ARINC 429 Decoder, Users Manual

Started January 2013, last update August 2019

Contents

List of Figures

1 Introduction

Since its first release, countless improvements have been made to the ARINC 429 Decoder. The vast majority of those pertain to the Symbolic Decoding, driven by the User Label Definition files. The system has become clearer, and easier to use for those wishing to harness the power of symbolic decode. We gladly thank E. Schütz formerly at SRTechnics, A. Netz and P. Lemberger at Avionik Straubing, D. Michaux at the University of Bordeaux, B. Morel and C. Wittwer at Ruag, A. Therry at Zodiac, F. Raimondi at Teledyne LeCroy, Simon Mccormick, at GB - Airframe Systems. All of them have contributed through their help and comments to the improvements of the tool.

2 The User Interface

2.1 Overview of the tabs offered by the ARINC Decoder

The User Interface of the ARINC 429 Decoder consists of 4 Tabs under the Decoder. Each one of the Tabs will be described in detail in the following pages. An additional Tab is located under the Zoom controls whenever a decoded Zoom is in use.

Figure 1 Tabs overview for ARINC 429

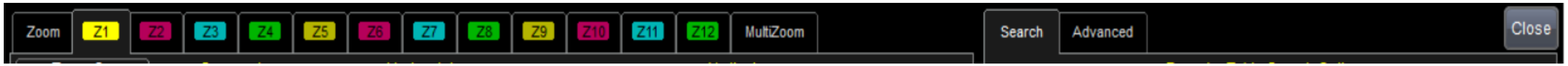


Figure 2 ARINC 429 Search tab, when Zoom focused on Decoded Trace

As in all most other decoders, there is no strict order in which to visit the tabs, **except for setting the bitrate** in the Basic tab. Once the decoding is correct, all the controls might be used in any order. Experience shows that the usage modes are as diverse as the users. The problem at hand usually dictates the usage, and the course of action. This document will attempt to convey usage hints in connection with the features described.

Also note that the common decode infrastructure such as the “Configure Table”, “Zoom” and the “Export Table” buttons at the bottom of the Decode Setup Tab are documented in the Teledyne LeCroy Serial Data Manual. The “Table filtering” by columns criterions are also documented in general manuals since they are used by all decoders.

2.2 The Basic Tab

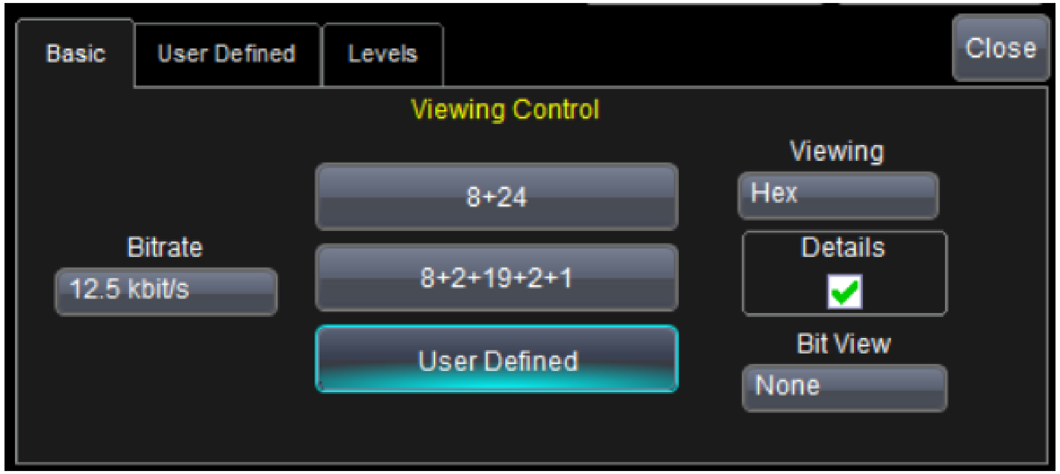


Figure 3 The Basic Tab

The Basic Tab offers the fundamental controls driving the decoding algorithm. Before looking at any other functionality, the Bitrate needs to be set correctly. In many cases the system user will know the bitrate and introduce it in the corresponding field. As ARINC predominantly uses 12.5 kbits/s or 100 kbit/s it is usually a simple choice. If there are any doubts on the bitrate of the ARINC stream take a moment to measure it, with the following simple method.

