Chapter 51

Using the KAD/UBM/103

TEC/NOT/079

This paper discusses the following topics:

51.1 RS-232/RS-422/RS-485 overview

This section introduces RS-232/422/485, focusing on the physical layer and the bit definition.

51.1.1 Physical layer

RS-232 is single ended, so the difference voltage is relative to ground. RS-422 and RS-485 are differential ended, so the difference voltage is between the positive and negative terminals. RS-422 and RS-485 are nominally independent of ground, but if the ground potential differs by too much between end nodes, then no data is received.

The RS-232, RS-422 and RS-485 logic 0 is less than -200 mV. Logic 1 is more than 200 mV (see the following figure).

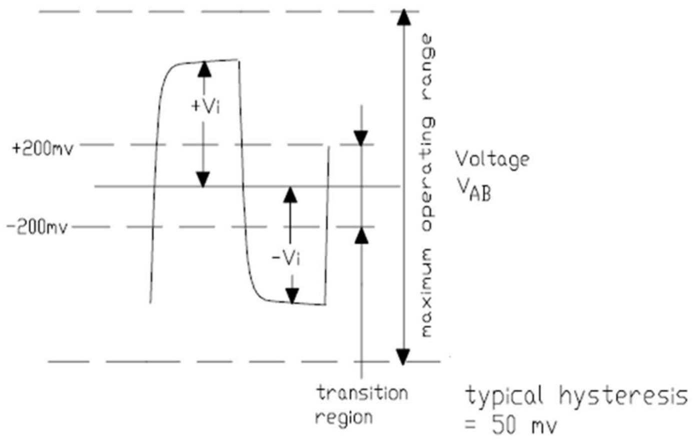


Figure 51-1: Differential input

RS-422 and RS-485 require a termination at the end of the transmission line (see the following figure).

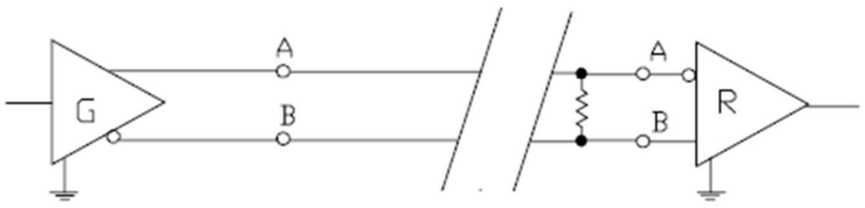


Figure 51-2: Termination differential input

51.1.2 Bit definition

A data word can be either seven or eight bits in length. There is 1 start and 1 stop bit and 1 optional parity bit. A parity bit can be placed at the end of each data word (see the following figure).

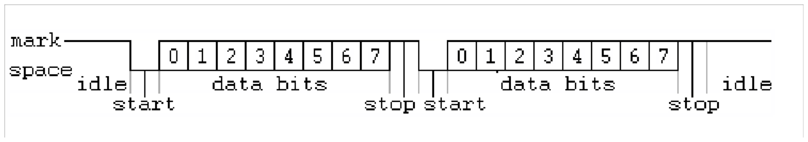


Figure 51-3: Serial bit definition

As shown in the previous figure and the following figure, before the serial communication starts, there is an idle voltage in the bus which is represented as a constant value. At the beginning of each transmission, the start bit is transmitted indicating to the receiver that a flow of 7 or 8 bits of data is about to follow. The start bit is represented as a signal edge opposite the idle time. After the data bits are transmitted, there is an optional bit called parity bit which is an error basic detection mechanism that indicates whether the transmission contains an even or odd number of bits transmitted. The stop bit is the last one in the transmission and is represented as a signal edge opposite the start bit (it is the same as the idle time). The KAD/UBM/103 uses 1 stop bit; this setting is not programmable.

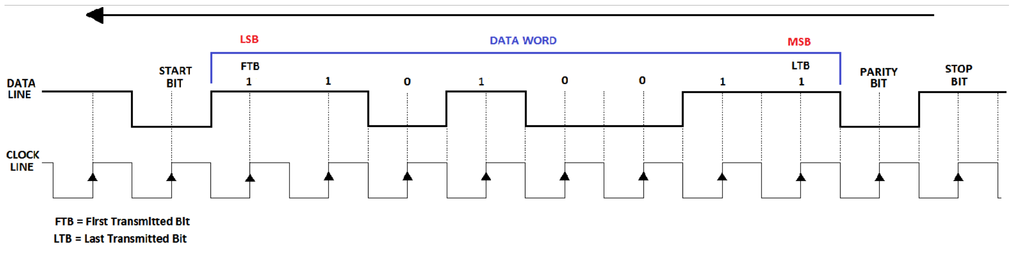


Figure 51-4: Example of sending 0xCB as 8 bits payload and parity

51.2 Module overview

The KAD/UBM/103 is a 16-channel RS-232, RS-422, RS-485 which can parse and/or packetize each channel at the same time.

You can select the Signal Type (RS-232, RS-422, RS485), Baud Rate (0.2-1 Mbps), Data Bits Per Word (7/8 bits per word), and Parity (None, Even, Odd) on a channel-by-channel basis.

Screen shots and descriptions of settings shown in this technical note are from DAS Studio 3 software. DAS Studio 3 is used to create a configuration, which contains the various elements that make up your data acquisition system. You then use this configuration file to manage and program these elements.

To see how hardware is represented in the DAS Studio 3 graphical user interface, refer to the *DAS Studio 3 User Manual*.

51.2.1 Key features

- Monitors up to sixteen independent input RS-422/485/232 busses
- Bit-rates from 300 bps to 1,000,000 bps
- 7/8 bits per word with odd, even or no parity
- Programmable start sequence (1 to 8 bytes), stop sequence (1 byte or by fixed length) and message gap (idle time)
- Coherently parses traffic and tags for up to 511 messages per module. Each message can be 4 to 1024 bytes long
- Aperiodic transmission of packetized serial messages including tags as iNET-X parser-aligned payload structure
- Message wide stale and skipped indication

51.3 Serial module history

The following table below describes the different serial modules.

Module	Description
KAD/UAR/002	RS-232/422/485 bus monitor parser – parse up to 126 messages from 9 to 511 bytes - 300 bps to 115 kbps - 4ch

Module	Description
KAD/UAR/102	RS-232/422/485 bus monitor parser and snarfer – parse up to 125 messages from 4 to 512 bytes - 300 bps to 1Mbps - 4ch
KAD/UBM/103	RS-232/422/485 bus monitor parser and packetizer – parse up to 511 messages from 4 to 1024 bytes - 300 bps to 1 Mbps - 16ch
KAD/UBM/105	RS-232/422/485 bus monitor parser and packetizer – parse up to 511 messages from 4 to 1024 bytes - 300 bps to 5 Mbps - 12ch
KAD/UBM/106	RS-232/422/485 bus monitor snarfer - 300 bps to 1 Mbps - 16ch

51.4 Parser operation

51.4.1 How parsing works

Like other Curtiss-Wright bus monitors, the KAD/UBM/103 uses a triple buffer for parsing. The following figure illustrates the triple buffering of data words (green) and time message tags (white) used for each bus in the KAD/UBM/103's parser.

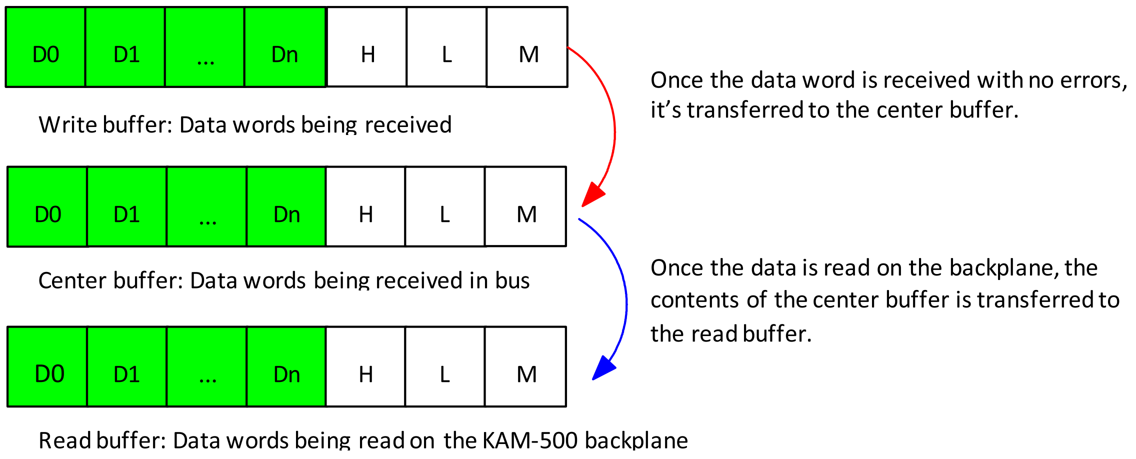

D0
D1
...
Dn
H
L
M
Write buffer: Data words being received
Once the data word is received with no errors, it's transferred to the center buffer.
Center buffer: Data words being received in bus
Once the data is read on the backplane, the contents of the center buffer is transferred to the read buffer.
Read buffer: Data words being read on the KAM-500 backplane

Figure 51-5: Triple buffering of traffic and associated message tags

In the previous figure, D0, D1, D2, Dn corresponds to the traffic data words with n < 511.

