Width***.
Select ***>80 microseconds***. (use the up/down control to adjust this, if needed).

Tip: Additional information about other trigger modes is supplied in the Keysight tips (Tip 5).

Identify the control slot (the first slot after the break):

- Record the value of the ***start code*** (in hex), the slot following each break.
- Also record the value of the following slot in hexadecimal (the first data slot).
- Save the results to the USB stick for future use.

D4 Determine the values of specific slots in a DMX frame

Now that the scope is triggering based on the *break*, you can easily change the time base to see multiple slots and determine the position (address) of each slot within the frame.

- What are the four values in slots 5,6,7 and 8 of this DMX frame? (Record these 4 waveforms.)

*Tip: Adjust the **timebase**, and offset the **trigger point** in time, to see the required number of slots.*

*Tip: If the scope does not trigger, you may need to also set the **voltage level of the trigger signal** to 50% of the waveform. This level is shown as a "T" in the voltage display on the right.*

D5: Controlling the DMX value

This part of the activity makes a simple DMX control surface by connecting a potentiometer.

- Turn off the power supply to the microcontroller.
- Connect the black wire of the potentiometer to ground (GND) on the microcontroller.
- Connect the red wire to 3.3V on the microcontroller. The figure identifies the power pins!
- Check your wiring is correct. When you are sure, turn-on the power supply.
- Use the scope to measure the voltage at the centre pin of the potentiometer (green) as you rotate the potentiometer spindle.

Change the DIP switch to control the output value of a specific slot in each DMX frame, by setting the DIP switch to a non-zero value: 1000 0000 0 000 in binary. The A3 analogue input to the Arduino is used to set the value of the slot number selected by the DIP switch (in this case, slot 1). Although you do not need to see this, the following line of C code outputs this value on the DMX bus: "DMXSerial.write(dmxAddress,(uint16_t)analogRead(A3)/4)", and the green status turns on.

After checking the voltage, now connect the centre pin of a potentiometer (green) to the A3 analogue input. Vary the potentiometer setting and observe the DMX waveform.

Save a series of slot values to a USB stick:

- Check the output on the yellow LED (it should light when the slot value is >128, 0x80).
- Record the voltage at the centre pin of the potentiometer (the A3 analogue input).
- Record the resulting slot value using the Scope and the corresponding waveform.
- Try setting a different base address e.g. setting the DIP switch in binary to 1100 0000 0 000.

Activity D Information: Useful tips on triggering using a Keysight scope

Two tips on advanced triggering are part of a series of tips available on line to illustrate how to best use a professional scope. See also notes on the next page describing the steps to decode a serial bus.

TIP 5

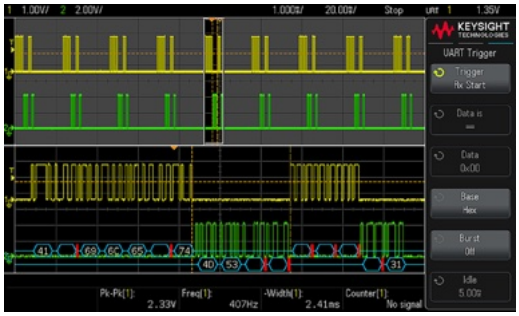
See More Detail Using Advanced Triggering

In TIP 1 we talked about basic triggering, but there are many more triggering options to choose from.

Rise/fall time trigger	Helpful if there's an impedance mismatch or loading on your system that's causing your edges to be too slow.
Setup and hold time trigger	Usually used to trigger on setup and hold timing violations.
Built-in protocol triggers	Extremely useful if you are working with serial buses.

Rise/fall time trigger

The rise/fall time trigger looks for a rising or falling edge transition from one level to another level in greater than or less than a certain amount of time. It triggers on signals that change state either too fast or too slow. This trigger is helpful to see if there's an impedance mismatch or some extra loading on your system that is causing your edges to be too slow.



Setup and hold time trigger

A setup and hold time trigger is used for any data and clock signal. One oscilloscope channel probes the clock signal and another channel probes the data signal. Setup time is the time a data signal level must be present before the clock edge. Hold time is the time a data signal level must remain after a clock edge.

This is an important trigger because digital designs require that the data line's state be setup (0 or 1) for a certain amount of time before the clock edge occurs. Set the trigger conditions to your specified setup and hold requirements to check for violations in your design.

Protocol triggers

Many oscilloscopes today have built-in protocol triggers. These are extremely useful if you are working with serial buses. For each of these different buses, there is a series of different triggers (Start condition, Stop condition, Missing Ack, Address with no Ack and more).

Aerospace/Defense	ARINC 429, MIL-STD 1553, etc.
Automotive	CAN, I²C, SPI, etc.
Computer	USB, etc.

You can begin your debugging by triggering on a start condition, which will give you a stable view of the packets coming through and insight into how your system is operating. If you're getting system errors or want to prove that everything's functional, you can even **trigger exclusively on errors**. This will allow you to focus only on the areas causing problems and not waste time wading through hundreds of error-free packets. If your oscilloscope has segmented memory, you can turn it on and exclusively capture errors over very long periods of time.

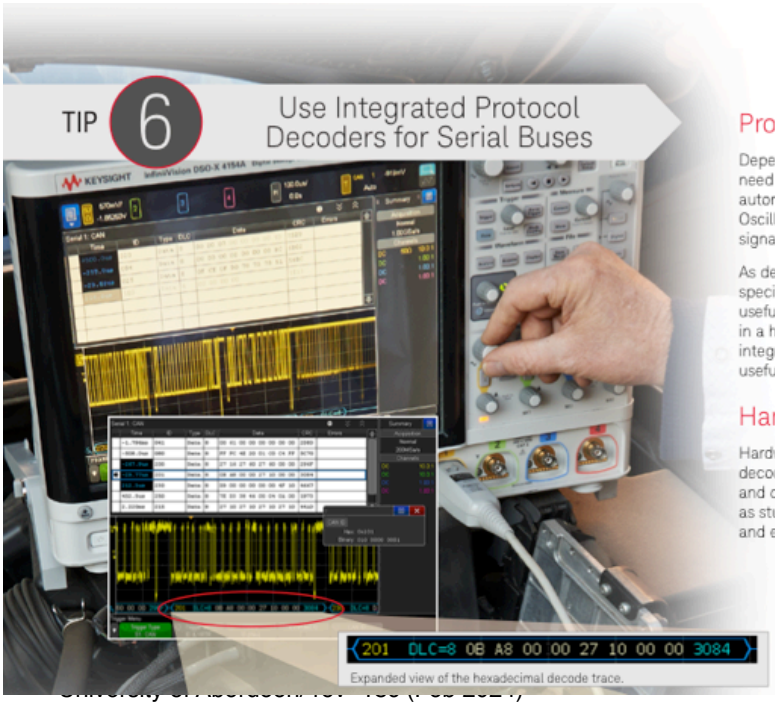
Learn more

Blog: Advanced Triggering

Webcast: Advanced Triggering (14:30)

TIP 6

Use Integrated Protocol Decoders for Serial Buses



Protocol decode

Depending on what type of device you're testing, you might need to test certain serial buses (such as CAN and LIN for automotive and I²C and RS-232 for embedded designs). Oscilloscopes can characterize the analog quality of these signals by making physical layer measurements.

