Low speed serial buses are widely used today in mixed-signal embedded designs for chip-to-chip communication. Their ease of implementation, low cost, and ties with legacy design blocks make them ideal for a wide variety of applications across a broad number of industries. At debugging stage, low speed serial buses often provide a key visibility point for engineers. However, using traditional oscilloscope to debug serial buses is ineffective and requires significant manual effort.

In this session, I will share with you on how low-speed serial scope application software simplifies serial bus trigger and decode, and help you to accelerate in your debug. In addition, debugging serial bus may sometimes require engineers more than just to trigger and decode on serial bus signal. We will discuss methods for viewing long time of serial activity and capture error packet data.
First we will look at the overview of various low-speed serial buses technology of different industries. We shall move on to appreciate how serial bus application help us automate in serial bus trigger and decode by comparing to tedious manually trigger and decode method. Although most serial buses are digital in nature but the triggering and decoding can be achieve by using a Digital Storage Oscilloscope (DSO) that has only scope channel, or with a Mixed-Signal Oscilloscope (MSO) which has both scope and logic channels. We will investigate into what are the advantages of using a MSO as compared to DSO for serial bus debugging. Debugging embedded design including serial buses often requires engineer to capture long period of signal activity. We will look into the technology that enable oscilloscope to capture long period of serial buses data.
Low-speed serial buses are used in a wide variety of electronics products. Various low-speed buses exist in different industries. Low-speed serial buses are used to communicate between entities. The entities could be chip-to-chip, different function blocks, or board to IO or vice versa. The other key characteristics of low-speed serial bus is its few pin counts makes signal trace routing simple.

I2C (Inter-Integrated Circuit) and SPI (Serial Peripheral Interface) are popular bus that used to attach low-speed peripherals. They are very popular in almost all industries because of its simplicity and low manufacturing cost factor. I2C uses only two bidirectional open-drain lines, Serial Data (SDA) and Serial Clock (SCL), with three basic types of messages, each of which begins with a START and ends with a STOP. SPI is a two or three wire with Chip Select (~CS), Data and Clock (CLK) to send stream of bits.

RS232 is a standard for serial binary data signal connecting between a DTE (Data Terminal Equipment) and a DCE (Data Circuit-terminating Equipment). In RS232, user data is sent as a time-series of bits. Both synchronous and asynchronous transmissions are supported by the standard. RS232 is a two-way (full-duplex) communication single-ended data transmission. RS422 and RS485 are designed for greater distances and higher Baud rates than RS232. Both the technologies uses the same protocol as RS232 but with differential data transmission.

For automotive serial buses, CAN (Controller Area Network) is a differential 2-wire interface with data rates ranging from 10kbps to 1Mbps. CAN serial bus is used in multiple applications include window & seat control, engine management and anti-skid systems. LIN (Local Interconnect Network) is a class A protocol operating up to 19.2kbps over a cable length up to 40 meters. Typical applications include window control and other non-time or none safety-sensitive functions such as comfort controls. FlexRay is a time-triggered, fault-tolerant high-speed communication system using point-to-point links up to 10Mbps. Future application include x-by-wire (steering, braking).

Some automotive systems involve a few low-speed serial buses employed together to improve efficiency of system communication and reduce cost. Example like the I2C and SPI protocols are most often used for chip-to-chip communication within electronic control units (ECUs). For long-haul serial communication between various automotive subsystems (such as anti-lock breaks, airbag, engine control, and GPS navigation), the CAN or LIN protocols implemented.
There are few areas where engineers could use oscilloscope in debugging or validating low-speed serial bus design. First of all, oscilloscope can be used to measure the electrical signals of a device-under-test (DUT). This includes the voltage and timing measurements on the signals of interest to ensure they are within their I/O standard.

A more powerful usage model is to use the oscilloscope to measure low-speed serial bus signal as system level debugging. Most embedded designs nowadays are really a mixed-signal design, where it could take analog signal as input with parallel or serial digital signal as intermediate or output signal; or at the reverse way where it takes parallel or serial digital signal as input and output analog signals depending on how you look at the system block. A mixed-signal oscilloscope (MSO) excels at system level debugging. This is because MSO not just able to view analog and digital signals, but to trace the serial bus content to specific point at which data become corrupted. This, in turn, helps engineers solve questions such as whether the system is passing the right serial data value out, tracing to the corrupted serial data bit, know the frequency and when exactly the error occur and etc.
Let's quickly review what are the tools available for low-speed serial bus debugging before we move on. Each of the tool listed up here can be used to debug low-speed serial buses. Most digital oscilloscopes include both the DSO and MSO have serial application package that helps user to debug and test on serial buses. Other tools such as Logic Analyzers and bus specific tools also can be used for low-speed serial bus design. Each tool excels at different aspects than the others. For today’s seminar, we will focus on the serial application package with oscilloscope.
Impossible to set unique trigger using Edge/Pulse Width
Lacks sequential pattern triggering

"Engineering solutions!"
Save the signal capture and feed to software program that specially developed
Be bold, go head-to-head with the serial bus!!

