Creating a Custom Protocol Decoder

Contents

.NET Dll based Decoder

- Create a .NET Dll using C#(Recommended) and Visual Studio(Recommended).
- Dll should implement the interface ISerialProtocoPlugin or BasePlugin class.
- ISerialProtocolPlugin or BasePlugin can be found in Pico.ProtocolPluginBase.dll found in PicoScope installation directory.
- There are 2 ways PicoScope recognise the new decoder dll.
  - Either place it under PicoScope Installation Directory/Decoders folder
  - Or Use “Import” button in SerialDecoding Setup dialog and locate the dll as in *Fig 1*.

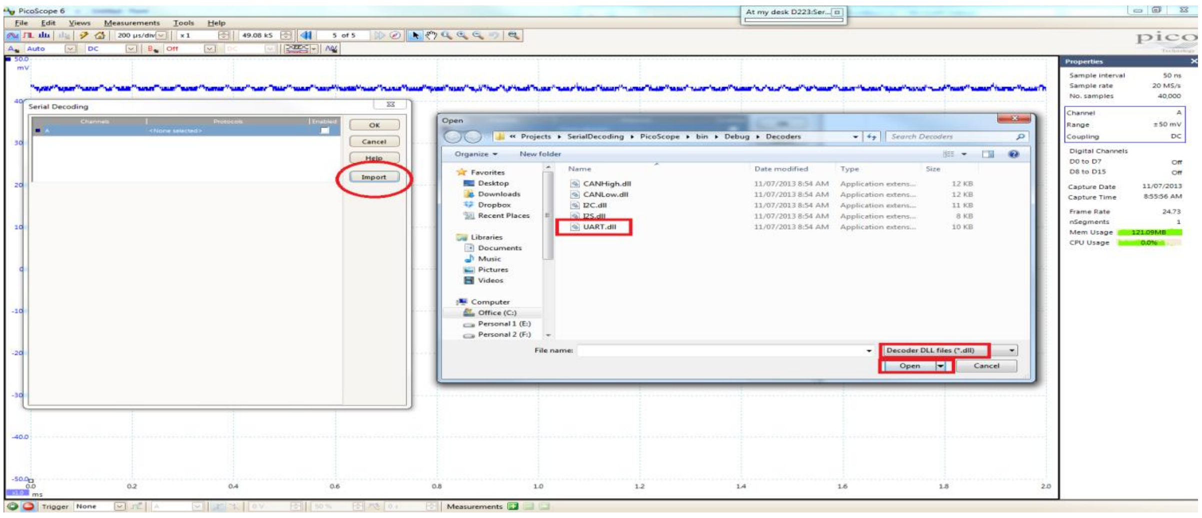


Fig 1

- The Protocol decoder(dll) should then be available in the Protocol List in *Fig 3*

Note: Details of .Net Interface members are explained in below sections

XML based Decoder

- Create a xml file based on the XMLSchema SerialProtocolPlugin.xsd.
- SerialProtocolPlugin.xsd can be found in PicoScope Installation Directory/Decoders folder
- There are 2 ways PicoScope recognise the new decoder xml.
    - Either place it under PicoScope Installation Directory/Decoders folder
    - Or Use “Import” button in SerialDecoding Setup dialog and locate the xml as in *Fig 2*.