As described in Tip 5, a protocol trigger can help capture a specific instance or event on the bus, which is tremendously useful. However, many of the serial buses used today are encoded in a hexadecimal format and can be difficult to understand. An integrated protocol decoder translates those events into a more useful format.

Hardware-based decoding

Hardware-based decoding provides a real-time update of the decode trace. This enhances the scope's probability of capturing and displaying infrequent serial bus communication errors, such as stuff bit errors, form errors, acknowledge errors, CRC errors, and error frames.

Activity E: DMX Control

The goal of this activity is to generate signals that could be used to remotely control a range of equipment.

Equipment

DMX Control Surface(s) - for the lab with a DMX Splitter if required.
DMX XLR lead (1 each)
Keysight scope and leads to crocodile clip
USB Stick to record waveforms in PNG format
Arduino-based DMX Receiver (1 each)
Breadboard (1 each)
D-9 to 3 wire connector (1 each) - For Digital control
Multimeter (1 each) -
LEDs (2 each) - For PWM control
1k Ohm (or larger) resistor (2 each) - For PWM control

Background

The control bus is controlled by a computer, or a control surface that generates the DMX frame. Sometimes a splitter or merger is used to regenerate the signal from one bus to another. Equipment is connected by plugging XLR cables between the connectors.

The equipment for the activity may have been setup for you before you enter the lab, but check that you understand the configuration:

- Do not power equipment yet.
- Connect the (male) XLR connector of the DMX cable to the output (female) connector of a control surface or the output of the another receiver.
- Connect equipment to the bus as needed
- A terminator is placed in the output (female) connector of the device positioned at the end of the cable bus.
- If a DMX splitter is used then there will be multiple cable buses - one for each lab bench.

Tip: Each student will be allocated a set of DMX control channels on the control surface . Each channel has a slider to set the value or the corresponding DMX slot. There is often a button below the slider that can be used to "flash" the channel to a preset value.

Tip: Students need to coordinate so that one person does not change the value of DMX slots using by others!

Activity E Information: University of Aberdeen Control Shield

The controller hardware uses an Arduino Mega based on the ATmega2560 microcontroller and a purpose-built DMX shield. On the shield a line driver (top) connects to the DMX bus. A row of three opto-isolators provide DC isolation for the bus and local power supplies, and a dc-dc convertor supplies the power needed for the line driver.

The mode of operation of the DMX software is configured using a dual-in-line package (DIP) switch. Inputs float to a logical “1” via an internal “pull-up" resistor to the positive supply. Each switch is wired so that when switched it connects the input to ground or logic “0”.

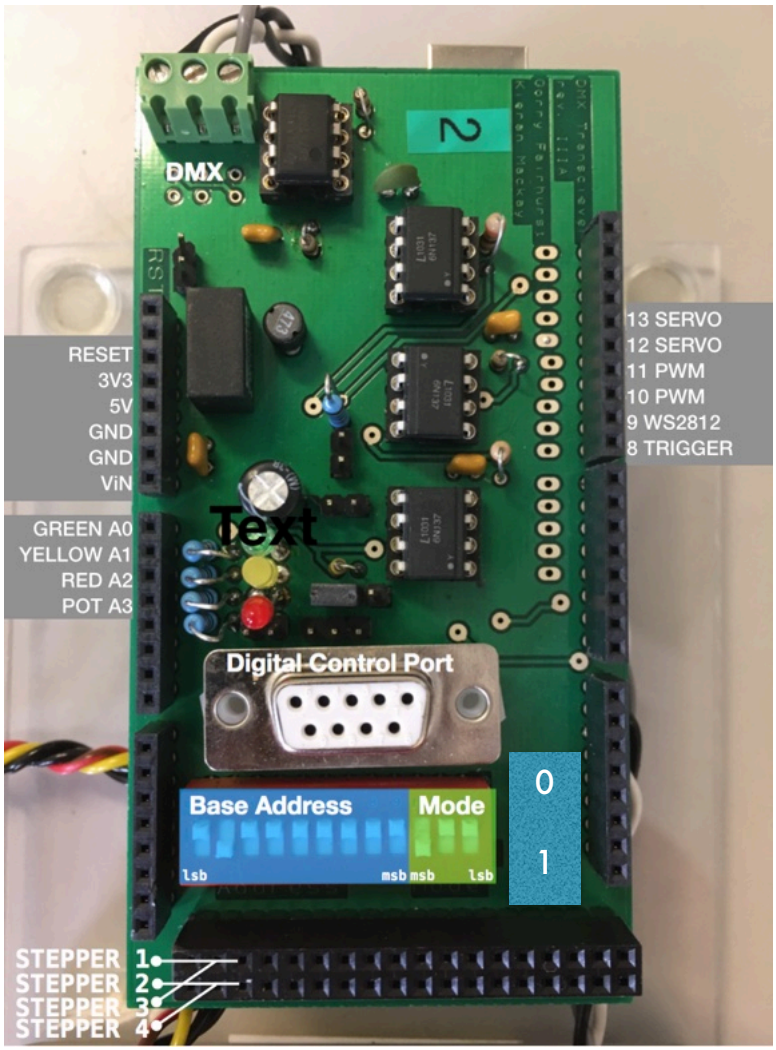


Figure: Location of DIP switch and output pins on the headers. The address switches are shown in blue (lsb first, in this case showing the setting for the address of 2 .A zero is represented as upwards in the figure).

For this board, the mode configuration uses the last 3 bits of the 12 bit DIP switch. The three mode switches shown in green (msb first, in this case showing the value 4, with a zero also represented as upwards).

R	R	R	G	G	G	B	B

Figure: Mode 1: Format of a Compressed DMX Pixel Control Byte for the Neopixel WS28XX

Slot 1	Set the rotational speed.
Slot 2	Control direction of rotation: Enabled when >0;
Slot 3	Enable Index mode >50; Set index reference when 255
Slot 4	Set index angle

Figure: Mode 5: Profile for DMX Control of the Stepper Motor Mode

Activity E Information: Setting the DMX Base Address using DIP Switches

The DIP switch method usually uses a 10-way set of microswitches. Each switch is wired so that when it is "on", it connects an input to ground (inputs to microcontollers normally float to a logical "1" when they are not connected, although for some inputs it is preferable to add a high-value resistor to the positive supply to ensure predictable operation). Since a DMX address requires 9 bits (each digit corresponds to the binary weights:1 2 4 8 16 32 64 128 256).