The time tags H, L, M correspond to High time, Low time and Micro time, which is the time of the first transmitted bit of the message with a 1-µsec resolution.

The way triple buffering works is as follows:

Time message tags are added to each message received and stored in separate buffers for each of the four busses. As soon as a message is received with no errors, the contents of the write buffer is transferred to the center buffer. If the data in the center buffer has not been transferred to a read buffer, a skipped flag is set.

As soon as the last parameter of interest has been read from the buffer being read by the backplane, the contents of the center buffer (if new) are transferred to the read buffer. If no new data word has been received, the stale flag is set. A center and read buffer exist for every message ID (parser slot). Skipped and stale bits can be found in the Message Info register to indicate whether messages are lost or repeated (undersampling or oversampling situations).

Additional tags such as Message Count, Message Size, and Message Info registers are also available as additional information and can be added from DAS Studio's Serial Builder application as explained in "51.6.4 Adding parameters to the package" on page 9. For further information regarding these registers, refer to the *KAD/UBM/103* data sheet.

51.5 Module Settings tab

51.5.1 Parser Data Endianness

In the parser, a total of up to 511 complete messages are triple buffered so that the stale indication is message-wide. Each message can be up to 1024 characters (bytes) long (including start and stop characters). Each message is tagged to 0.1 ms resolution; a message is considered found when a start sequence up to 8 user-defined characters are received. The end-of-message delimiter is determined by either a user-defined stop character or a specific number of bytes. A parsed message is not updated/parsed if the start sequence does not match.

NOTE: To view the screen shots shown in this section in DAS Studio 3, ensure the KAD/UBM/103 module is in context and the Settings tab is selected.

To configure the parser, first you must decide on the endianness of the data you wish to receive.

As shown in the following figure, there are three choices available for Parser Data Endianness.

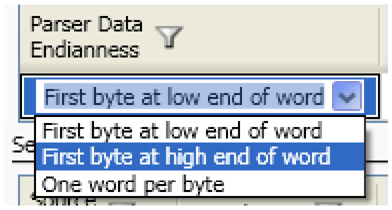

Parser Data Endianness
First byte at low end of word
First byte at low end of word
First byte at high end of word
One word per byte

Figure 51-6: Parser Data Endianness settings

Considering the input data ABCD:

First byte at low end of word returns the hex values 0x4142, 0x4344, which corresponds to the decimal values 65, 66, 67, and 68, the ASCII codes for ABCD.

First byte at high end of word returns the hex values 0x4344 0x4142, which corresponds to the decimal values 67, 68, 65, and 66, the ASCII codes for CDAB.

One word per byte returns one byte of data per 16-bit word, with the lower byte padded with 0s. That is, 0x4100, 0x4200, 0x4300, and 0x4400, which corresponds to the decimal values 65, 66, 67, and 68, which are the ASCII codes for ABCD once the padded bits are removed.

NOTE: This byte arrangement is equivalent to KAD/UAR/102 even or data words.

51.5.2 Setting up the incoming signal

In this section you define the signal type of the incoming data for that channel.

As shown in the following figure, there are three choices available for Signal Type.

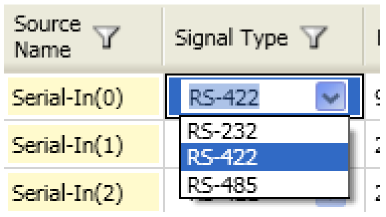


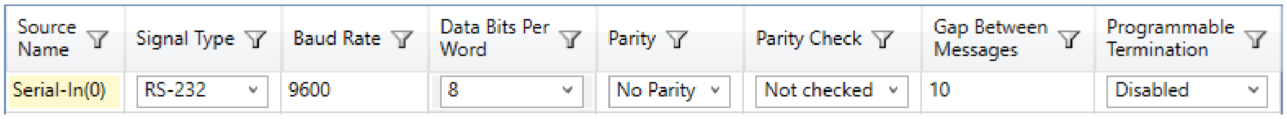
Figure 51-7: Signal Type settings on channel 0

RS-232 is a single ended input signal. For this mode, the data source must be connected to the positive input pin of the channel pair and the negative input must be left unconnected.

RS-422 and RS-485 are both differential inputs, so the source positive signal must be connected to the positive input, while the source negative input must be connected to the negative input.

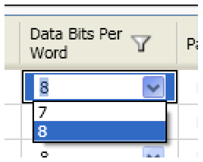
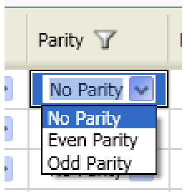
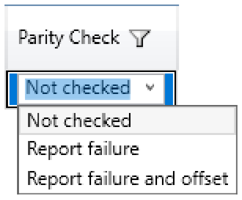
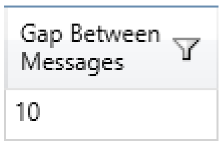
For each input pair, there are individual termination pins. To use these, the negative input must be connected to the correct termination pin for that input.

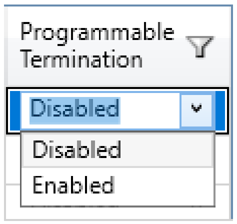
The remaining channel settings to be configured for the parser are Baud Rate, Data Bits Per Word, and Parity as shown in the following figure and the table that follows.



Source Name	Signal Type	Baud Rate	Data Bits Per Word	Parity	Parity Check	Gap Between Messages	Programmable Termination
Serial-In(0)	RS-232	9600	8	No Parity	Not checked	10	Disabled

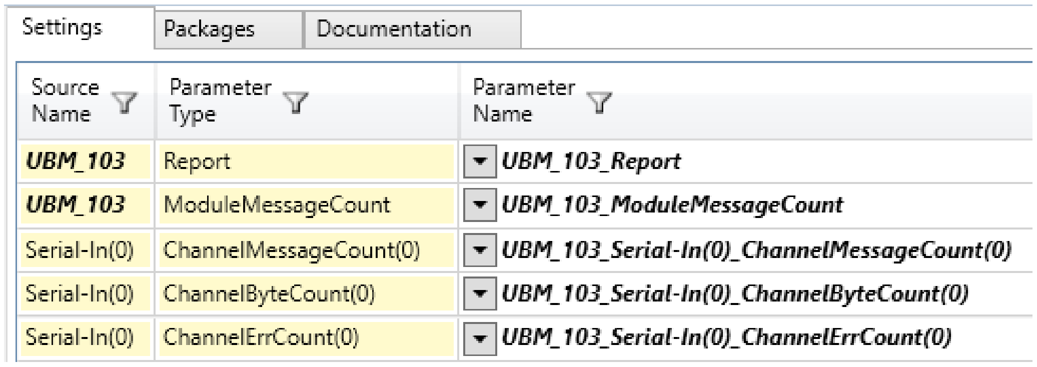
Figure 51-8: Baud Rate, Data Bits Per Word, and Parity settings

Setting	Description
Baud Rate (or bit rate)	Specifies the number of symbols (0s or 1s) transmitted per second. The KAD/UBM/103 can coherently parse messages at any bit rate in the range 300 bps to 1 Mbps.
Data Bits Per Word  Data Bits Per Word: 8 (options: 7, 8)	Can be either 7 or 8 bits. This does not include Start, Stop, or Parity bits.
Parity  Parity: No Parity (options: No Parity, Even Parity, Odd Parity)	Parity configures whether a parity bit is present in incoming data. It can be set to No Parity, Even Parity, or Odd Parity. **No Parity** means no parity bit is expected to be present in the incoming data words. **Even Parity** means a parity bit is expected to be present at the end of the incoming data word. This bit is set to either 1 or 0 so that the total number of 1s in the word adds up to an even number. Similarly, **Odd Parity** means this bit is set to either 1 or 0 so that the total number of 1s in the word adds up to an odd number.
Parity Check  Parity Check: Not checked (options: Not checked, Report failure, Report failure and offset)	Parity error detection can be enabled by choosing an option from the Parity Check drop-down list as shown in the following figure. The following three Parity Check options are available: • **Not checked** results in all parity errors being ignored. • **Report failure** results in a parity error being indicated in the parser block header and Report Word parameter. • **Report failure and offset** is a packetizer only option which reports the error in the packet and parser block headers and in the Report Word parameter, while also appending a 32-bit word to the packetizer parser block in question, which indicates the byte offset of the first error. An example of this setting is shown in Figure 51-18 on page 14.
Gap Between Messages  Gap Between Messages: 10	This represents the time gap (represented in characters) between consecutive characters required before starting a new message. The value is expressed in units of character periods at the configured bit rate. This setting can be used as both a packetizer and parser. When using as a packetizer, set to 0 to ignore all gaps and create a message filled with all incoming messages. Refer to “51.7 Packetizer operation” on page 11 for more information. When using as a parser, the message gap (also known as idle time or sync time) can be used to parse messages which do not have a start sequence and/or the start sequence appears within the message. Refer to “51.11.3 Message gap” on page 16 for more information.