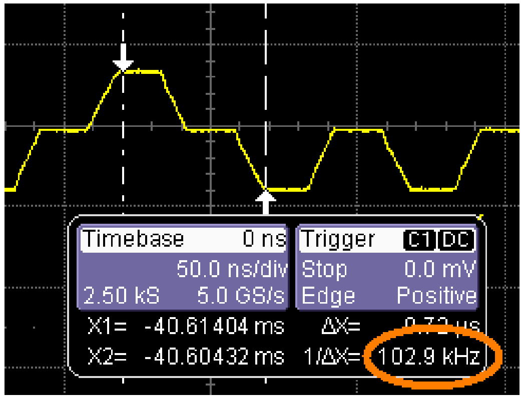


Figure 4 Measuring the Signal bitrate using the Time cursors

As the Bitrate governs the decoding, it is important to have it right from the onset. The simplest method is to use the Horizontal Relative Cursors. By positioning each cursor on equivalent positions of 2 consecutive bits, the 1/delta X read out will yield the Bitrate. Here we chose to measure on the beginning of each bit plateau. The readout of 102.9 KHz can be rounded to 100 KHz and is good enough for the decoding algorithm.

Once the Bitrate is selected, we will usually start with the 8+24 bit Decoding, until we have reached the conviction that our stream is decoded correctly. More in-depth explanations of the 3 Decode Modes will be provided further down in this document.

The **Viewing Mode is Hexadecimal or Decimal**. It governs some of the rendering both in the table and the annotation. Usually the task at hand will dictate the Viewing Mode. Note that it can be switched at any time during a session.

The Details check box allows the rendering of low-level details such as hexadecimal of BCD digits. Its importance also depends on the intent of the work. In some cases, it is necessary to track down the origin of a value to the individual BPRZ bits of the signal.

The **Bit View None, Bit Index, Bit State** provides an additional level of details when necessary.

When Bit Index is selected, individual bits are shown, numbered from 0 through 31 on every message.

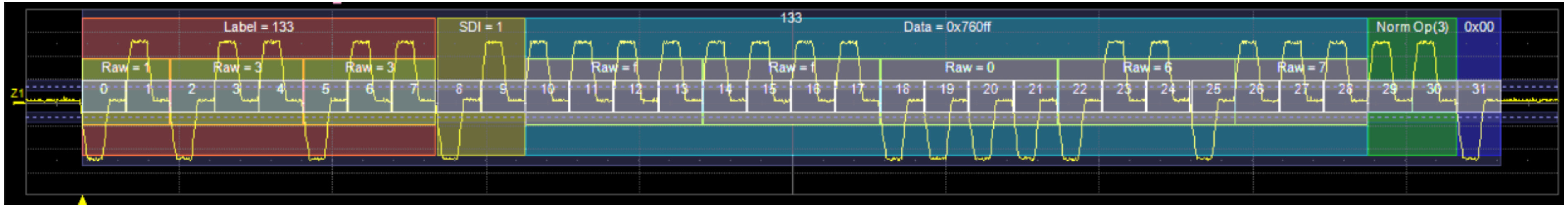


Figure 5 Bit Index annotation

When Bit State is selected, individual bits are shown, annotated with their value, 0 or 1, on every message

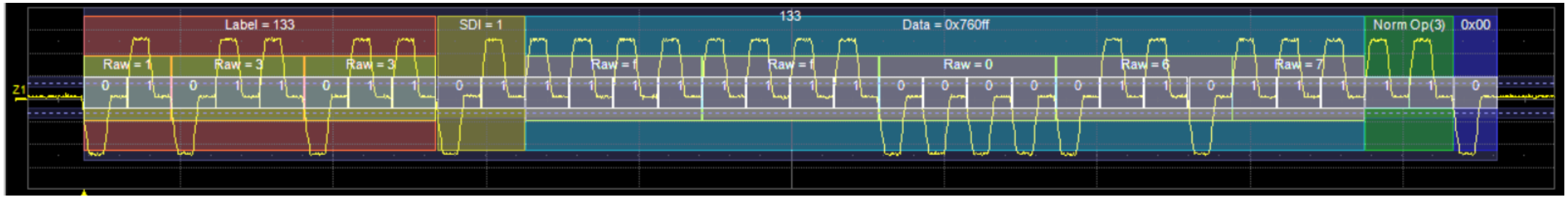


Figure 6 Bit State annotation

The modes “Viewing”, “Details” and “Bit View” can be set independently of the main “Viewing Control” explained above.