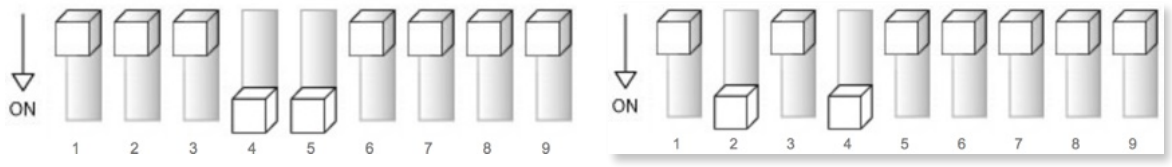


Figure: Left: DIP switch setting for an address of 10 (2+8=10).
Right: DIP switch setting for 24 (16+8=24).

The least significant bit is usually on the left side, hence switch 1 has a value 1, switch 2 has a value 2, switch 3 has a value 4, switch 4 a value 8, etc.

Tip: Many people find binary manipulation hard to visualise, and some helpful people have built applets for phones, iPods and computers to help people get to grips with this bit ordering.

Activity E Information: Setting the Mode using DIP Switches

For this board, configuration uses the last 3 bits of a 12 bit DIP switch. This can be configured to use one of eight modes (values are shown in the table below, most significant but first).

0 (000) DMX output (Activity C)
1 (001) Neopixel WS2812 (Activity F)
2 (010) PWM control (Activity E)
3 (011) Reserved for future use.
4 (100) Digital control (Activity E)
5 (101) Stepper motor (Activity G)
6 (110) Servo control (Activity H)
7 (111) Long Neopixel WS2812 (not used)

3-bit Mode values (shown in msb first order)

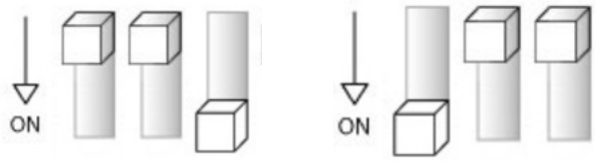


Figure: Left: DIP switch setting for a mode of 1 (Neopixel driver)
Right: DIP switch setting for a mode of 1 (Digital control)

Experiment

This lab requires you to use a set of two DMX slots. You may need to coordinate with colleagues, when you are sharing a controller. It will make triggering easier on the scope to view the DMX waveform if a demonstrator allocates slots 1 and 2 to a well-known value, such as 0x55.

In this activity you will use the Digital Multiplex (DMX) serial control bus to understand:

- DMX Digital Control – Control of a digital output from the received DMX frame.
- DMX PWM Control – Output of a PWM signal to control the illumination of a LED.

E1: Digital Control

This activity will use a microcontroller to decode the DMX signal and generate a digital signal. This signal is output on the digital control port (the D-9 Connector).

Do not power the board yet. You should avoid the possibility of transmitting a signal on the bus, which would otherwise interfere with the signal from the control surface.

Set the DIP address to the address of the first slot assigned to you, and the mode switches to 4 (100 in msb binary, where the last 0 is furthest from the board edge). This enables digital control. (For example, if you have address 3 and mode 4, then switches should be set: 1100 0000 0 100).

Check again the setting of the last 3 bits of the DIP Switch have a value of 4.

- When correctly configured, power the DMX receiver from a 9V power supply.
- The ***red LED on the board should flash*** to show correct decoding of the DMX frames.

*Tip: If the **red LED is lit continuously** it indicates that no DMX signal is received. **Turn-off the power to the board**. You may be jamming the DMX bus and preventing others from receiving a signal. Now check the setup and the mode setting of the DIP switch before restoring power. If the **red LED is not lit,** this may indicate that the board is configured to send a signal to the bus.*

*Tip: If the Red **and Yellow LEDs flash randomly**, this indicates that the received DMX signal is inconsistent. Someone else may be jamming the bus. Check the DIP switch on the other receivers.*

E2: Digital Control

The yellow LED lights when the value of the addressed DMX slot is greater than zero. Check that this digital output can controlled by the appropriate slot sent by the DMX control surface, by setting the slot to 0 or 255.

- ***Connect a cable to the top BNC output*** to monitor the received frames using a scope.
- Program the serial decode function of the scope to decode each DMX frame (see instructions in the information sheet for this activity).
- Save the waveform for the DMX slot values and record the corresponding decode.
- ***Connect the D-9 Connector*** and use a ***multimeter*** to measure the voltage present on each output when an addressed slot have a value of 0 and when it has a value of 255.

Tip: Enabling a serial bus decoder allows a scope to trigger on a specific DMX slot value sent on the bus. See instructions for scope serial decoding.

Tip: You also need to trigger the scope at the start of each DMX frame, by setting the scope to trigger on a signal that is low for more than 80 microsecs, corresponding to a break signal.

E3: Pulse Width Modulated Control

This activity will use a microcontroller to decode the DMX signal and generate a Pulse Width Modulated (PWM) signal, with a pulse repetition frequency that is suitable to control the brightness of a LED. This PWM signal is generated using a Arduino library that uses PWM hardware in the AVR to generate an output signal with a duty cycle that is now controlled by the value in the selected DMX slot.

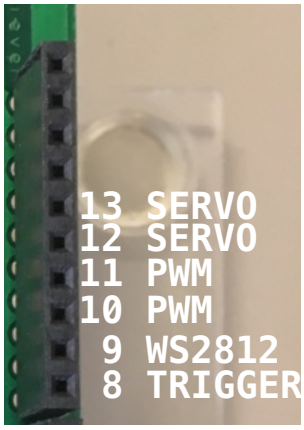


Figure: Position of the PWM output signals.

The PWM mode is enabled by setting the last three right-most bits of the DIP Switch to a value of 2 (010 in binary). The board will now output two independent PWM signals on pins 10 and 11 of the shield header.

- The red status LED should flash if data is being received from DMX bus.
- Check that the yellow status LED is controlled by the value of the received DMX slot.
- Check that the green status LED is controlled by the value of the second DMX slot.

Identify the two output pins (marked PP on the board, the first corresponds to the value of the first DMX slot at your configured base address, the second corresponds to the value of the next consecutive slot).

- Connect the scope to the output pins and observe their output value.
- Connect a current-limiting resistor to the output (i.e., limiting current to 5mA)
- Connect an LED in series with this resistor, connecting between it and the ground.
- Check the output LED illumination is controlled by the value of a received DMX slot.
- What is the pulse repetition frequency for this waveform?

Tip: You could measure the time period of a cycle using the scope cursors. The measure function on the scope can be used to directly display the pulse repetition frequency.

- Vary the DMX slot value – how does the waveform change with the value sent by the control surface?
- Observe the corresponding PWM output for various DMX slot values. Now record a set of waveforms showing the DMX slot value and the corresponding PWM waveform.
- Ensure you capture the results for a DMX slot with a value of zero, a decimal slot value of 100 and a decimal slot value of 128.