Setting	Description
Programmable Termination  Programmable Termination; Disabled; Disabled; Enabled	This is an optional setting to enable an internal 120-ohm termination resistance. This termination is only active when the module power is powered on. Use wiring selectable termination instead if termination is required at all times. Note: Implementing wiring termination and enabling programmable termination by mistake reduces the value of the termination resistor by half, making termination less efficient. Refer to "51.11.1 Termination" on page 15 for more information.

51.5.3 Global parameters settings

Like most bus monitors, the KAD/UBM/103 has global parameters, which can be used for debugging. The main global parameters are Report word and several counters such as module, message, error and byte counters as described in the following table.



Source Name	Parameter Type	Parameter Name
UBM_103	Report	*UBM_103_Report*
UBM_103	ModuleMessageCount	*UBM_103_ModuleMessageCount*
Serial-In(0)	ChannelMessageCount(0)	*UBM_103_Serial-In(0)_ChannelMessageCount(0)*
Serial-In(0)	ChannelByteCount(0)	*UBM_103_Serial-In(0)_ChannelByteCount(0)*
Serial-In(0)	ChannelErrCount(0)	*UBM_103_Serial-In(0)_ChannelErrCount(0)*

Figure 51-9: Global parameters in DAS Studio 3 Settings tab

Setting	Description
Report	The Report word is a 16-bit register, which provides information regarding errors detected in the card and the bus. The Report word is recommended to be monitored as a debug register when abnormal conditions are detected. Refer to the *KAD/UBM/103* data sheet and "51.5.3 Global parameters settings" on page 6 for further information.
ModuleMessageCount	ModuleMessageCount increments by one each time the parser logic detects a complete message. Note that the module does not know which bytes form a message until it is parsed. This register counts how many messages the module has parsed by any of the channels.
ChannelMessageCount (0 to 15)	Increments by one each time the parser logic detects a complete message. Note that messages which are not defined in the parser are just considered bytes, which can be tracked with ChannelByteCount. Note: Do not confuse this register with MessageCount, which corresponds to a counter added to each message (see "51.6.4 Adding parameters to the package" on page 9).
ChannelByteCount (0 to 15)	Count the overall bytes received on this bus, regardless whether they are defined in the parser or not.
ChannelErrCount (0 to 15)	Count of errors detected on this bus. See "51.5.4 Errors" on page 7 for error definitions.

51.5.4 Errors

There are several errors reported by the KAD/UBM/103. As explained in the previous section, the Report word provides an indicator of the different errors detected in any of the busses, and ChannelErrCount provides an indicator of the number of errors detected on each bus. Additional packetizer headers contains an error flag Er (1 bit) and a 6 bits Error Code indicator.

The supported errors are:

- **Parity Error**: value 0x1 - This error can be ignored if Parity Check is set to *Not checked*.
- **Bad Stop Bit**: value 0x2 - Refer to “51.11.5 Bad Stop Bit” on page 16.
- **Too many data words**: value 0x4 - Stop character has not been found in 1024 characters. This error is not reported in the packetizer error code.

51.6 Serial Builder

For the following sections, use the Serial Builder application in DAS Studio 3. Refer to the “Builder application GUI overview” section of the *DAS Studio 3 User Manual* for a brief overview of navigating the application.

“51.6.1 Defining parsing rules” on page 7

“51.6.2 Parsing Mode” on page 8

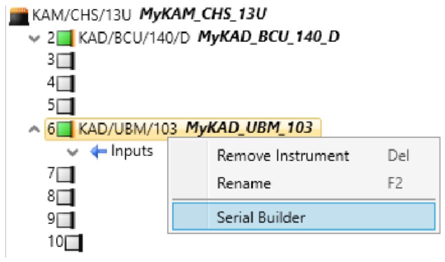
“51.6.3 Start/Stop Sequence format” on page 9

“51.6.4 Adding parameters to the package” on page 9

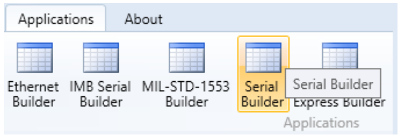
51.6.1 Defining parsing rules

After you have all channel settings configured, refer to the following to define rules to identify messages.

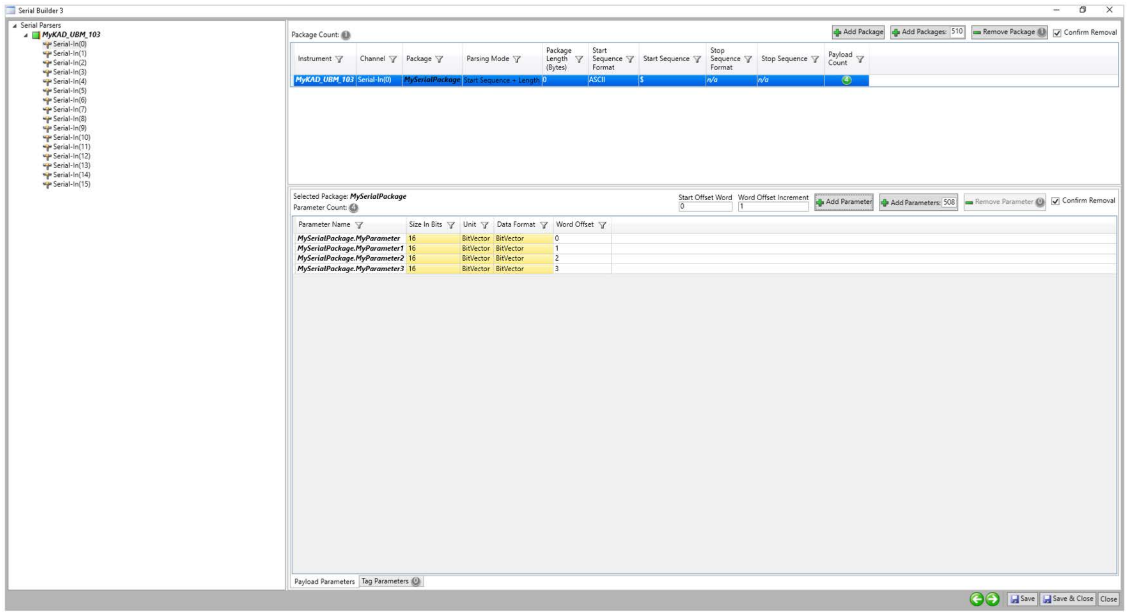
1. Do one of the following.
   - In the Navigator, right-click the KAD/UBM/103 module and then click **Serial Builder**.


KAM/CHS/13U MyKAM_CHS_13U
2 KAD/BCU/140/D MyKAD_BCU_140_D
3
4
5
6 KAD/UBM/103 MyKAD_UBM_103
Inputs
7
8
9
10
Remove Instrument Del
Rename F2
Serial Builder

- On the **Applications** tab click **Serial Builder**.



The **Serial Builder** application opens.