At this stage a **word of caution on the signal polarity.** ARINC 429 is usually transmitted over a twisted pair. While it is highly recommended to use a differential probe to measure ARINC 429 signals, it is possible to use a single ended probe on the Positive or Negative lines of the pair. In this case the signal appears either normal or inverted, without common mode rejection. **If, by accident, the decoder input is using the inverted signal,** all Label value will appear complemented to 377. In fact, everything will look right, even the Parity bit, but none of the Labels expected for the Equipment will appear! It is enough to switch the probe to the other wire to get the right information contents. If for any reason the positive signal cannot be probed, it is also easy to check the “Invert” control in the channel dialog. The “Invert” control numerically flips the signal in the channel and avoids having to electrically invert it.

Note that the inversion of polarity can also occur when using a differential probe whose inputs leads have been inadvertently swapped. In this case it is recommended to swap the leads and verify that the pair polarity is correct. It goes without saying that the polarity of the line on the aircraft is essential.

2.3 The User Defined Tab (for using ULDF files)

The User Defined Tab contains all of the necessary controls when working in User Defined Mode. The contents are explained here, but refer to Appendix A for all of the detailed explanation on Terminology, concepts and Syntax of the ULDF files.

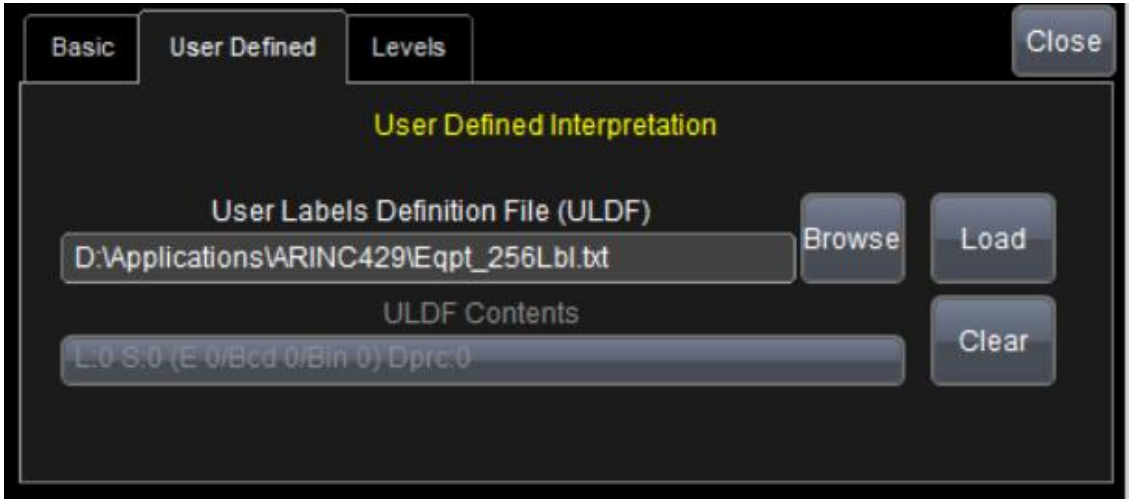


Figure 7 The User Defined Tab and its controls

The "User Defined" tab contains the File Selection Widget necessary to pick a ULDF. The primary modus operandi is to have a Dedicated File for each ARINC 429 Equipment. When a given channel is connected to the Equipment and Decoded, this File can be selected here and loaded. When the “User Defined” mode is selected in the Basic tab, all of the messages will be interpreted based on the ULDF.

The Browse button leads to the File Picker. The Load button reloads the file when it has been edited in parallel. When creating the ULDF files, it is convenient to proceed step by step to verify the correct decoding while adding Signal to the ULDF.