Activity F: TRIAC AC Power Control

This lab activity will use the Digital Multiplex (DMX) serial control bus. This will help you to understand:

- DMX Addressing - Control of receiver using a base address [8] and set of consecutive slot values. You will also use receivers configured with a base address such that more than one receiver is controlled by the same sets of slots within the DMX device that uses four channels.
- DMX Power Control - You will lean about phase-control of AC power and its use to control the brightness of an incandescent lamp.
- The equipment uses either a 3-pin or 5-pin XLR connector.

Equipment

Shared between groups of students:
1 DMX control surface and multi-port DMX repeater
For each group of 8 students:
Milford Instruments DMX Receiver (generating 8 TRIAC control signals)
5-pin XLR cable and terminator (for Milford Instruments receiver)
8-way BNC breakout (from DB9 output of Milford Instruments receiver)
For each group of 4 students:
Lengths of 3-pin DMX cable to connect equipment, and terminators
1 x 4 Channel DMX Dimmer Pack (with Bulgin power output)
1 x Short 4 circuit Bulgin power cable
1 x 4 Channel current probe (BNC outlet, Mains powered)
1 x Long 4 circuit Bulgin power cable
4 x PAR-56 lamp or equivalent (1 for each student)
For each student:
Keysight
USB Stick to record waveforms in PNG format
1 x BNC to croc cable
2 x BNC/BNC cables (Ensure each scope input is set to 1:1)

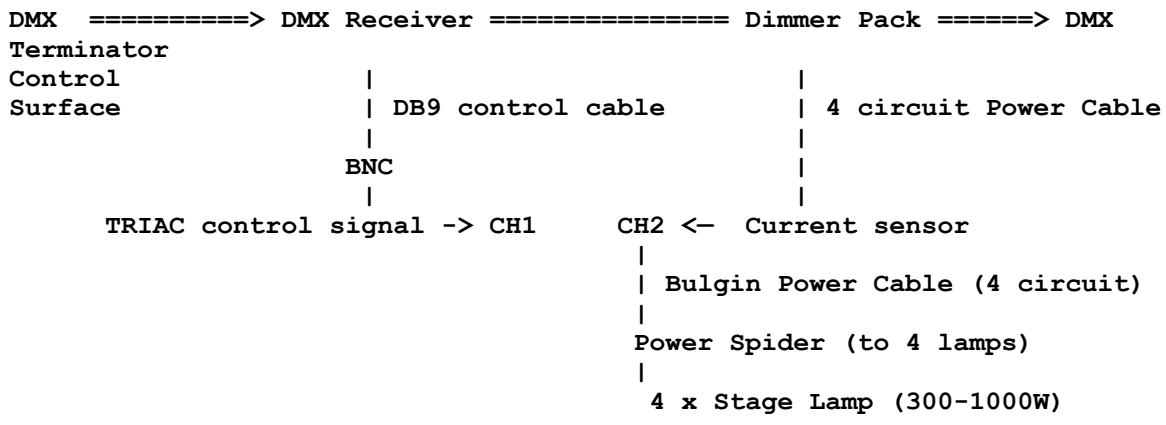


Figure: Connections to the TRIAC controller and AC Dimmer Pack

[8] Pages 53-56 of the Recommended Practice for DMX512 describes the configuration of a base address.

Background

The bus is controlled by a computer, or a control surface that generates the DMX frame. A control surface is used to output a frame to the DMX bus. A set of sliders assigns a value between 0 and 255 to each channel, and hence each slots within the frame. A set of DMX receivers are connected to this bus. Often a splitter or merger is used to regenerate the signal from one cable bus to another. Equipment is connected by plugging XLR cables between the connectors.

The bus will be setup in this way:

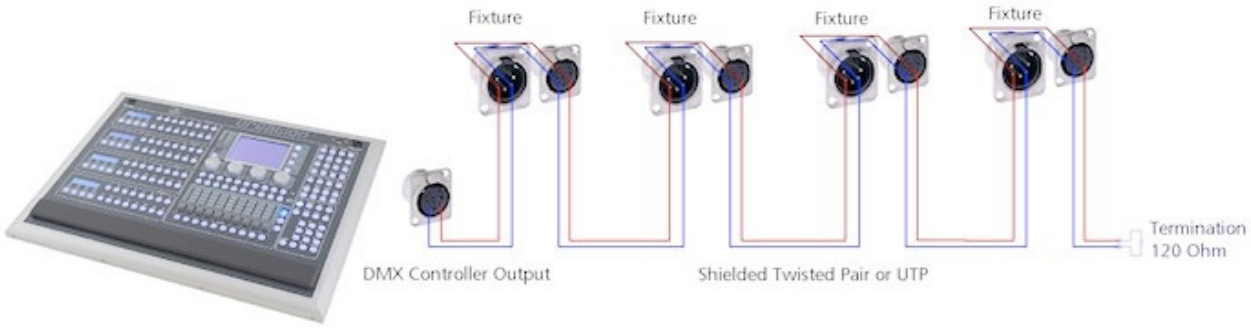


Figure: Connection of Equipment to the DMX Control Bus

- Connect a DMX cable from the output connector of the control surface or the first receiver.
- Connect the other receivers in series, one after another.
- A terminator is placed in the output (female) connector at the end of each cable bus.

Each DMX receiver (e.g. TRIAC Dimmer pack) is configured with a base address. This can be represented by a 9-bit binary number. There are three common ways to configure the base address:

1. The most common way is to use a Dual In Package (DIP) switch, where each switch corresponds to one bit of the address byte, reading left to right (i.e., least significant bit first).
2. Devices that require more frequent configuration can store the address in flash memory and show it using a LED/LCD display. The address is set using up and down switch inputs.
3. Remote Device Management (RDM) allows the address to be set remotely over the bu[9].

The base address determines the first slot within the DMX frame used by the receiver. The receivers in this lab use 4 or 8 consecutive slots. Two receivers with the same address use exactly the same sets of slots. This simultaneously controls two receivers with the same set of values. One could be a dimmer pack, and a Milford Instruments receiver could observe the TRIAC firing signal.

Here are some examples:

1. A device that uses eight channels and has a base address of 1 will interpret the slots 1, 2, 3, 4,5,6,7, and 8 to control its output values.
2. A device that uses four channels and has a base address of 5 will ignore (skip) the first 4 slots (1-4) and will receive the slots 5, 6, 7, and 8.
3. A 4-channel receiver with a base address of 100, reads data from slots numbered 100, 101, 102, and 103. The next receiver could then be configured to use a start address of 104.

To minimise the need to share equipment the bus system may be setup with one controller for each group of 4 or 8 outputs, check the current setup in the lab to determine your own setup.

---

[9] The Recommended Practice for DMX512 describes the use of RDM and this is specified in detail in the ANSI E1.20 specification.