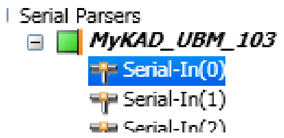
Package Count: 1

Instrument	Channel	Package	Parsing Mode	Package Length (Bytes)	Start Sequence Format	Start Sequence	Stop Sequence Format	Stop Sequence	Payload Count
MyKAD_UBM_103	Serial-In(0)	MySerialPackage	Start Sequence + Length	0	ASCII	$	n/a	n/a	4

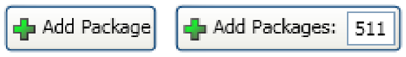
Selected Package: MySerialPackage
Parameter Count: 4
Start Offset Word: 0
Word Offset Increment: 1

Parameter Name	Size In Bits	Unit	Data Format	Word Offset
MySerialPackage.MyParameter	16	BitVector	BitVector	0
MySerialPackage.MyParameter1	16	BitVector	BitVector	1
MySerialPackage.MyParameter2	16	BitVector	BitVector	2
MySerialPackage.MyParameter3	16	BitVector	BitVector	3

2. In the Navigator (left pane), select the channel on the KAD/UBM/103 that you want to parse data off.



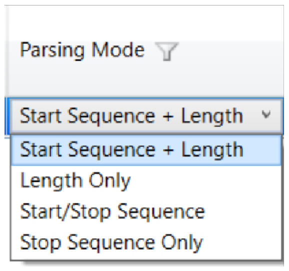
3. Click the **Add Package** button to add a single package. To add multiple packages (up to 511), click the **Add Packages** button (typing the number of packages in the field).



51.6.2 Parsing Mode

Now you must define the rules to identify or parse the desired message.

4. The first rule to define is **Parsing Mode**. In the **Parsing Mode** field, open the drop-down list.



Choose **Start Sequence+Length** if you know the start pattern of the message and the number of bytes of data that make up the entire message.

Choose **Length Only** if there is no unique start sequence and you know the number of bytes of the entire message. A message gap needs to be known for it. Refer to “51.11.3 Message gap” on page 16 for further information.

Choose **Start/Stop Sequence** if you know a fixed sequence of 1 to 8 characters that uniquely identify this message, and the stop character that indicates the end of the message.

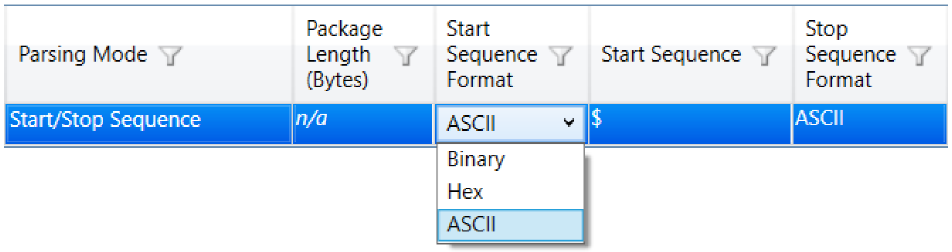
Choose **Stop Sequence Only** if there is no unique start sequence and you only know the stop character that indicates the end of the message. A message gap will need to be known for it. Refer to “51.11.3 Message gap” on page 16 for further information.

When **Start/Stop Sequence** is chosen, the **Package Length** field is not available and the relevant **Start/Stop Sequence** fields are made available.

51.6.3 Start/Stop Sequence format

Now you define the **Start Sequence Format** as **ASCII**, **Hex** or **Binary**.

The **Start Sequence** can be up to eight characters long. Binary and Hex can be used when wildcards are required to identify the message. See “51.11.4 Wildcard in Start Sequence” on page 16 for more information.



Parsing Mode	Package Length (Bytes)	Start Sequence Format	Start Sequence	Stop Sequence Format
Start/Stop Sequence	n/a	ASCII	$	ASCII

Figure 51-10: Start Sequence Format options

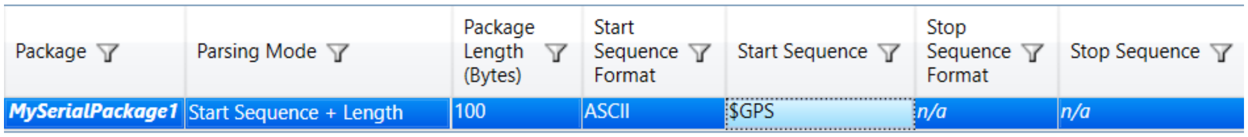
A common start sequence example is the one used with NMEA messages such as ASCII characters $GPSZDA.

Stop Sequence is a single byte used to identify the end of the message. By default, this is the ASCII character for line feed (\l).

A typical NMEA message ends with ASCII line feed (0xD) and stop sequence of ASCII Carriage Return (0x0A).

Similarly when **Stop Sequence Only** is selected, only the **Stop Sequence** field is available.

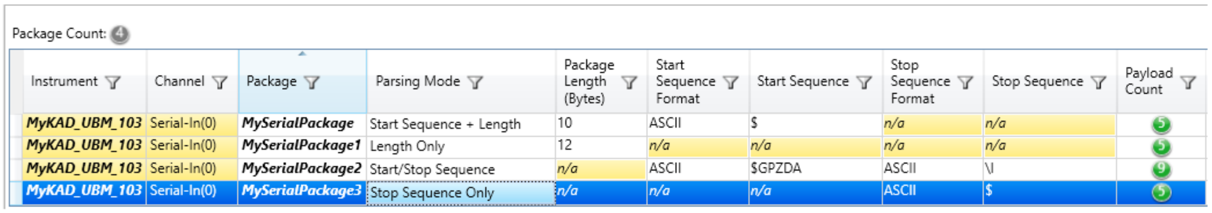
When **Start Sequence+Length** is chosen, the **Package Length** and **Start Sequenc**e fields are made available and the **Stop Sequence** field is not available.



Package	Parsing Mode	Package Length (Bytes)	Start Sequence Format	Start Sequence	Stop Sequence Format	Stop Sequence
MySerialPackage1	Start Sequence + Length	100	ASCII	$GPS	n/a	n/a

Figure 51-11: Start Sequence + length option

Similarly when **Length Only** is selected, only the **Package Lengt**h field is available.



Package Count: 4

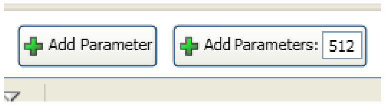
Instrument	Channel	Package	Parsing Mode	Package Length (Bytes)	Start Sequence Format	Start Sequence	Stop Sequence Format	Stop Sequence	Payload Count
MyKAD_UBM_103	Serial-In(0)	MySerialPackage	Start Sequence + Length	10	ASCII	$	n/a	n/a	5
MyKAD_UBM_103	Serial-In(0)	MySerialPackage1	Length Only	12	n/a	n/a	n/a	n/a	5
MyKAD_UBM_103	Serial-In(0)	MySerialPackage2	Start/Stop Sequence	n/a	ASCII	$GPZDA	ASCII	\l	9
MyKAD_UBM_103	Serial-In(0)	MySerialPackage3	Stop Sequence Only	n/a	n/a	n/a	ASCII	$	5

Figure 51-12: Example of the setup of each message type available in the KAD/UBM/103

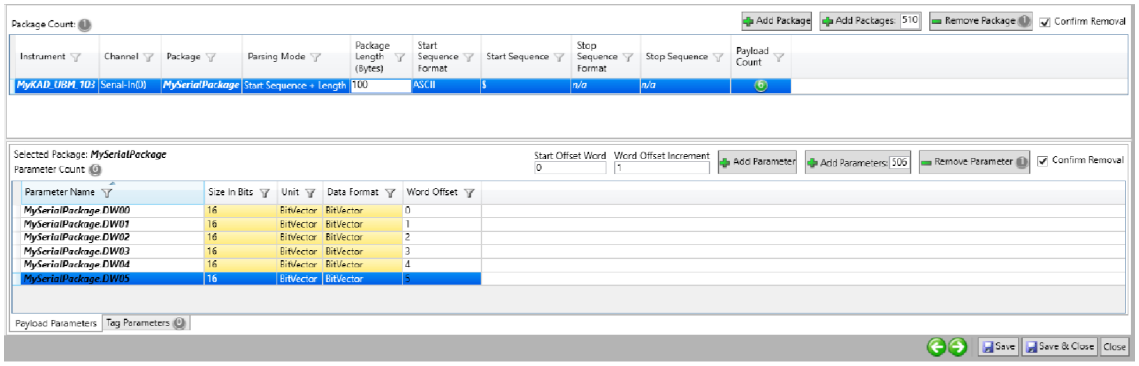
51.6.4 Adding parameters to the package

After you have defined rules to identify a message, refer to the following to select the number of bytes required to be parsed.

1. Click **Add Parameter** to add a single parameter. To add multiple parameters, click **Add Parameters** (typing the number of parameters in the field). Up to 512, 16-bit parameters can be defined for every message. This corresponds to a maximum of 1024 characters per message.



In the following example, 6 parameters are added.
Serial message with 6 data words starting from offset 0 and increasing by 1 position.