The Clear button Clears the internal Database of USigD. When the Database is cleared, all of the Labels will appear with only Label and Parity bit decoded.

The ULDF Contents box reflects the currently loaded database. It shows the total number of Labels in the Database, as well as the total number of Signals. It also shows the number of Signals of each type, E stands for Enumerated, BCD and Binaries. This control is grayed out because its values are not user selectable, but rather computed by the parsing algorithm of the ULDF file.

2.4 The Filter Tab (replaced by Table Filter)

Figure 8 The obsolete Filter Tab and its actions when using "Only Show Listed Labels"

~~The Filter gains importance when looking at long message streams. It is common that only a few messages require attention, typically when their contents are under scrutiny. Then, all other messages clutter the viewing and can be left out without impairing the debug process.~~

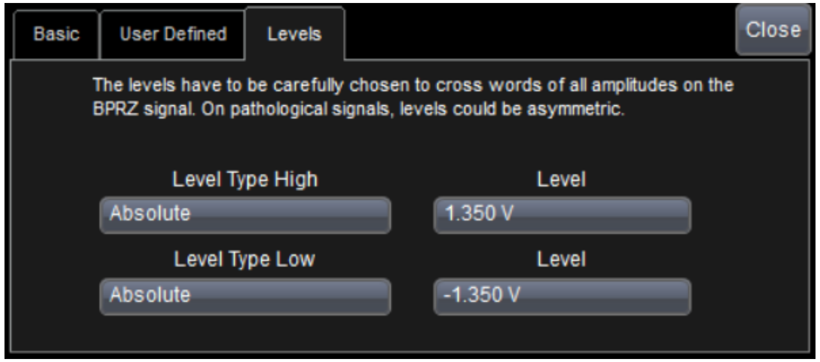
~~The Figure shows the **"Only Show Listed Labels"** mode with one Label values, 350 entered. The resulting display shows the sparsed stream and table.~~

~~Conversely, "**Show All Labels Except Those Listed"** applies the opposite logic. This method is useful when some messages have a higher recurrence (shorter time interval) on the bus. Here the filter unclutters the view, especially when the frequent messages are not the focus of the investigation.~~

~~Also note that applying a filter increase the decoding performance, as well as the export performance.~~

This tab has been superseded by a general infrastructure called "Table Filter" available for all protocols. Just like in Excel, the Decode Table can be filtered by applying Filter Criterions to the column. The equivalent functionality to the obsolete filter tab is achieved by filtering the Label column. **Appendix B describes the functionality in more details on a specific example, using various filtering operators.**

2.5 The Level Tab



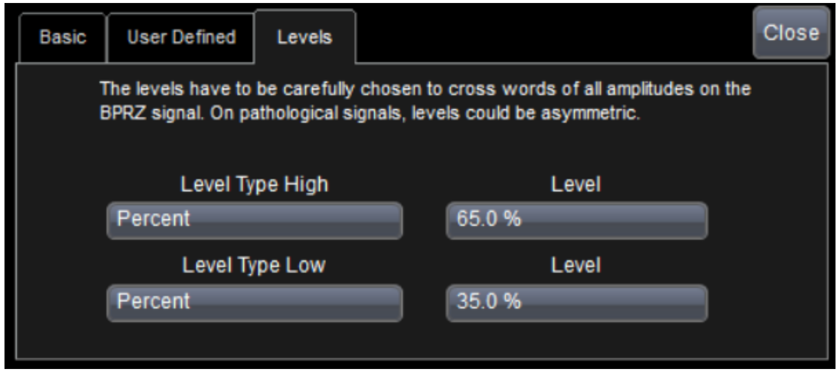


Figure 9 Level Tab Controls, in Absolute or Percent of Amplitude

The Level Tab contains 6 controls out of which only 4 are visible at any point in time. This is since the Levels can be specified either in Percent or in Volts, with the corresponding values being shown.

The Level selection is largely automated since by default both levels are set to be in percent of the amplitude. This shortens the initial learning curve for the tool, but it is important to make sure the levels are set correctly to get a correct decoding. The levels become very important when dealing with pathological signals.