Experiment

This activity will use receiver equipment that is connected to a DMX bus to understand how a TRIAC can remotely control a single AC circuit.

A dimmer pack is connected to the DMX bus and is assigned a base address (1-512). The base address of the receiver associates the control channel with one of the power circuits controlled by a Dimmer Pack. The receiver uses a microcontroller, connected to a set of TRIAC circuits. Each decoded slot is used to control the power of one lighting circuit. A stage light will be connected to each output of the dimmer pack.

F1: Controlling the current in a lighting circuit

In this experiment, an ACS712 Hall Effect current sensor board will be placed in series with the lighting circuit, to monitor the current in the power circuit. This sensor costs less than £10. This can be used to view the TRIAC ***output waveform*** on a channel.

First, identify the DMX base address of your four channel Dimmer Pack (by reading the DIP switch setting or checking the LED display). Each student must choose one of the four output channels from the four supported by their Dimmer Pack.

You can now calculate the DMX slot that is being used to control your output. By calculating the slot number used (Base_address+channel-1).

- Find the channel corresponding to your assigned slot on the control desk.
- Use the channel fader slider to control the level of your DMX slot within the DMX frame.
- As the percentage value changes, observe the output level of the light that you are controlling.
- Some values cause the lamps to "sing"! Can you work out why this happens? To prevent singing, professional equipment often uses a snubber circuit to reduce the slew-rate of the output waveform. A rise times of around 300μS. can significantly reduce interference to other equipment.

Tip: Use the short BNC cable to connect to the current sensor - you may need a longer for part D2.

F2: Viewing the control signal for the lighting circuit

A second set of receiver is also connected to the DMX bus. This is an 8 slot DMX receiver produced by Millford Instruments in the UK, and hosted in a steel box. It uses a PIC microcontroller to output 8 independent PWM /phase control signals (each controlled by one slot in the DMX frame). The PWM ***control signal*** is a aligned to the zero-crossing of the mains AC cycle. and is output on a BNC socket, labelled 1 to 8.

Identify the BNC socket corresponding to the value for your DMX slot. Connect a BNC cable to see the signal on channel 1 of the scope. This signal is a TRIAC control waveform that changes with the value of the DMX slot. This could be used to drive a TRIAC to build a dimmer.

In this experiment, this trigger signal output is connected to an oscilloscope.

- Gradually increase the DMX value of your slot and note the control waveform;
- Connect the output of your current sensor to channel 2 of the scope to observe the current flowing through the lamp.
- What happens as you move the slider to the TRIAC control signal and the current flowing through the circuit? Does the lamp conduct at a low trigger value? (If other students are nearby, see if this value is different for a different type of lamp filament.)

Tip: The receiver control signal is aligned to the zero-crossing point of the AC mains supply.

F3: Recording the signals

The control desk will display the level for each fader.

- Record the waveforms for a DMX slot value of 0, 64, 128 and 255. Pay particular attention to record the horizontal scale. Is the output linear? - i.e. pulse width proportional to the slot value?

Tip: The vertical axis for the current sensor needs to be multiplied by the appropriate ratio to determine the actual current flowing through the lighting sensor. Information to do this is provided in the current sensor data sheet for the ACS712 Hall-effect current sensor[10].

Tip: There are three design of TRIAC dimmer packets in use in the lab. It would be good to compare your waveforms with those of students using other designs equipment - is it the same?

This lamp presents a low inductance to the TRIAC, i.e. the current and voltage are in phase. What would happen if an inductive load were dimmed in this way? Why would this not be a good idea?

[10] A Hall Effect-linear current sensor, https://www.sparkfun.com/datasheets/BreakoutBoards/0712.pdf

Activity F Information: Software Dimmer Control

A Dimmer Pack can use a simple microcontroller to generate the TRIAC gate signal, e.g:

1. Convert the light value to a software loop count value
   (This may imply mapping the value using a lookup table to set a particular curve[11]).
2. Wait for a zero crossing event to be detected (when the AC supply crosses zero).
3. Use a software loop to wait for the required time (or initialise a hardware timer).
4. The mapping between the slot value and pulse width could utilise a transfer function. The rationale between choosing an appropriate transfer function, or "dimmer curve" to fire the track is described here: https://www.etcconnect.com/Support/Articles/Dimmer-Curves.aspx
5. On completion of the required period, a pulse is sent to trigger the TRIAC gate.

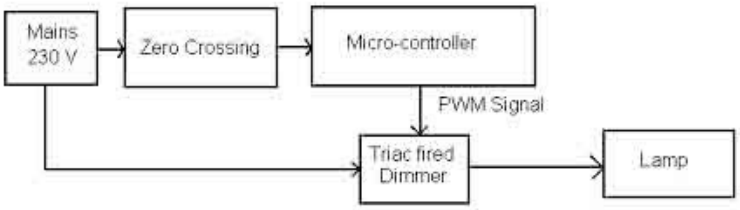


Figure: Simple design of microcontroller DMX Dimmer

Care needs to be taken to avoid an inductive/capacitive load to a simple TRIAC circuit when the connected equipment contains a motor or transformer. An inductive load means the current lags behind the voltage, and therefore the current and voltage will not be in phase. When the load current falls below the TRIAC threshold level, the TRIAC will turn off.

A zero-crossing voltage detector would cause the microcontroller to send a trigger pulse to the TRIAC at the wrong time to retriever the TRIAC, resulting in erratic dimming behaviour. To reduce the effect of the current being out of phase, dimmer packs designed for transformer loads can use a "hard" firing technique (e.g., using a "pulse" capacitor). This ensures that the gate firing pulse is maintained for a long enough period of time to ensure that the current reaches the device's threshold level.

[11] The transfer function from DMX slot value to pulse position may not need be linear!

Activity G: TDM Output via DMX

The goal of this short activity is to use a time division multiplex (TDM) control bus.

Background

In this activity, DMX control bus carries input values to the receiver board where a profile of DMX slots are used to encode a value for a string of Neopixels. Before you proceed, make sure you read and understand the additional information on the Neopixel WS2812/WS2813.

Equipment

DMX Control Surface(s).
DMX Terminator (1 per DMX cable bus) and DMX XLR lead (1 each).
Oscilloscope and 2 scope probes.
USB Stick to record waveforms in PNG format.
Arduino-based DMX Receiver (1 each).
Tri-colour LED (1 strip each) - This LED strip has +5V, Ground and DI/signal wires.
Set of jumper wires (e.g.yellow, red, black).

Experiment

- Set the base address to match a unique channel on the control desk. (Use 1 for a small desk.)
- To enable control of the TDM bus, configure the mode using the three right-most bits of the DIP Switch with a value of 1 (001 in binary, to select Neopixel control).
- Ensure you bus is operating correctly with all boards attached. See troubleshooting instructions in lab C. A set of correctly configured receivers will pulse the red LED to flash on and off.