Package Count: 1

Add Package | Add Packages: 510 | Remove Package | ✓ Confirm Removal

Instrument	Channel	Package	Parsing Mode	Package Length (Bytes)	Start Sequence Format	Start Sequence	Stop Sequence Format	Stop Sequence	Payload Count
MyKAD_UBM_103	Serial-In(0)	*MySerialPackage*	Start Sequence + Length	100	ASCII	$	n/a	n/a	6

Selected Package: *MySerialPackage*
Parameter Count 6

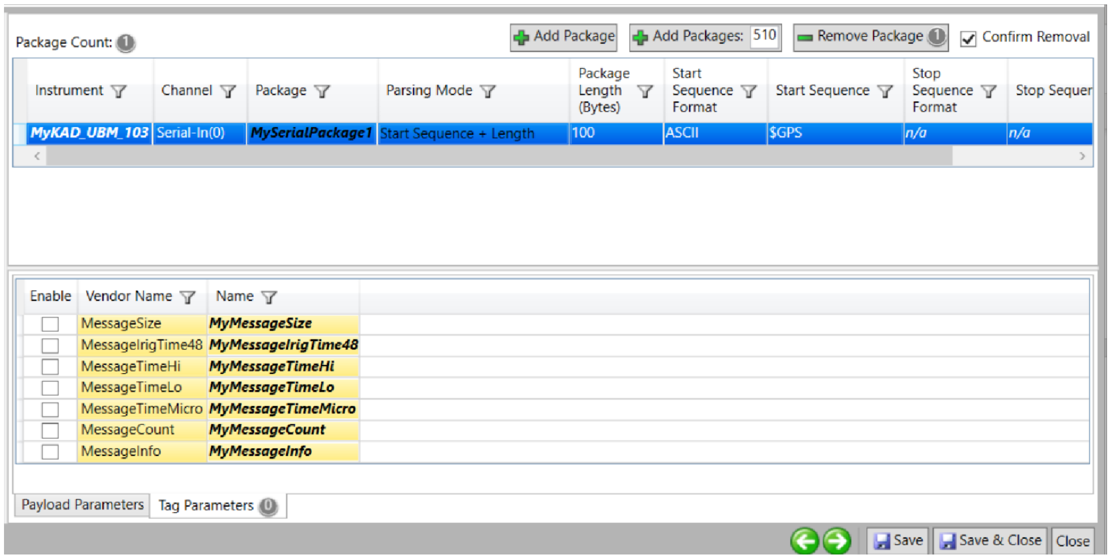
Start Offset Word 0 | Word Offset Increment 1 | Add Parameter | Add Parameters: 506 | Remove Parameter | ✓ Confirm Removal

Parameter Name	Size In Bits	Unit	Data Format	Word Offset
MySerialPackage.DW00	16	BitVector	BitVector	0
MySerialPackage.DW01	16	BitVector	BitVector	1
MySerialPackage.DW02	16	BitVector	BitVector	2
MySerialPackage.DW03	16	BitVector	BitVector	3
MySerialPackage.DW04	16	BitVector	BitVector	4
MySerialPackage.DW05	16	BitVector	BitVector	5

Payload Parameters | Tag Parameters 0

Save | Save & Close | Close

2. To tag a message, select the message and then click the **Tag Parameters** tab.



Package Count: 1

Add Package | Add Packages: 510 | Remove Package | ✓ Confirm Removal

Instrument	Channel	Package	Parsing Mode	Package Length (Bytes)	Start Sequence Format	Start Sequence	Stop Sequence Format	Stop Sequen
MyKAD_UBM_103	Serial-In(0)	*MySerialPackage1*	Start Sequence + Length	100	ASCII	$GPS	n/a	n/a

Enable	Vendor Name	Name
☐	MessageSize	*MyMessageSize*
☐	MessageIrigTime48	*MyMessageIrigTime48*
☐	MessageTimeHi	*MyMessageTimeHi*
☐	MessageTimeLo	*MyMessageTimeLo*
☐	MessageTimeMicro	*MyMessageTimeMicro*
☐	MessageCount	*MyMessageCount*
☐	MessageInfo	*MyMessageInfo*

Payload Parameters | Tag Parameters 0

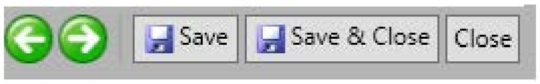
Save | Save & Close | Close

The tags associated with the message are described as follows:

Setting	Description
MessageIrigTime48	MessageIrigTime48 is a 48 bit register consisting of three 16 bits time register: TimeHi , TimeLo and TimeMicro. Represents the time stamp of a valid parsed message
TimeHi, TimeLo and TimeMicro	Same information as MessageIrigTime48 but split in three 16 bit register. Note: these registers are implemented to provide compatibility with legacy systems
MessageSize	Number of received bytes including start bytes.
MessageCount	Counter of the received messages , however this counter might not increase smoothly. As explained in section “51.5.3 Global parameters settings” in this document, the ModuleMessageCount is the only counter in the module counting messages received in all channels of the module. ModuleMessageCount stores the current value of that counter in the slot with each message , therefore ModuleMessageCount increases smoothly. MessageCount indicates where, in the arrival order, the particular message was received. For this reason consecutive messages on a single bus may not have consecutive values as messages on other channels may have been received, however it increase. Individual MessageCount from different channels can be correlated with ModuleMessageCount
MessageInfo	Stale/skipped indication for each parsed message. These flags indicate whether messages are repeated or lost, that is, oversampling or undersampling situations respectively

Refer to the *KAD/UBM/103* data sheet for further information regarding these additional tags.

3. To save your changes and close **Serial Builder**, click **Save & Close**.



When package building in Serial Builder is complete, these parameters become available to be placed in a PCM stream or placed packet.

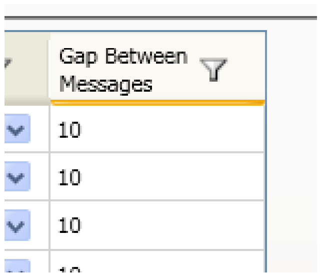
51.7 Packetizer operation

Independently of the parser, when packetizer is enabled, an iNET-X packet stream is generated for each channel. All received bytes are encapsulated in an iNET-X parser-aligned payload structure. The programmable message gap allows the module to split the incoming bytes into shorter timestamped sequences. A block header attached to each sequence stores the channel index, length, and the time each message is received. These parser-aligned packets may be transmitted aperiodically to optimize network bandwidth utilization and memory usage when recording serial traffic.

There are many settings available to configure or tune packetizer behavior.

While the structure of the packetizer packets are what we refer to as parser-aligned packets, the KAD/UBM/103 relies on gaps between the data to identify the start/end of the parser-aligned blocks. The size of the gap between messages can be set on a channel-by-channel basis. The Gap Between Messages setting (see the following figure) is expressed in characters, and ranges from 0 to 10,000 characters.

NOTE: To view the screen shots shown in this section in DAS Studio 3, ensure the KAD/UBM/103 module is in context and the Settings tab is selected.

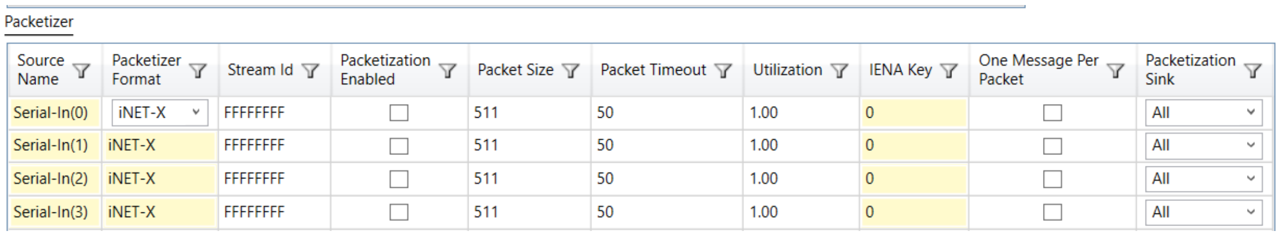


Gap Between Messages
10
10
10

Figure 51-13: Gap Between Messages setting from Settings tab

Setting 0 characters results in all gaps being ignored and the packetizer packets being filled with bytes of data until the packet is either full or times out; thereafter the packet is transmitted.

The other available packetizer settings are shown in the following figure and described in the table that follows.