The screen shot on the right shows the Serial data (SDA) and Serial Clock (SCL) of an I2C signal captured with scope. How could engineers trigger on specific event on this I2C signal using traditional oscilloscope? The short answer is that it is almost impossible for engineers to set unique trigger using Edge or Pulse Width. The situation will become even harder if engineers need to identify first serial pattern before triggering on second serial pattern, i.e. lacks of sequential pattern triggering!

The alternative ‘engineering solutions’ is that user needs to specially develop a software program that can identify a certain serial pattern selected by the user, feed the single-capture signal from scope into this software program and pray for that specific serial pattern is in this single-capture signal by chance. If the serial pattern does not show up in the signal, user needs to recapture another single-capture signal and feed to the software again. The process repeat until the specific serial pattern appeared in one of the single-capture signal.

Or, if the engineer is ‘bold’ enough, he or she will go on head-to-head with the serial bus captured! Trying to manually identify the specific serial pattern!
This page illustrates how an engineer would have manually identify a specific serial pattern of his interest. The yellow trace is the Serial Data (SDA) and the blue trace is the Serial Clock (SCL) of a I2C serial stream. The engineer needs to first ‘single’ capture this stream, and follow by identifying the start and stop condition of this stream. After this, based on the SCL high, user need to identify the ‘1’ and ‘0’ of this stream. Based on the ‘1’ and ‘0’ that being identify, user needs to start counting the ‘1’ and ‘0’ to identify the Control Byte, Write, Ack, first Data, second Data. After all these efforts then only the engineer can identify that the data value of this stream is 20 (Hex). But, oppss! 20(Hex) is not what this engineer is interested in!

So, naturally the engineer will try to repeat all the steps and re-capture another ‘single’ capture of I2C stream!

As an engineer, should you spend your time this way to try to get the I2C pattern you want?
Luckily you don’t have to do what’s shown in previous slide to capture the specific I2C serial pattern you want. Most modern oscilloscope offer serial bus trigger and decode capability. Agilent serial trigger and decode option enables user to quickly pinpoint to the area of interest on serial bus signal and includes capabilities such as sequential and error frame triggering. When serial application packaged is used on Mixed-Signal Oscilloscope (MSO), it present a power tool for user to view all the analog, digital and serial signal from system activities. Lastly, Agilent serial trigger and decode key differentiator is the hardware-accelerated decode engine that enables fastest decode rate and therefore correlation with analog or digital channels for mixed signal debug.
To trigger an I2C signal, the user first needs to let the scope know the correct I2C data (SDA) and clock (SCL) channels. The screenshot on the left-hand side shows channel 3 is selected for the I2C clock signal SCL. Note that logic channel D0 to D15 can also be selected as the I2C clock signal SCL if we probe the I2C clock signal using a logic probe. In this case, channel 1 is selected for the I2C data signal SDA.

The next step is to select a trigger condition for this I2C signal from a selection menu. In this example, we are triggering on EEPROM Data Read at data 41Hex. Now we have a stable trigger on the I2C signal.

To modify the trigger condition, just select a different triggering condition from the selection menu. Changing one triggering to another triggering condition is just a matter of a few seconds with the Serial Trigger & Decode application. Note that the user can also select a two-stage sequencing triggering condition at “Frame(Start:Addr7:Read:Ack:Data:Ack:Data2)” and set the data value for ‘Data’ and ‘Data2’. All of these are just within seconds!
To decode the data of the triggered I2C signal, just select and active the serial decode capability.

With this trigger and decode, handling serial bus signal is definitely easier than the 'head-to-head' manual triggering. In addition, once the triggering condition and decode is set, the scope will automatically read out the serial decode signal from DUT.
Similar to I2C, triggering and decode on RS232 signal is easy with application software package. This page here outline the RS232 trigger and decode selection. User can select the number of bits of 5 to 9 bits, with parity odd, even or none. User can also select the wide range of UART baud rate from 1200b/s to 2.7648Mb/s. Other selection such as polarity and bit order can be made.