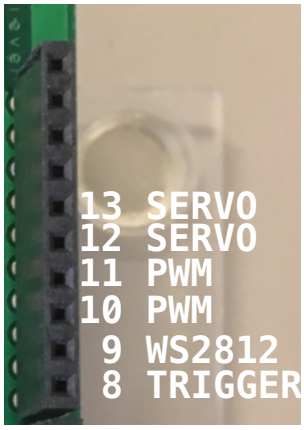


Figure: Position of the WS2812 output signal on pin 9.

This mode causes pin 9 of the shield to output a digital TDM control signal (labelled “W” on the connector).

- Find the Data Input control (DI) (also sometimes labeled Signal (SIG)) of the LED strip.
- Connect DI (usually a yellow wire) to the output from the receiver board provided on pin 9.
- Connect a +5V supply to the LED strip (usually a red wire).
- Connect the ground (usually a black wire on the LED strip) to the supply ground.
- The fourth remaining wire, if present, is not used.

G1: Sending DMX Data for a Pixel

The values sent to the control strip are controlled by a contiguous set of DMX slots stating with the value configured as the base address. Set the control surface to send the following values for a DMX slot:

- White (red, green and blue LEDs at full) sent when the DMX slot has a value of 0xFF
- Black (all LEDs off), sent when the DMX slot has a value of 0x00.

G2: Setting the Colour of a Pixel using DMX Data

Each slot carries encoded colour information for a single LED pixel in the strip. The value for the there LEDs is encoded into one single DMX slot in the following compact way:

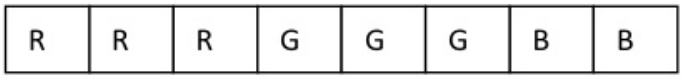


Figure: Format of a compressed DMX Pixel control byte for the Neopixel WS28XX bus

The three most significant bits are used to control the red LED intensity, three are used to control the green intensity and the two least significant bits are used to control the blue intensity. There are only two blue bits, because the human eye is less sensitive to blue light.

For example, a green light is produced when the control value is 0x1C, which is 28 in decimal. Expressed as a percentage of 255, this is about 11%. A Cyan colour is represented by 0x1E, which is 30 in decimal, corresponding to approximately 12%.

Tip: You will need to use a scope with decoding of your slot to determine accurately the slot value that is being output by the control surface, because usually the control surface output is only shown as an (approximate) percentage. Lab C contains notes about how to setup and use the serial decode function on the oscilloscope.

- Set pixels in the LED strip to a specific colour. You will find it useful to calculate and check the DMX slot values, because you will find it hard to guess the correct values simply by only moving sliders on the control surface. Try setting the pixel to produce a specific colour:

  Red, Blue, Orange, etc

- If you have more than one pixel, try setting the second pixel to produce a different colour, etc.
- Use a scope to observe the TDM burst to understand how the values are represented as they are fed to a string of Neopixel LEDs by attaching a probe to the DI input (yellow wire) of your LED strip. This signal carries control data to drive the string of LEDs, as explained next.
- Record the TDM bus output for a DMX input of 0, 255, and 128.

Activity G Information: The Neopixel WS28XX Time Division Multiplex Bus

The Neopixel Library generates a Time Division Multiplex (TDM) control signal to set the colour of a set of pixels. using an on-chip controller for the Red,Green, and Blue LEDs in each pixel.

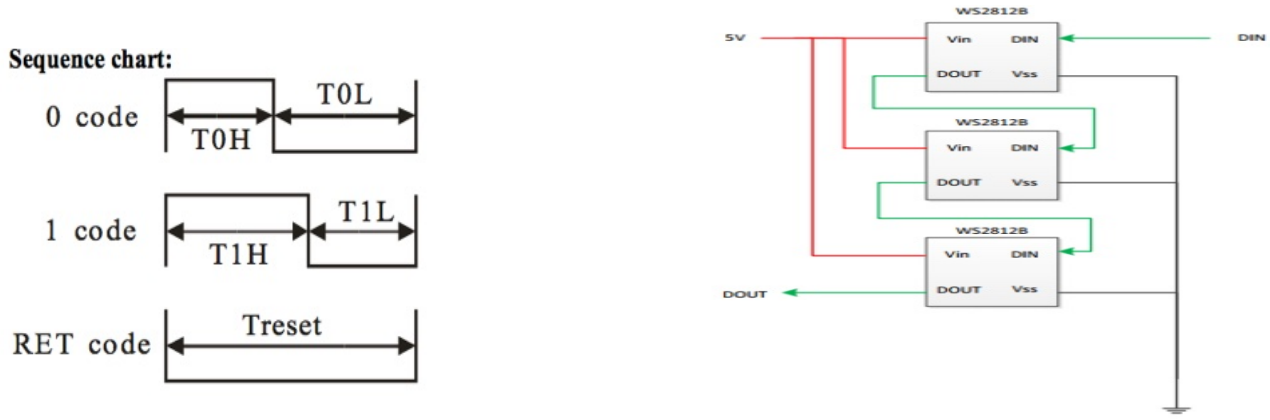


Figure: Encoding data using the Neopixel WS28XX TDM Bus

Data transfer time(TH+TL=1.25µs±600ns)

T0H	0 code ,high voltage time	0.35us	±150ns
T1H	1 code ,high voltage time	0.7us	±150ns
T0L	0 code , low voltage time	0.8us	±150ns
T1L	1 code ,low voltage time	0.6us	±150ns
RES	low voltage time	Above 50µs	

Figure: Bus Timing in the Neopixel WS28XX TDM Bus

Each LED pixel is connected to a common 5V and ground. The TDM signal is passed into the pixel chip via the Data Input (DI) pin (sometimes labeled SIG) and regenerates a reconditioned signal at the Data Output (DO) pin. The external connection to the DI pin uses a yellow wire. (Some LED strips use two input pins to provide robustness when a single pixel fails.)

The TDM bus transmits a control value for each pixel as three serialised 8-bit colour values. These 24 bits are sent contiguously (one after another) using pulse width encoding for 0 and 1.

Multiple pixels can be linked together. The DO output is connected to the next pixel's DI input. This passes the TDM signal through each pixel in series. The first 24 bits are removed within each pixel using a digital port data latch to store the value. After latching, all data passes through the pixel directly to the DO line (256 pixels could be controlled with the current software - but each receiver has been configured to drive 32 pixels). Pixels are available in a range of formats: a single tri-colour LED, strings of LEDs, and pre-assembled as strips, circles, etc.

Figure: String of Neopixel LEDs

A reset signal (greater than 50µs low fro the WS2812) indicates the start of a new frame. This resets all pixels, allowing a new value to be latched into the first pixel and then the next, etc - until all pixels have been set. Each TDM frame is followed by an idle period.