Packetizer

Source Name	Packetizer Format	Stream Id	Packetization Enabled	Packet Size	Packet Timeout	Utilization	IENA Key	One Message Per Packet	Packetization Sink
Serial-In(0)	iNET-X	FFFFFFFF	☐	511	50	1.00	0	☐	All
Serial-In(1)	iNET-X	FFFFFFFF	☐	511	50	1.00	0	☐	All
Serial-In(2)	iNET-X	FFFFFFFF	☐	511	50	1.00	0	☐	All
Serial-In(3)	iNET-X	FFFFFFFF	☐	511	50	1.00	0	☐	All

Figure 51-14: Packetizer settings

Setting	Description
Source Name	Channel to packetize.
Packetizer Format	iNET-X or IENA-iNET-X hybrid. This setting defines the packetizer format of all channels. Note, IENA-iNET-X hybrid is a non-standard format.

Setting	Description
Stream ID	iNET-X stream identifier for selected channel if a packet is generated via the assertion of Packetization Enabled. This is a conditional setting and is only active when the Packetizer Format is set to iNET-X.
Packetization Enabled	Enables the transmission of a packetizer packet containing the contents of this channel if an packetizer transmitter or memory module is present in the chassis. DAS Studio 3 automatically creates a packetizer packet after verification/programming.
Packet Size	The number of words in the packet buffer, ranges from 200 words to 511 words. The default value is 511 words; reducing this value results in smaller and therefore generally more frequent packets.
Packet Timeout	The timeout in milliseconds before a packet is generated if insufficient messages have been received to reach the Packet Size. Packets generated due to Packet Timeout are tagged in the iNET-X header. The Packet Timeout ranges from 10 ms to 999 ms (default value is 50 ms). Reducing this value results in more frequent and generally smaller packets. Increasing the value results in less frequent, but generally bigger packets.
Utilization	The setting is a way for the user to schedule fewer packets per second for situations where they know that traffic on the bus is less frequent. It is expressed as a decimal number in the range 0.0 to 1.0 where 1.0 = 100% bus utilization. As an example for an input bit-rate configured to 1 Mbps, the KAD/UBM/103 schedules 256 packets per second with 100% bus utilization. If set to 50% bus utilization, the scheduler can drop this as low as 128 packets per second. This setting can be considered as an advanced setting and can be useful on a chassis populated with multiple packetizer channels enabled for which the compiler has difficulties to schedule all the traffic in the KAM-500 backplane. Reducing the utilization on a channel alleviates the backplane traffic.
One Message Per Packet	This setting determines whether multiple or a single message should be included in a packet. When enabled, a packet is generated after a valid message has been detected. Subsequent messages are placed in separate packets. This simplifies message extraction as there is only one message per packet.
Packetization Sink	Selects which modules the packetizer package is sent to for transmission or storage. The choices are *Controller only*, *All slots* or *Slot* in which a sink module that supports packetizer logging such as KAM/MEM/113 is placed. For example, on a system consisting of KAD/BCU/140/D controller on J2, KAD/UBM/103 on J3, and KAM/MEM/113 on J4, setting *Slot 4* creates packetizer packets on the KAM/MEM/113 only and setting *All* creates packets on both KAD/BCU/140/D and KAM/MEM/113.

Conversely, if a single message continues past the end of one packet with no gap, the succeeding bytes are packetized in a new parser block in the next packet. A bit in the header of the first parser block in the following packet, is set to indicate that this block is a continuation of the message in the final block of the previous packet. The location of this continuation bit is marked **Cn** (Continuation Indicator) in the example parser blocks shown in Figure 51-17 on page 14.

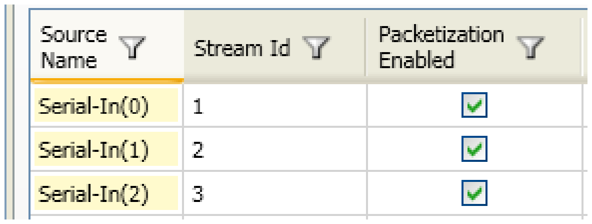
For further information regarding iNET-X Placed packets used by the packetizer refer to *TEC/NOT/067 - IENA and iNET-X packet payload formats*. Additionally the *KAD/UBM/103* data sheet provides several examples of packetizer parser blocks.

51.8 Parity Check– Report failure and offset

This is a packetizer only option. See *Parity Check* in Figure 51-8 on page 5 and the description that follows.

51.9 Enabling packetizer

To turn on packetizer operation on any channel, define a unique stream ID for that channel and then select the Packetization Enabled check box for that channel as shown in the following figure. The packetizer is enabled the next time the module is programmed.



Source Name	Stream Id	Packetization Enabled
Serial-In(0)	1	✓
Serial-In(1)	2	✓
Serial-In(2)	3	✓

Figure 51-15: Packetization Enabled setting

51.10 Packetizer packet format (parser aligned)

The generalized iNET-X payload structure for parser-aligned packets is shown in the following figure and examples of parser block formats are shown in Figure 51-17 on page 14. A parser block is created for each logical message, based on the gaps between sequences of bytes. A new parser block is created at the start of each iNET-X packet and also when the time gap between successive bytes exceeds a programmable threshold.

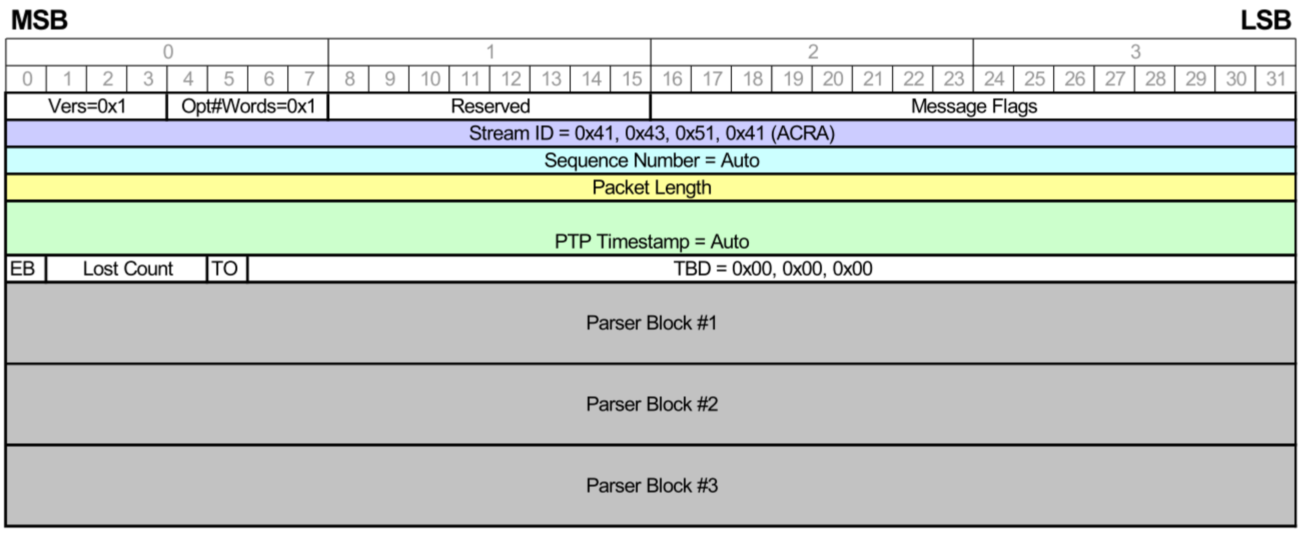
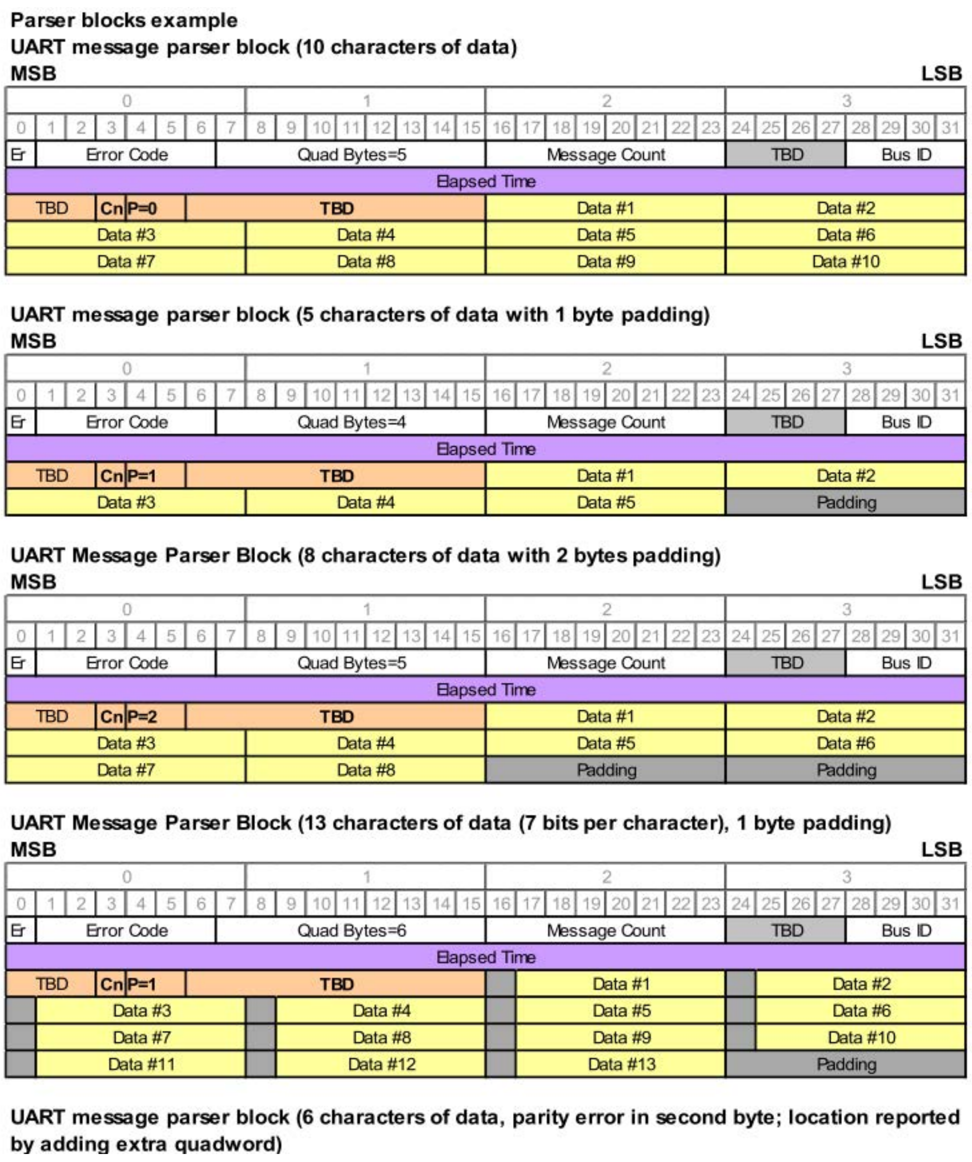