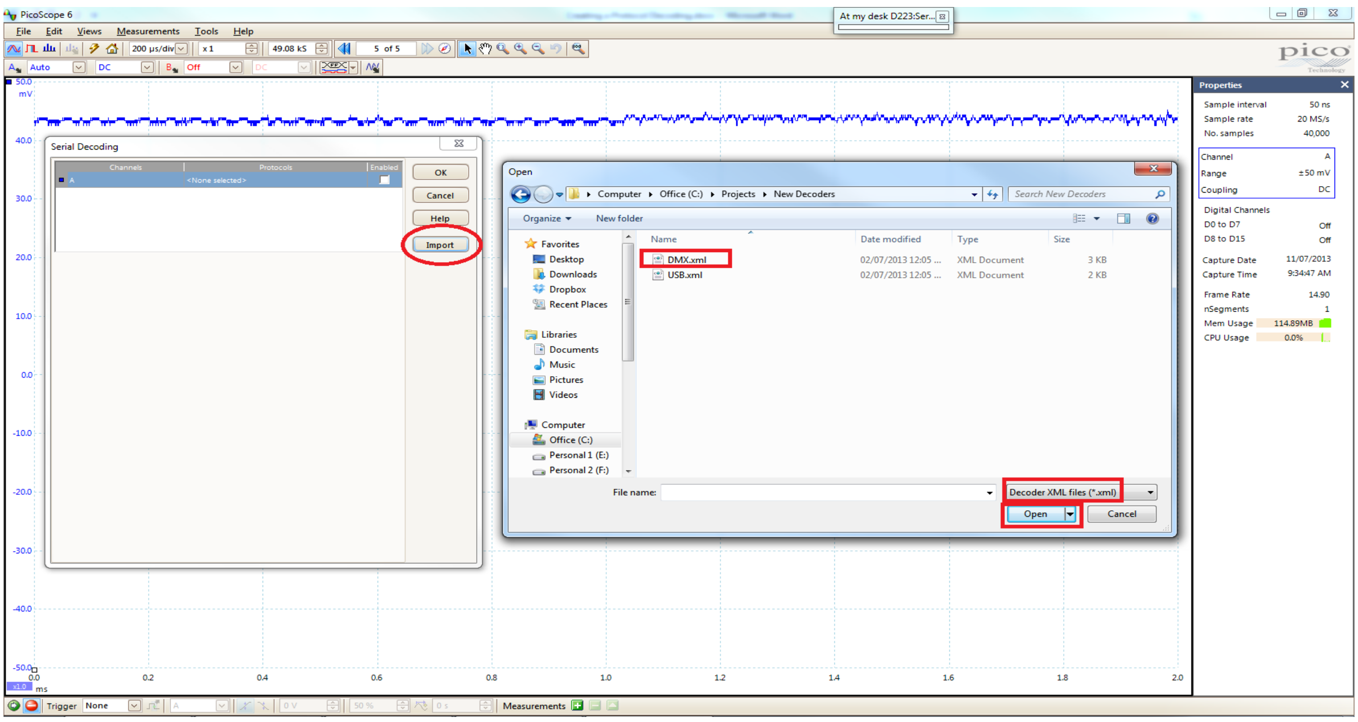


Fig 2

- The Protocol decoder(xml) should then be available in the Protocol List in *Fig 3*

Note: Details of Xml Schema are explained in below sections

Setting up a newly created Protocol Decoder

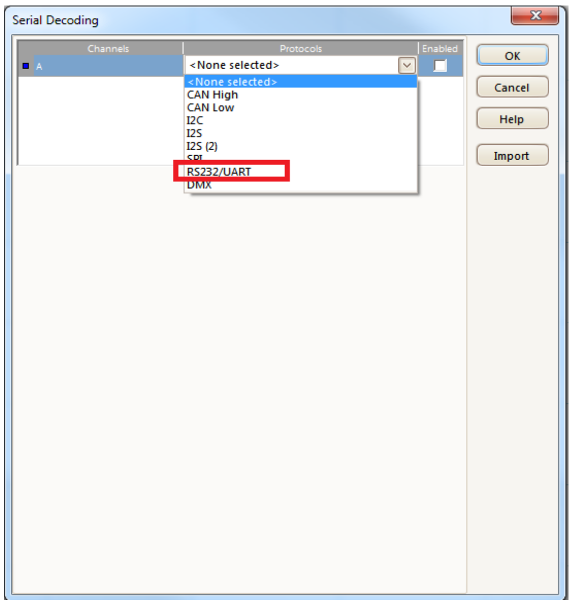
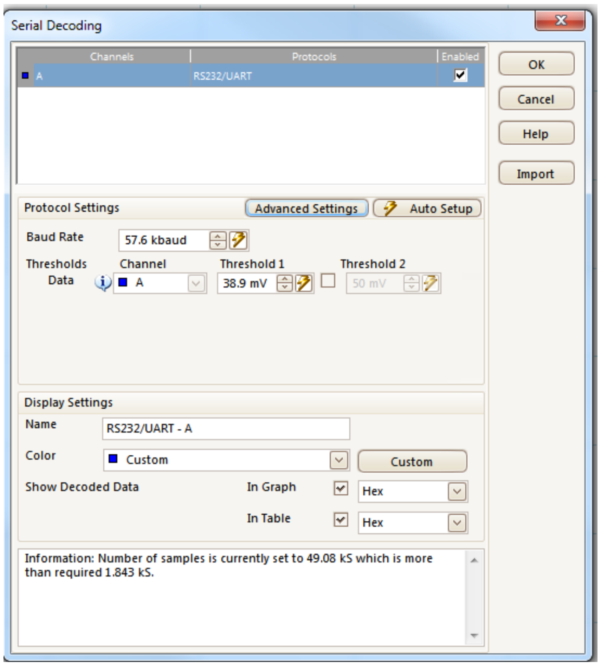