Also note that Levels High and Low can be specified in mixed ways, i.e. High in Percent and Low in Absolute values

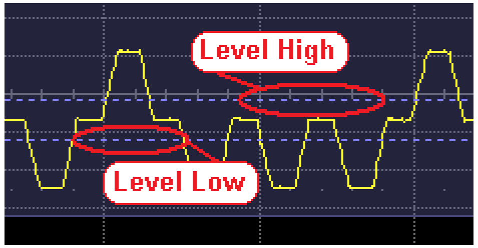


Figure 10 Looking at the levels in detail.

By default, the levels are set to be at 35% and 65 % of full amplitude on the BPRZ signal. In most cases, this is a perfect choice for ARINC 429 and immediately leads to correct decode. However, this only works if the signal is clean, as in the Figure, and the levels do not cross the Baseline noise or the Plateau noise. The following image will look at an example where the levels must be tuned manually.

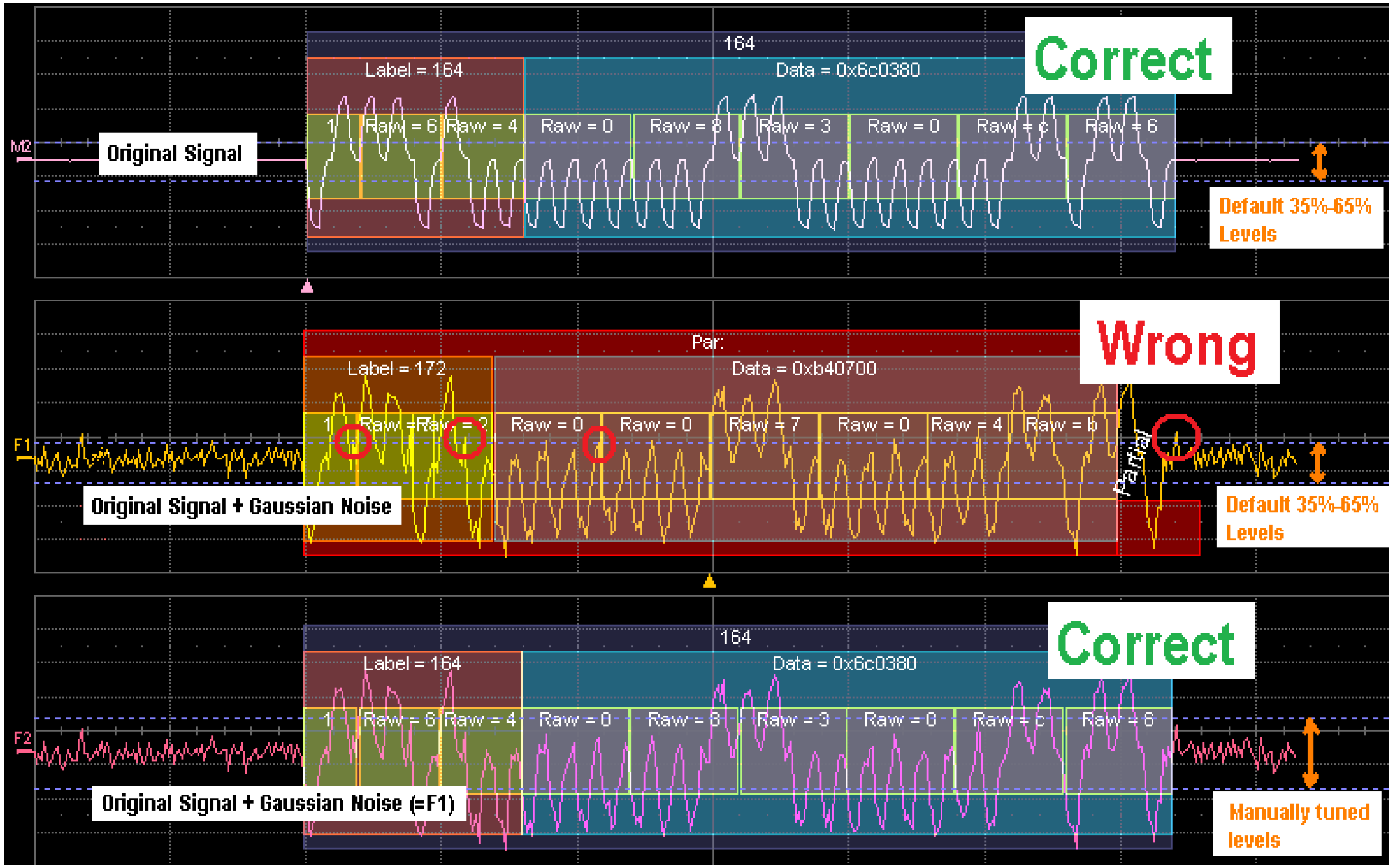


Figure 11 Rescuing a troubled Signal with manually tuned Levels

This is a real-world trace. Signals can be noisy, unbalanced, or corrupted in many other ways. In this case we would have to manually tune the levels to decode. In some cases, it is easier to use the Absolute level setting, in Volts. It is also possible to have one level in % and the other one Absolute.

This example shows 3 signals: M2, F1 and F2. F1 and F2 are identical traces. Both are based on M2, with the same Gaussian noise added. If you look closely at the images you will see that F2 decodes correctly (164/0x6c0380) thanks to the manually tuned levels. F1 is incorrect because the levels cross into the noise, where indicated by the red violation circles.

2.6 The Search Tab

Figure 10 The Search Tab when used on a Decoder

The search tab is available when a Zoom is connected to a decoded Trace. In this situation the normal Zoom controls are augmented by the Search Controls, as show in the Figure above.

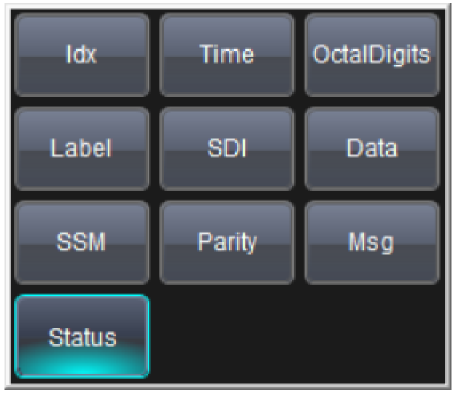
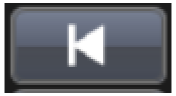
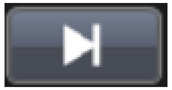
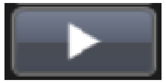
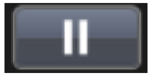