Figure 51-16: Generalized parser-aligned iNET-X packet



Parser blocks example

UART message parser block (10 characters of data)

MSB **LSB**

0								1								2								3							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Er	Error Code						Quad Bytes=5									Message Count								TBD				Bus ID			
Elapsed Time																															
TBD			Cn	P=0		TBD										Data #1								Data #2							
Data #3								Data #4								Data #5								Data #6							
Data #7								Data #8								Data #9								Data #10							

UART message parser block (5 characters of data with 1 byte padding)

MSB **LSB**

0								1								2								3							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Er	Error Code						Quad Bytes=4									Message Count								TBD				Bus ID			
Elapsed Time																															
TBD			Cn	P=1		TBD										Data #1								Data #2							
Data #3								Data #4								Data #5								Padding							

UART Message Parser Block (8 characters of data with 2 bytes padding)

MSB **LSB**

0								1								2								3							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Er	Error Code						Quad Bytes=5									Message Count								TBD				Bus ID			
Elapsed Time																															
TBD			Cn	P=2		TBD										Data #1								Data #2							
Data #3								Data #4								Data #5								Data #6							
Data #7								Data #8								Padding								Padding							

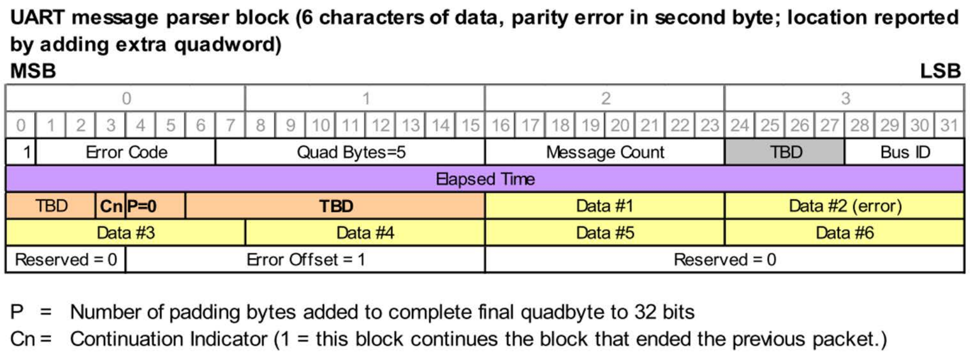
UART Message Parser Block (13 characters of data (7 bits per character), 1 byte padding)

MSB **LSB**

0								1								2								3							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Er	Error Code						Quad Bytes=6									Message Count								TBD				Bus ID			
Elapsed Time																															
TBD			Cn	P=1		TBD										▨	Data #1							▨	Data #2						
▨	Data #3							▨	Data #4							▨	Data #5							▨	Data #6						
▨	Data #7							▨	Data #8							▨	Data #9							▨	Data #10						
▨	Data #11							▨	Data #12							▨	Data #13							Padding							

UART message parser block (6 characters of data, parity error in second byte; location reported by adding extra quadword)

Figure 51-17: Parser block formats used in packetizer



UART message parser block (6 characters of data, parity error in second byte; location reported by adding extra quadword)

MSB **LSB**

0								1								2								3							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
1	Error Code						Quad Bytes=5									Message Count								TBD				Bus ID			
Elapsed Time																															
TBD			Cn	P=0		TBD										Data #1								Data #2 (error)							
Data #3								Data #4								Data #5								Data #6							
Reserved = 0				Error Offset = 1												Reserved = 0															

P	=	Number of padding bytes added to complete final quadbyte to 32 bits
Cn	=	Continuation Indicator (1 = this block continues the block that ended the previous packet.)

Figure 51-18: Parser block formats used in packetizer with option “Report failure and offset” enabled

51.11 Appendix

51.11.1 Termination

For an RS422/485 bus, the transmission line should be terminated at the last transceiver on the line (bus). Transmission line termination schemes is a topic of discussion beyond the scope of this technical note. The section gives a basic overview of the termination schemes.