Fig 3

Once the Protocol is selected, then a set of controls will be displayed Under "Protocol Settings" as seen in *Fig 4.*

- These controls will load up with any default settings specified in the Plugin(Dll or xml).
- These settings can be modified and saved.
- Use AutoSetup buttons to let PicoScope detect the settings for you.
- Use Advanced Settings button to customize any additional protocol settings.
- Additional Settings are populated with the settings from the Plugin(Dll or Xml) which are tagged as user settings.
- Under "Display Settings" you can assign a display name and color for the protocol displayed in the PicoScope.
- Click OK and protocol decoder should start displaying packets.



Channels	Protocols	Enabled
A	RS232/UART	✓

Baud Rate: 57.6 kbaud

Thresholds	Channel	Threshold 1	Threshold 2
Data	A	38.9 mV	50 mV

Name: RS232/UART - A

Color: Custom

Show Decoded Data — In Graph ✓ Hex; In Table ✓ Hex

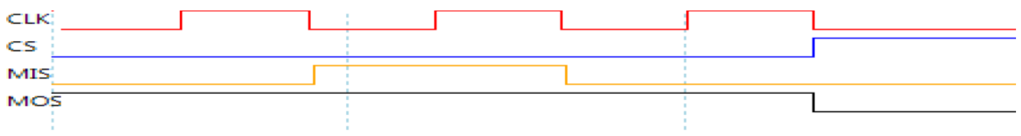
Information: Number of samples is currently set to 49.08 kS which is more than required 1.843 kS.