At the decode portion, Agilent RS232 serial trigger and decode application provides a real-time totalizer/counter of transmitted and received frames. This option allows user to count parity error bytes and see a percent readout that gives an indication of the quality of your serial bus. This is a unique solution from Agilent and not available from other oscilloscope currently in the market. This totalize function is independent from the scope’s acquisition or triggering, hence the counts run continuously, even when the scope’s acquisition is stopped. Other feature is that user can choose to view the decoded data in either Hex, binary or ASCII format.
In next section let’s review how a DSO differs from MSO and in the context of low-speed serial bus debugging.
A Digital Storage Oscilloscope (DSO) is a digital oscilloscope that has only scope channel (also known as analog channel), whereas a Mixed Signal Oscilloscope (MSO) has both the scope channel and logic channel (also known as digital channel). Both DSO and MSO can be used for low-speed serial signal measurement as long as the scope has the serial application packages. These packages include I2C, SPI, CAN, LIN and FlexRay. Note that both DSO and MSO would require a 4 channel configuration in order to run with serial application packages.

The first screen capture shows a DSO with I2C signal being triggered and decode. The DSO uses its scope channel for measuring I2C data (SDA) and clock (SCL) signals. The second screen shot shows a system level debugging scenario, one scope channel (red trace) is used to view at the serial signal input and the other scope channel (yellow trace) is used to view the Digital-to-Analog Converter (DAC) signal as system output. The intermediate digital signals are captured using MSO logic channel. Note that in this screen the I2C serial bus decoded data is show on screen as well. Therefore with a MSO, it has a total of 20 channels (4 scope channel + 16 logic channel) and the serial decode data for user to perform mixed-signal debugging.
Let us review on a debugging mixed-signal embedded design Mixed Signal Oscilloscope (MSO) example. At the core of product is a Microchip PIC18F452-I/PT microcontroller, which operates on an internal 16-bit instruction set. Other component from this design example include external analog-to-digital converter (ADC) and digital-to-analog converter (DAC). This mixed-signal device and its associated external circuitry provides a perfect example of using an MSO to turn-on and debug an embedded mixed-signal design.

The ultimate goal of this design was to generate various length, shape, and amplitude analog “chirp” output signals based on a variety of analog, digital, and serial I/O input conditions as mentioned earlier. Depending on the status of the three analog, digital, and serial inputs, the MCU was programmed to generate a series of parallel output signals to an external 8-bit DAC to create an analog chirp signal of various amplitudes, shapes, and lengths. The unfiltered stair-step output of the DAC is then fed through an analog low-pass filter to smooth the output signal and reduce noise. Finally, the MCU generates a parallel digital output via another available digital I/O port to drive an LCD display that provides the user with system status information.

As shown on the screen capture, we can all the digital, analog and serial signal with MSO scope.

Let’s now take a look at how this design was turned on and tested…
After turning on and verifying proper operation of the external DAC and analog filtering, the last step in this design/turn-on process was to write code to generate a specific number of non-repetitive sine wave pulses (chirps) based on a serial I²C input. The top-left waveform display shows an overlay (infinite-persistence) of various length chirps using standard oscilloscope edge triggering. Unfortunately, with conventional oscilloscope edge triggering it is impossible to qualify triggering on specific length chirps.

Using the I²C triggering capability, the Agilent MSO can synchronize its acquisitions on specific serial input conditions that instruct the MCU to generate specific length (number of pulses) output chirps. On the bottom left screen shot, the scope is triggering on 21Hex data and the scope channels shows a single chirp sine wave. On the bottom right screen shot shows scope is triggering on 45Hex with three sine wave chirps.
There are few ways in low-speed serial bus decode method performed by the oscilloscope. Why would you care about serial bus decoding methods? This is because different decoding method will affect the scope update rate. Faster update rate ensures all the serial frames get decoded in the oscilloscope, and this in turn enables meaningful signal correlation between scope channel and serial decode data.
There are two methods in low-speed serial bus decoding, the first one is called software-based decode, and the other method is hardware-accelerated decode. Software-based decode is post-processing decode via oscilloscope software. This method slows down the scope update rate especially when the scope is operated under deep memory. Slower update rate affect two key areas in the low-speed serial bus debugging. First, not all serial decode frames get decoded! And the scope channel will miss subtle signal details such as jitter or glitch signal and miss infrequent events.