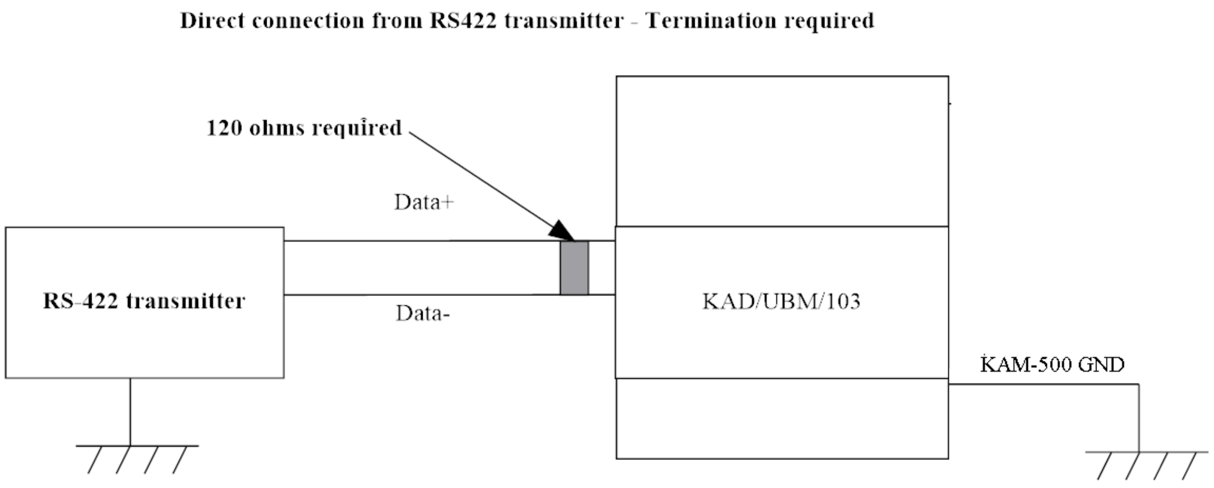


Figure 51-19: Example of termination required on the KAD/UBM/103

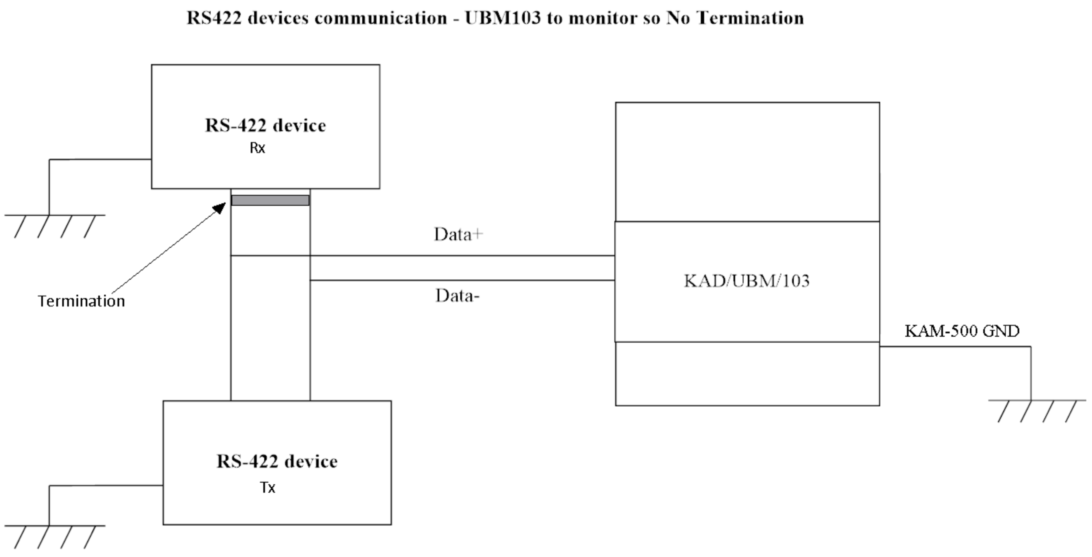


Figure 51-20: Example of no termination required on the KAD/UBM/103

The KAD/UBM/103 can be programmed to have termination. This programmable termination is not active when the power is off, that is, the bus is not terminated if the KAM-500 is powered off. If permanent termination is required, then external termination should be used. Refer to the Getting the most from section of the *KAD/UBM/103* data sheet.

51.11.2 Grounding

Grounding schemes is a topic of discussion beyond the scope of this technical note. Unlike RS-232, RS-422, and RS-485 transmission is differential. The previous two figures show a system configuration presented without a separate ground wire. The basic rules for electronic circuits still require a clean ground connection to ensure error-free communication between drivers (Tx) and receivers (Rx). For further information refer to *TEC/NOT/063 — Grounding and shielding of the Acra KAM-500*.

51.11.3 Message gap

The gap between messages can also be used for parsing. The idea is the same as the Idle Time/Sync Interval functionality on the KAD/UAR/102.The gap from the KAD/UBM/103 is expressed in units of character period at the configured bit-rate.

This gap needs to be defined if the start sequence appears inside the message or there is no unique start sequence.

The example below shows a chain of characters. The KAD/UBM/103 is programmed with AB as a Start Sequence and with a length of 10 bytes for the Stop Sequence, which means DW0 to DW4 are required. The Parser Data Endianness is set to First byte at low end of word.

If you set the gap at 0—because only the start sequence is used as a parser slot—the module may start parsing when it encounters the start sequence AB inside the message.

So the KAD/UBM/103 could output DW0 = AB, DW1 = CD, DW2 = EF then all the other DW will be random data or data previously stored in RAM, the next instance of the data word, will be DW0 = AB, DW1 = G\n then DW2 will be from the next instance, that is, AB, DW3= DD…

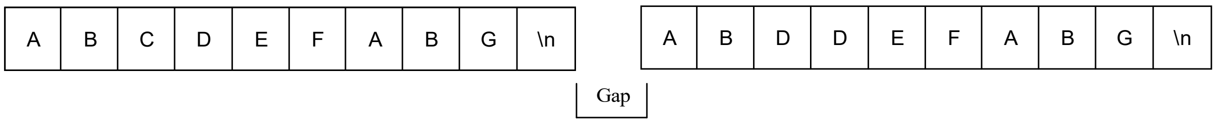
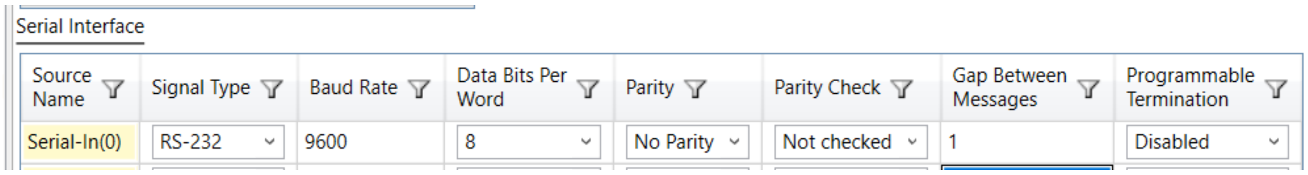


Figure 51-21: Example of gap between messages

To parse the whole message correctly, the KAD/UBM/103 gap between messages needs to be programmed with at least 1 byte. The module then outputs DW0 = AB, DW1 = CD, DW2 = EF, DW3= AB, DW4= G\n.



Serial Interface

Source Name	Signal Type	Baud Rate	Data Bits Per Word	Parity	Parity Check	Gap Between Messages	Programmable Termination
Serial-In(0)	RS-232	9600	8	No Parity	Not checked	1	Disabled

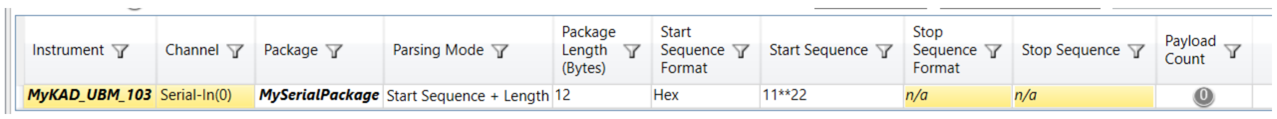
Figure 51-22: Example of gap between messages being set to 1 character in DAS Studio 3

As explained in “51.6.2 Parsing Mode” on page 8, parsing modes *Length Only* and *Stop Sequence Only* can use *Message gap* when the incoming message has no unique start sequence or the start sequence is present within the message. It is recommended to use these advanced parsing options with prior knowledge of the incoming message. Care must be taken in order to ensure the programmed message gap is not greater than the time of the next incoming message. Also power up conditions should be considered as incorrect parsing could occur when the module is powered up in the middle of the reception of a message.

NOTE: The KAD/UBM/105 does not support message gap for parsers.

51.11.4 Wildcard in Start Sequence

Serial Builder supports wildcard symbols '*' in the Start Sequence. When using Hex format for Start Sequence, each wildcard represents 4 bits (nibble), therefore two wild cards symbols “**” are required to represent one character boundary (1 byte). When using Binary format for Start Sequence, each bit can be wildcarded on a bit-to-bit basis.



Instrument	Channel	Package	Parsing Mode	Package Length (Bytes)	Start Sequence Format	Start Sequence	Stop Sequence Format	Stop Sequence	Payload Count
MyKAD_UBM_103	Serial-In(0)	***MySerialPackage***	Start Sequence + Length	12	Hex	11**22	*n/a*	*n/a*	0

Figure 51-23: Serial Builder - Example of wildcard used on a 3-character start sequence using Hex format

51.11.5 Bad Stop Bit

The module detects a Bad Stop Bit if it’s not as expected, see Figure 51-4 on page 2.

For example:

• No parity is configured and a message is received that was transmitted with a Parity bit. In this case the parity bit is decoded as a bad stop bit.

• Parity is configured and a message is received that was transmitted with no Parity bit. In this case since no Parity bit was sent, the module decodes the received Stop bit as an invalid Parity bit and then expects a Stop bit but never receives it and thus reports it as a Bad Stop Bit.

51.11.6 Recommended reading

To better understand this paper, read the following documents.

Table 51-1: Data sheets

Document	Description
KAD/UAR/102/C	RS-232/RS-422/RS-485 universal asynchronous parser and snarfer - 4ch
KAD/UBM/103	RS-232, RS-422 or RS-485 serial bus parser/packetizer - 16ch

Table 51-2: Data sheets

Document	Description
TEC/NOT/062	Using the KAD/UAR/102
TEC/NOT/063	Grounding and shielding of the Acra KAM-500
TEC/NOT/067	IENA and iNET-X packet payload formats

Table 51-3: User manual

Document	Description
DOC/MAN/030	DAS Studio 3 User Manual

This page is intentionally blank