Fig 4

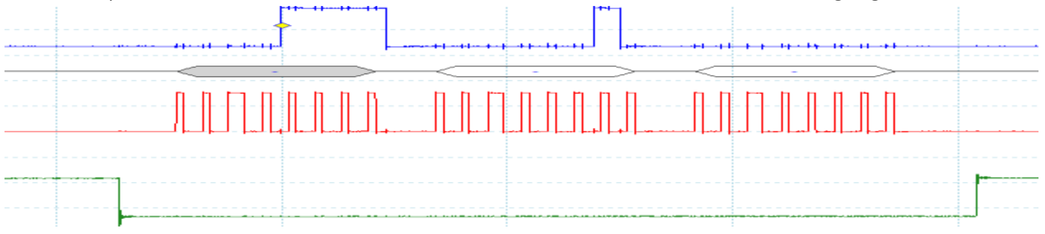
Details of .Net Interface members

Refer to UART.cs(attached sampled) for help with implementation

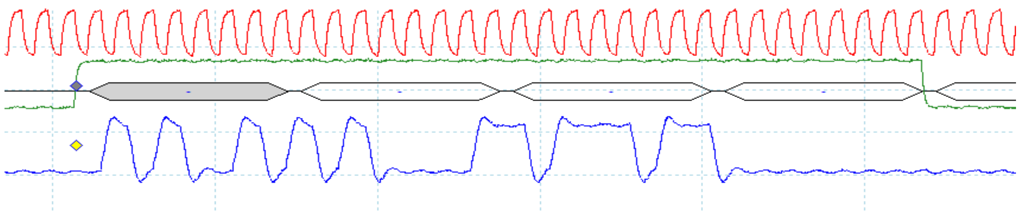
- ***Protocol*** : Gives a name to the protocol which is displayed in the Combo in Fig3
- ***BaudRate*** : If protocol has a fixed baud rate return it. Else return "float.NaN" .If a valid value is specified then a Spinner control will be created with default settings. It can be modified if required.
- ***BitOrder*** : Specify whether the First bit sent on the channel is LSB or MSB. In PicoScope packets are always displayed as MSB to LSB. For example, if MSB is specified then the packet bits will be swapped and displayed. This setting can be made to appear in the Advanced Settings Dialog. Set a string value to DisplayAsControl and it will displayed with this label and a combo with LSB and MSB as options
- ***InvertPacketContent*** : If set to TRUE, then 0 is converted to 1 and 1 to 0 before displaying. This setting can be made to appear in the Advanced Settings Dialog. Set a string value to DisplayAsControl and it will displayed with this label and a combo with TRUE and FALSE as options.
- ***HasBitStuffing*** : Specify the number of bits used in BitStuffing. Later in GetPacketFileds() we should also mention the bits which are involved in BitSutffing.
- ***ClockChannelIndex*** : All the multi channel protocol decoding is based on a clock channel which is very important. So, return the index of the channel in the Plugin which is used as Clock. Later in GetChannelDetails() all the channels are defined. Index returned here refers to the channel index in GetChannelDetails() list.
- ***NeedsEdgeValidation*** : This is used to validate the channel(mentioned in the GetChannelDetails()) other than the clock channel. If set to TRUE, then channel edges which happen very close to Clock channel edges are not considered and ignored. If set to FALSE, then edges that happen very close to Clock channel edge is considered as valid condition. This mainly affects in detecting the Packet Start and Bit validations.
  For example, in the below condition, CS changes from LOW to HIGH at CLK(Clock) falling edge.
  In the Plugin if you have specified a Packet Start condition as CS Rising when CLK Falling.
  If NeedsEdgeValidation = TRUE, then this condition is not met because CS Rising happens very close to CLK falling. If set to FALSE, then it is allowed and so condition is met.



- ***IsBitValidAtStartCondition*** : Specifies whether the bit at the Start Condition should be considered as the first data bit. This depends on the GetBitValidations() and GetPacketStartConditions().
  For example : If set to TRUE, then when the Start Condition happens, and if the BitValid condition is met, data will be logged. If set to FALSE, then then Bit Valid will not be logged and Data bit will be the next bit valid after the Start Condition.
- ***NumBitsBeforeNextPacketStart*** – If for a protocol the start of packet is defined by number of bits then set this value. For example, in SPI protocol ChipSelect can go LOW(Green) a long time back before the 1st Clock(Red) Rising Edge. But exactly after 8 bits is the packet start. So if NumBitsBeforeNextPacketStart is set to 8, then after 8 Clock Rising Edges a Packet Start is marked.



- ***NumBitsDelayAfterStart*** : Specify delay in number of bits after the start condition when the Packet actually is formed. For example in I2S protocol, Start condition is Clock(Red) rising and Wordselect(Green) Rising.
  But the packet actually starts at the next Rising Clock Edge and hence a delay of 1 bit.
  If this control has to be listed in the AdvancedSettings control, then return a string set to DisplayAsControl property and a list of options set to OptionsList property



- ***GetChannelDetails()*** : Set the number of channels involved in the Protocol decoding.

  For each channel, return a
  - **DisplayName** - name that should be shown in the Serial Decoding Setup dialog in *Fig 3*
  - **IsOptional** - Option saying whether the channel should be optional in decoding. Doing this will allow you to choose it for decoding or skip it.
  - **Thresdhold1** and **Threshold2** - Lower and Upper thresholds for the channel. When protocol is first selected these values will be shown. Later, if required an AutoSetup can be done to change the values
- ***GetBitValidConditions()*** : Specify the conditions at which a protocol data bit can be collected. BitValidConditions can use one or all channels.