Activity H: Stepper Motor Control

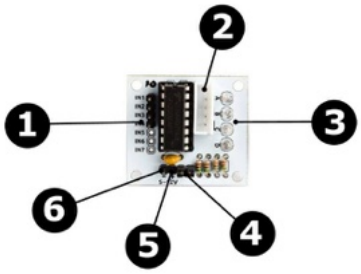
The goal of this activity is to control the rotation of a stepper motor using the DMX bus. Stepper motors are commonly used when the rotation speed needs to be controlled, or when it is required to rotate a shaft to a specific angle, e.g., in a robotic arm or actuator.

Equipment

DMX Control Surface(s) - DMX Terminator (1 per bus) and DMX XLR lead (1 each)
Oscilloscope and 2 scope probes.
Arduino-based DMX Receiver.
+5V power supply - to stepper controller.
Stepper motor and stepper motor controller board.
Red, Black and four other coloured jumper leads (each male to female).

Background

This activity uses mode 5 (101 in binary) to process a set of four slots that generate a control signal used by a driver to power a 5V DC four-phase stepper motor. The motor is driven using a ULN2003 driver chip with seven NPN Darlington Pairs (rated 500 mA). Each of the four motor outputs is protected with a common-cathode clamp diode allowing the driver to safety switch the inductive motor load. The driver chip is powered from an ***external*** 5V power supply.



1	Microcontroller I/O port connection
2	Stepper motor interface
3	Stepper motor driver LED display
5	+5V external power supply
6	Common ground

Figure: ULN2003 driver board

Each digital output controls the current to one of the stepper motor windings. Note that these signals are ground-referenced to the source of the signal (the Arduino).

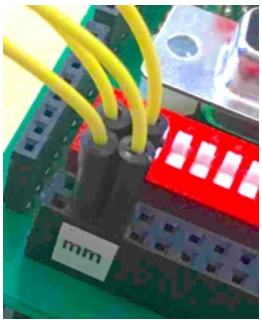


Figure: Position of stepper motor control pins. The order of connection to the driver inputs must be correct to control the rotation of the motor.

Position of the four 'm' motor pins (1-4) and corresponding Arduino pin numbers.

54 GND	52 (1)	50 (3)	48
55 GND	53 (2)	54 (4)	….

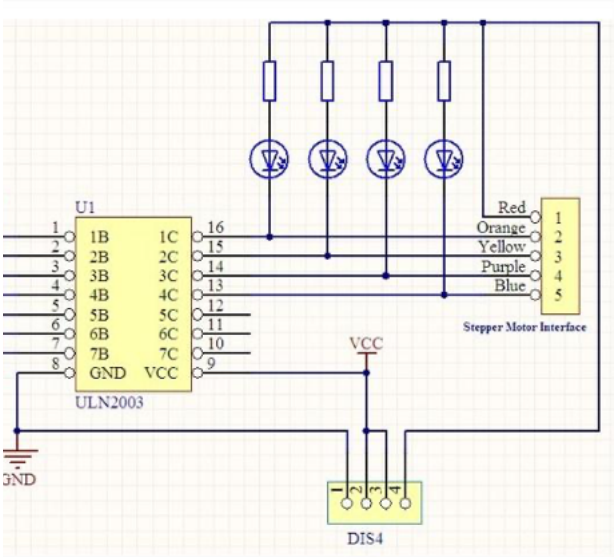
The stepper motor controller uses a profile consisting of four consecutive DMX. This profile starts at the base address configured by the DIP switches and provides the control data that selects the motor rotational speed or control in indexed mode (where the angle of rotation of the motor corresponds to the value in a specific DMX slot (also known as the index). This operation resembles commercial controllers.

Experiment

In this experiment, you first connect the driver and motor to the Arduino controller, then send DMX control frames to the Arduino to control the motor.

H1: Connecting the stepper Motor

Connect the four control outputs (50-53) to the four corresponding inputs of the motor driver. The outputs are provided on the dual-row connector at the left edge of the DMX board using pins 50-53. These are arranged in a 2x2 matrix, and must be connected in the pattern shown below. Connect jumper cables to each output in turn, starting with driver input 1 from Arduino pin 52, as shown below. Connect the driver ground connector to the Arduino ground signal and also to the ground of an external supply. Connect a 5V external power supply to power the driver. The power supply connector is at the bottom of the circuit in the figure below (DIS4), which includes a jumper between pins 3,4 that connects the motor power supply to the driver supply (see below).



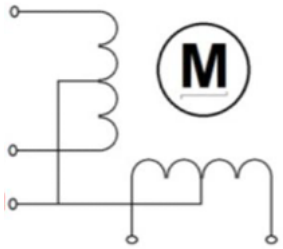


Figure: Motor windings (above) shopping four windings and one common (red) - connected to the stepper motor interface connector

Figure: Circuit diagram for UL2003 driver. Connection to the Arduino is on the left of this diagram (4 pins are used, one for each Arduino output)

Set the DIP switch for the DMX receiver: Set the DMX base address and set the mode to 5.

Using the control surface, set the value of the four slots starting at the base address of your device to a value of zero. Connect the Arduino shield to the DMX bus. The red LED on the Arduino DMX board should flash to show the DMX signal is received. This shows reception of frames from the DMX bus. The yellow and green status LEDs should be off.

Tip: Make sure that nobody else uses your set of DMX slots. This mode utilises each DMX slot for a specific function. Operation will become erratic if someone changes one of your DMX slots!

H2: Controlling Rotation

The value of the slot at the base address (slot 1) controls the speed of rotation. Move this slider up, to set a non-zero value. Although you have set the speed, the motor should not yet move.

The next slider, at the base address+1 (slot 2) controls the actual rotation. Slot 2 can stop the motor (when slot 2 = 0), or rotate clockwise or anticlockwise, controlled by a non-zero value of slot 2. The yellow LED show light to show that motor rotation is enabled.

The motor spindle rotates in discrete step increments when pulses of current are applied to the windings in the correct sequence. Each of the four phase LEDs on the driver board indicate the status of a motor phase. Vary the value of slots 1 and 2, and check you understand the operation.

Connect a pair of scope channels to two of the stepper motor inputs. (You may find it easiest to connect channels 1 and 4, by sliding up the female end of the jumper cable to expose some of the pin and connecting a scope probe to this exposed pin). Observer the signal to the motor using the scope and the motor LEDs connected to the driver chip and the scope waveform. Configure the scope to trigger on one of the stepper motor inputs.

- What is the effect of changing the speed of rotation in slot 1 of the profile?
- What is the effect of changing the rotation control value in slot 2 of the profile?

So far, this has used just two of the four slows. The remaining slots in the profile are shown in the figure below. All four slots will need to be used for the next part.