Agilent oscilloscope employs hardware-accelerated trigger and decode for low-speed serial bus application. It means that the low-speed serial bus is being decoded on internal ASIC at hardware speed and maintains fast update rate on scope channels. Therefore there is no compromises to serial decode application and all the serial frames get decoded. The update rate also maintained and there is no compromises to scope channel, enable user to see subtle signal details and finding infrequent events. Hence, meaningful signal correlation is possible for system level debug.
In this mixed-signal measurement comparison, we are now observing a CAN serial bus using both serial triggering and serial protocol decoding on both MSOs. The error rate in this example is approximately 2% (1 part in 50). In addition, when triggering on a particular CAN frame (07F_HEX), the maximum trigger rate is approximately 60 Hz. In other words, CAN frame 07F_HEX repeats approximately every 17 ms. When using the Agilent MSO, the scope updates both waveforms and serial decoding at 60 waveforms per second, which is the maximum trigger rate. This is an example where the scope’s update rate is limited to the trigger rate of the signal. But this also means that the Agilent MSO captures **EVERY** trigger event and misses nothing. The probability of capturing the error frame condition (indicated by the flashing red ERR message) is 100%. In this example, the scope’s dead-time has effectively been eliminated.

Notice that the other vendor’s MSO updates waveforms and CAN protocol decodes approximately just once every five seconds. This extremely slow waveform and decoding update rate is primarily due to software-based decoding of the waveform. Not only does this MSO have a very poor probability of capturing and detecting the error frames, but this is also a good example of how slow updates can degrade the usability of the scope. After making a scope setup change, such as s/div, you would have to wait 5 seconds before seeing the effect.

Serial decoding on the Agilent MSO has no affect on waveform update rates. This is because the Agilent MSO uses hardware-based serial protocol decoding. InfiniiVision is the only oscilloscope family in the industry that utilizes hardware-based decoding.
In the next section, we will discuss on how to capture a long period of low-speed serial data.
In actual debugging, there will be needs to either capture long period of serial bus signal or to identify certain serial frames out from all the frames transmitted on a share bus. Some serial bus signal like CAN bus is a shared bus. The signal of interest may be just portion of the overall frames that’s in long period of time.

In an example of 2Mpts memory and 1MSa/s on an oscilloscope, how much CAN bus frames can you capture? When you work on the math, you will find the scope can only capture 2 seconds of CAN bus frames (2Mpts x 1MSa/s = 2s). But the question is that will 2 seconds of CAN bus capturing good enough for your needs to observe one particular frame out of all that transmitted? What happen if your particular frame of interest did not repeat within 2 seconds? Is there a more efficient method to capture even longer time periods?
Agilent InfiniiVision oscilloscope offers an application called Segmented Memory to enable users to effectively utilize memory to capture long period of burst and packetized signals. When the low-speed serial bus application pairs up with Segmented Memory application, it presents a powerful usage model so that users may capture meaningful serial bus frames for a long period of time.

The diagram at the lower left depicts the idea of Segmented Memory application. Basically, the application would divide up scope memory into segments with a maximum of 2000 segments. This application then only uses the first segment to capture the first burst or packet signal based upon triggering condition, and uses second segment to capture the second burst or packet signal, and so on and so forth. The end result is that the scope only captures those bursts of packet that is meaningful and leaves behind the portion of signal that is inactivity.
The first screen capture shows that scope is triggered on CAN bus frame ID 7F(Hex), and turning on Segmented Memory with 2000 segments. After all the 2000 segments of frame ID 7F(Hex) are captured, user can navigate each segment by scrolling through using the turning knob on scope front panel. Note that reference time information is available in each segment. The very first segment indicated that the reference time stamp is at 0 second. As user scrolling through the segments, they can perform analysis such as read out of the decoded data and measurement on scope channels. When user scroll to the 2000th segment, note that there is a total of 38 seconds of CAN bus signal being capture. It is impossible to capture 38 seconds of CAN bus signal using traditional oscilloscope without Segmented Memory function.

Another possible usage model is that user can trigger on CAN bus error frame and using Segmented Memory to capture all the error frames. User can perform analysis at each of this error frames.
So far, we have reviewed how useful of low-speed serial application package can help engineer in low-speed serial bus signal. There is no other way to accurately and quickly trigger on low-speed serial bus signal without the application package. Trigger and decode on low-speed serial bus become easy and with hardware-accelerated decode method that provide fast update rate, Agilent Serial Trigger & Decode application enables engineer to perform meaningful signal correlation and save days to weeks of debug time. In addition, if a Mixed Signal Oscilloscope is used, it provides advantages of more scope and logic channels for system level debugging capability. Lastly, the combine usage of Agilent Serial Trigger & Decode and Segmented Memory application packages present a powerful usage mode to capture and analyze longer period of serial signal activity.