  For each BitValidCondition, return a
  - **ConditionType** – which is either SerialEdge or SerialSignalLevel
  - **ConditionValue** – which is Rising/Falling/RisingOrFalling if Type set is SerialEdge and High/Low/HighOrLow if Type set is **SerialSignalLevel**
  - **DisplayAsControl** – set a string if you want this setting to be displayed in the AdvancedSettings control
- ***GetPacketStartConditions()*** : Specify the conditions at which a protocol packet start is defined. PacketStartConditions can use one or all channels. For each PacketStartCondition, return a
  - **ConditionType** – which is either SerialEdge or SerialSignalLevel
  - **ConditionValue** – which is Rising/Falling/RisingOrFalling if Type set is SerialEdge and High/Low/HighOrLow if Type set is **SerialSignalLevel**
  - **DisplayAsControl** – set a string if you want this setting to be displayed in the AdvancedSettings control
- ***GetPacketTypes()*** : List all the packet types in the protocol.
- ***GetPacketsFollowingStartCondition()*** : List all the packet types which are likely to happen soon after the Start condition is met.
- ***GetPacketFields(packetName)*** : For a given packet name, return all the packet fields.

  For each Field, return a
  - **Name** - Name for the field
  - **BitCount** - Number of bits in the field
  - **OptionsList** - If the number of bits is flexible, then a string se to DisplayAsControl so that it will be displayed in the Advanced Settings control
  - **IsInvolvedInBitStuffing** - Whether the field is involved in BitStuffing
  - **FixedValue** – If there are any fixed values to the field to recognise a packet type. For example, in CAN protocol, if Field IDE = 1, then it is a Data packet type and if it is 0, then it is Remote packet type.
  - **LenDependsOn** – If the bit count depends on another field, return the name of the other field. Later GetPacketLength() will be called to get the bit count.
  - **MappedColumn** – If the Field name and the Column in which it is displayed in the PicoScope is different. Set the column name to which the field decoded value should be displayed.
  - **DisplayAsControl** – set a string to be displayed in the AdvancedSettings control with the OptionsList for BitCount
- ***GetTableColumns()*** : Return all the table columns to be displayed in the PicoScope . Specify the Name of the column and Data type like Hex, Binary , Decimal or ASCII.
- ***GetPacketLength(fieldname, fieldContent)*** : If the bitcount of a field depends on the contents of a previous field, then implement it here.
- ***Intialize()*** : Do an initialization process in the .Net dll before packets creation start.
- ***Finish()*** : Do any completion tasks once the decoding is completed.

IsValid(Dictionary<strin,string>) : For every packet formed from the plugin definition above, this call will be sent to the .NET Dll to check whether the packet is valid or are there any errors in the packet. You can do tasks like CRC checks and other validations in this call and return appropriate messages to PicoScope to be displayed along with the packet.

You can return,
- Whether the packet is Valid or not.
- Any error type to be displayed in the Error column of PicoScope
- Any Special Packet name to be displayed in the Packet column of PicoScope

Details of XML Schema elements

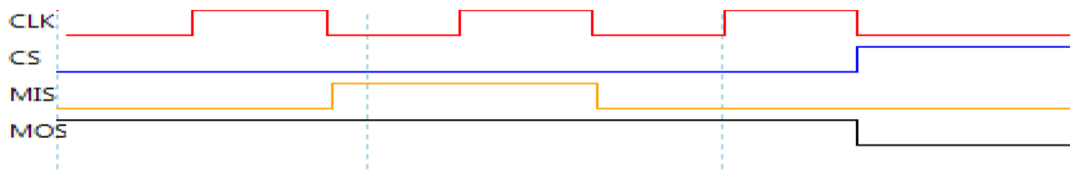
Refer to UART.xml(attached sampled) for help with implementation

Element **ProtocolDetails** : Contains basic protocol details. Elements are:

- ***Name*** : Gives a name to the protocol which is displayed in the Combo in Fig3
- ***BaudRate*** : If protocol has a fixed baud rate return it. Else return “float.NaN” .If a valid value is specified then a Spinner control will be created with default settings. It can be modified if required.
- ***BitOrder*** : Specify whether the First bit sent on the channel is LSB or MSB. In PicoScope packets are always displayed as MSB to LSB. For example, if MSB is specified then the packet bits will be swapped and displayed.
  - **Value** = LSB or MSB
  - **DisplayAsControl** = string text. If you want it to be displayed in AdvancedSettings control
- ***InvertPacketContent*** : If set to TRUE, then 0 is converted to 1 and 1 to 0 before displaying.
  - **Value** = True or False
  - **DisplayAsControl** = string text. If you want it to be displayed in AdvancedSettings control

Element **ChannelDetails** : List of Sources used in Protocol decoding. Elements are

- ***ClockChannelIndex*** : All the multi channel protocol decoding is based on a clock channel which is very important. So, return the index of the channel in the Plugin which is used as Clock. Later in Sources all the channels are defined. Index returned here refers to the channel index in Sources list.
- ***NeedsEdgeValidation*** : This is used to validate the channel(mentioned in the Sources) other than the clock channel. If set to TRUE, then channel edges which happen very close to Clock channel edges are not considered and ignored. If set to FALSE, then edges that happen very close to Clock channel edge is considered as valid condition. This mainly affects in detecting the Packet Start and Bit validations.
  For example, in the below condition, CS changes from LOW to HIGH at CLK(Clock) falling edge.
  In the Plugin if you have specified a Packet Start condition as CS Rising when CLK Falling.
  If NeedsEdgeValidation = TRUE, then this condition is not met because CS Rising happens very close to CLK falling. If set to FALSE, then it is allowed and so condition is met.



- ***Sources*** : Contains a list of channels used in the protocol decoding. Each *Source* element has sub elements like
  - **ChannelIndex** : This is important index, which is used later in the Schema to refer channels by index.
  - **DisplayName** : name that should be shown in the Serial Decoding Setup dialog in *Fig 3*
  - **IsOptional** : Option saying whether the channel should be optional in decoding. Doing this will allow you to choose it for decoding or skip it.
  - **Threshold** 1, **Threshold** 2 : Lower and Upper thresholds for the channel. When protocol is first selected these values will be shown. Later, if required an AutoSetup can be done to change the values

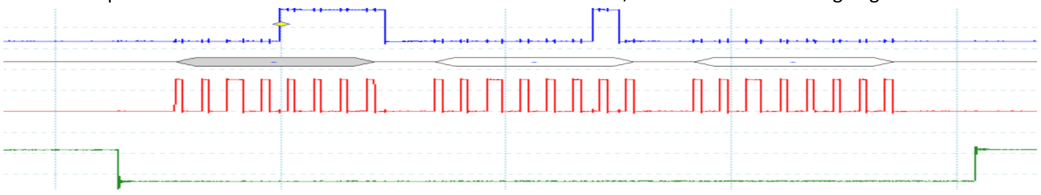
Element **BitValidityDetails** : Specify the conditions at which a protocol data bit can be collected. BitValidConditions can use one or all channels. Elements are:

- ***Conditions*** :
  For each Condition has sub elements like
  - **ChannelIndexRef** : referring to a channel in Sources
  - **ConditionType** – which is either SerialEdge or SerialSignalLevel
  - **ConditionValue** – which is Rising/Falling/RisingOrFalling if Type set is SerialEdge and High/Low/HighOrLow if Type set is **SerialSignalLevel**
  - **DisplayAsControl** – set a string if you want this setting to be displayed in the AdvancedSettings control

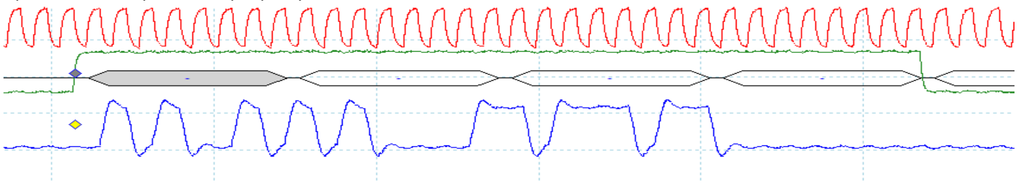
Element **PacketStartDetails** : Specify the conditions at which a protocol packet start is defined. PacketStartConditions can use one or all channels. Elements are:

- ***PacketsFollowingStartCondition*** : List all the packet types which are likely to happen soon
  after the Start condition is met. If multiple packets are to be listed then separate with a “;”. For example “Data;Address”

- ***IsBitValidStartCondition***: Specifies whether the bit at the Start Condition should be considered as the first data bit. This depends on the BitValiditDetails.Conditions and PacketStartDetails.Conditions.
  For example : If set to TRUE, then when the Start Condition happens, and if the BitValid condition is met, data will be logged. If set to FALSE, then then Bit Valid will not be logged and Data bit will be the next bit valid after the Start Condition.
- ***NumBitsBeforeNextPacketStart*** – If for a protocol the start of packet is defined by number of bits then set this value. For example, in SPI protocol ChipSelect can go LOW(Green) a long time back before the 1st Clock(Red) Rising Edge. But exactly after 8 bits is the packet start. So if NumBitsBeforeNextPacketStart is set to 8, then after 8 Clock Rising Edges a Packet Start is marked.



- ***NumBitsDelayAfterStart*** : Specify delay in number of bits after the start condition when the Packet actually is formed. For example in I2S protocol, Start condition is Clock(Red) rising and Wordselect(Green) Rising.
  But the packet actually starts at the next Rising Clock Edge and hence a delay of 1 bit.
  If this control has to be listed in the AdvancedSettings control, then return a string set to DisplayAsControl property and a list of options set to OptionsList property



- ***Conditions*** : Specify the conditions at which a protocol data bit can be collected.
  PacketStartConditions can use one or all channels.
  For each Condition has sub elements like
  - **ChannelIndexRef** : referring to a channel in Sources
  - **ConditionType** – which is either SerialEdge or SerialSignalLevel
  - **ConditionValue** – which is Rising/Falling/RisingOrFalling if Type set is SerialEdge and High/Low/HighOrLow if Type set is SerialSignalLevel
  - **DisplayAsControl** – set a string if you want this setting to be displayed in the AdvancedSettings control

Element **PacketDetails** : Specifies a list of packets defined in the protocol. Elements are:

- ***Packet*** : Specifies a packet type. The sub elements are:
  - **Name** : name of packet
  - **PossibleNextPacketTypes** : defines the possible packet types after this packet type. If multiple packets are to be listed then separate with a ";". For example "Data;Address" .

  Each *Packet* contains multiple fields. Each ***Field*** element contains sub elements:
  - **Name** - Name for the field
  - **BitCount** - Number of bits in the field
  - **OptionsList** - If the number of bits is flexible, then a string se to DisplayAsControl so that it will be displayed in the Advanced Settings control
  - **PacketRecognisers**– If there are any fixed values to the field to recognise a packet type. For example, in CAN protocol, if Field IDE = 1, then it is a Data packet type and if it is 0, then it is Remote packet type.
  - **LenDependsOn** – If the bit count depends on another field, return the name of the other field. Later GetPacketLength() will be called to get the bit count.
  - **MappedColumn** – If the Field name and the Column in which it is displayed in the PicoScope is different. Set the column name to which the field decoded value should be displayed.
  - **DisplayAsControl** – set a string to be displayed in the AdvancedSettings control with the OptionsList for BitCount

Element **TableColumnDetails** : Specifies a list of Columns that will be displayed in the PicoScope. Elements are:

- ***Column***: Specifies a column type. The sub elements are:
  - **Name** – name of the column
  - **Type** - display type of the column which is either Hex, Binary, Decimal or ASCII

Limitations

- Clock channel is always required for a multi-channel protocol decoding
- Supports only 1 data channel decoding
- Supports only the following line coding : NRZ, Unipolar NRZ, Bipolar NRZ,