Slot 1	Set the rotational speed, rotation enabled when >0
Slot 2	Enable rotation when >0; Rotate the opposite direction when value >=128
Slot 3	Index mode disabled < 50; Indexing enabling = 50-200; No movement when = 201-254; Set index reference = 255
Slot 4	Set the index angle (0 to 360 degrees)

Figure : The profile of four DMX control slots used for the stepper motor control

H3: Controlling Angular Position

The same circuit design can be used to select an angle of rotation, rather than a rotational speed. This is often called *indexing*. In this mode, a change in the value of slot 4 causes a reproducible angular movement relative to a set reference index position. The speed of rotation (slot 1) is still used to control the speed at which the motor rotates.

Before indexing can be used, it is necessary to set as reference angle against which the index value is measured.

Use the control surface to rotate the motor to your choice of reference angle:

- Set both slot 3 and slot 4 of the profile to zero.
- The green LED should be off, to show no reference has been set.
- Using the rotation control (Slots 1 and 2), rotate the motor (Slot 1 > 0, Slot 2 > 0).
- When at the required angle, set slot 2 to zero.
- The motor stops and the yellow LED should now be off
- Now record this angle as the rotation reference point, by momentarily setting slot 3 to 255.
- Return slot 3 to 50% to enable indexing mode.
- The green LED should remain on to show that the controller is in indexed mode.

Rotation is not enabled (therefore, the yellow LED is off).

- Set the speed of rotation to the maximum by setting slot 1 = 100%

A practical motor could be used to move a load with significant inertia, to rotate a large object or with gearing - in these applications the speed of movement might need to be controlled.

You can now move the motor in indexed mode:

- Enable rotation by setting slot 2 >0.
- The yellow LED should be on (showing rotation is enabled), but the motor will not yet move.
- The value of slot 4 now controls the indexed angle relative to the selected reference position.
- Adjust the value of slot 4 to control the angle relative to the set reference angle.

The motor is driven to the required angle that is set by slot 4, and the speed of this rotation is controlled by slot 2.

Examine how the signal to the motor windings that rotates the motor by observing the waveform as you select different angles of rotation. Observe also the effect of changing the speed (slot 1).

*Tip: Select a slow timebase to capture the sequence of signals activating the windings. Enable the scope's "**single shot**" mode to arm the trigger. This will store the waveform after the first change in the signal. You can scale the waveform on the scope display **after** it has been captured.*

H4: Controlling Relative Position

Set another index reference point:

- Rotate the motor to any required position.
- Disable rotation by setting slot 2 = 0 (the yellow LED will turn off).
- Set slot 4 to 50%, i.e. the index position is set to 180 degrees.
- The motor will not yet move.
- Set a new reference by momentarily setting slot 3 to 100% (the green LED will be on).
- Return slot 3 to 50% to enable indexing mode (the green LED will stay on).
- Enable rotation by setting slot 2 > 0. The yellow and green LEDs will now both be on.
- The value of slot 4 now controls the indexed angle relative to the selected reference position.

The motor will again be driven to the angle set by slot 4, adjust this slider both up and down.

Do you notice that the indexing value now moves the motor either clockwise or anticlockwise relative to when the reference angle that was set?

Estimate how many steps would be required to make a change of 45 degrees of angle?

- Set the scope to trigger when the waveform changes.
- Change the rotation rapidly from 0 to 360 degrees.
- You could alternatively press the bump button (below the slider on some control desks).

Now examine the waveform:

- Measure the duration of each pulse sent to the motor.
- How many pulses are required to complete a 360 degree move?
- Divide the count by 8 to estimate the number of pulses required for 45 degrees of rotation.

Activity I: Servo Control (Optional)

The goal of this activity is to use the microcontroller to decode the DMX signal and generate a more complex PWM signal in the format required to control a servo motor.

Additional Equipment

5v Servo (1 each) +5v PSU (1 each)
Jumper wires to connect the servo

Background

The servo signal is generated using the Arduino servo library. This uses the PWM hardware on the AVR to generate the control signals. The hardware uses timers to generate an accurate signal. Each servo is controlled by a digital control pin. However, a large servo will draw ***significantly more*** current than available from a microcontroller, and hence ***must*** be fed from an independent power supply. Connect the servo power (red) and ground (black) respectively to a +5V DC supply and ground.

The pulse width controls the rotation of a servo using a standard 50Hz control signal. The servo controller generates a high pulse for a certain duration. Most standard remote control servos (e.g. used in model ships and model aircraft) use a pulse of 1 millisecond duration to indicate 0 degrees of rotation and 2 milliseconds for 180 degrees. e.g. 90 degrees is set via a 1.5 millisecond high signal. This signal is suitable for a wide range of +5V servos (e.g. products manufactured by Futuba or Tower Pro). A typical servo uses gearing that has been designed so that the angular movement corresponds to a rotation between 0 and 90 degrees.

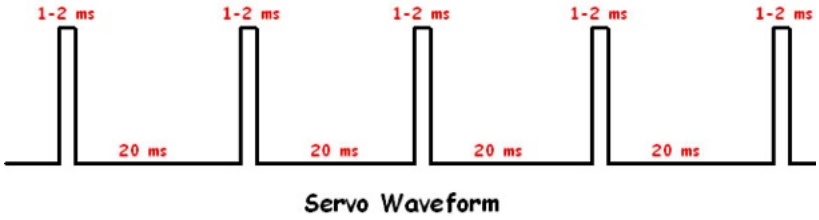


Figure: A Typical Servo Waveform

The output signal to the servos should be connected to pins 12 and 13 on the Arduino shield.
Set the board to servo mode (6, 110 in binary), and a suitable base address.

Connect the scope to the output pin and observe the actual output value.

- Connect the servo control input (white) to the servo output of the DMX receiver.
- Vary the slot value - does the servo track the control value?
- Find the maximum and minimum rotations and their corresponding DMX slot values.
- Use your measurements to calculate the control slot value required to achieve a rotation of 15 degrees and and record the waveform for this DMX slot with the corresponding output that is provided to the servo controller.
- Use your measurements to calculate the control slot value required to achieve a rotation of 45 degrees and record the waveform for this DMX slot with the corresponding output that is

provided to the servo controller.

Activity H: Demonstration of Stepper Motor Demo Unit

Description

The demonstration unit uses an Arduino Mega

It uses the same stepper motor and controller as used in the lab.

The stepper motor driver is powered from an external 9V-12V supply

Code compilation notes: The unit is based on a Rev II board and the software Rev from Aug 2023. It must be compiled with local_use enabled.

Control

Control A3: Speed (Can be used without a DMX signal)

0-255 - over-rides the DMX rotation speed (and sets forward direction).

Full - disables the control knob and uses a DMX value to set the rotation speed.

Control A4: Index (DMX required to set the index reference)

0-255 - over-rides the DMX index value (indexing enabled and reference set using DMX).

Full - disables the control knob and uses a DMX value to control indexing.

A sensor is provided - and this output is on a BNC connector, this could later be used to set the index reference position for the receiver. This part of the code has not been written yet.