Figure 12 The Column to Search Selection

The "Search for" selection shows all the columns generated by the ARINC 429 decoder.

The set of single arrow buttons  and  allow moving from field to field, with or without added condition.

The set of double arrow buttons  and  jump to the beginning and end of the record.

The set of play/stop buttons    start/stop scrolling from the current position in either direction.

The “Use Value” box allows a search by value on the selected columns. The search criterion is column type sensitive, which is reflected by the type of value (decimal, hexadecimal, text and float). The search will focus on the syntactic element chosen under “Search for”, i.e. the SSM bits.

The “Show frame” button allows to search on a syntactic element shorter then the Frame, but still results in showing the entire Frame as a result.

Note that this behavior is common to all protocols within the LeCroy Decode Framework, which explains the usage of generic terms such as “Frame”

3 The Decode Modes

3.1 The 8+24 bit Decode Mode

The simplest of all modes (8bits +24bits) displays the Label in decimal and the data field in hexadecimal. Nibbles are swapped since they are encoded LSN first and nibbles are encoder lsb first.

Figure 13 Signal decoded in the 8+24 bits mode

This mode is very useful when starting the work, to quickly verify that each Label indeed contains 32 bits. For example, in the image above it is immediately clear that the Label contains 8 bits, interpreted as 3 BCD digits, and the Data 24 bits grouped in 6 hexadecimal digits of 4 bits.

3.2 The 8+2+19+2+1 bit Decode mode

The second hexadecimal mode (8+2+19+2+1) also decodes in hexadecimal but separates the SDI, SSM and parity from the data bits. The SDI and SSM fields can also be represented in the Symbolic Decode Mode with more flexibility.

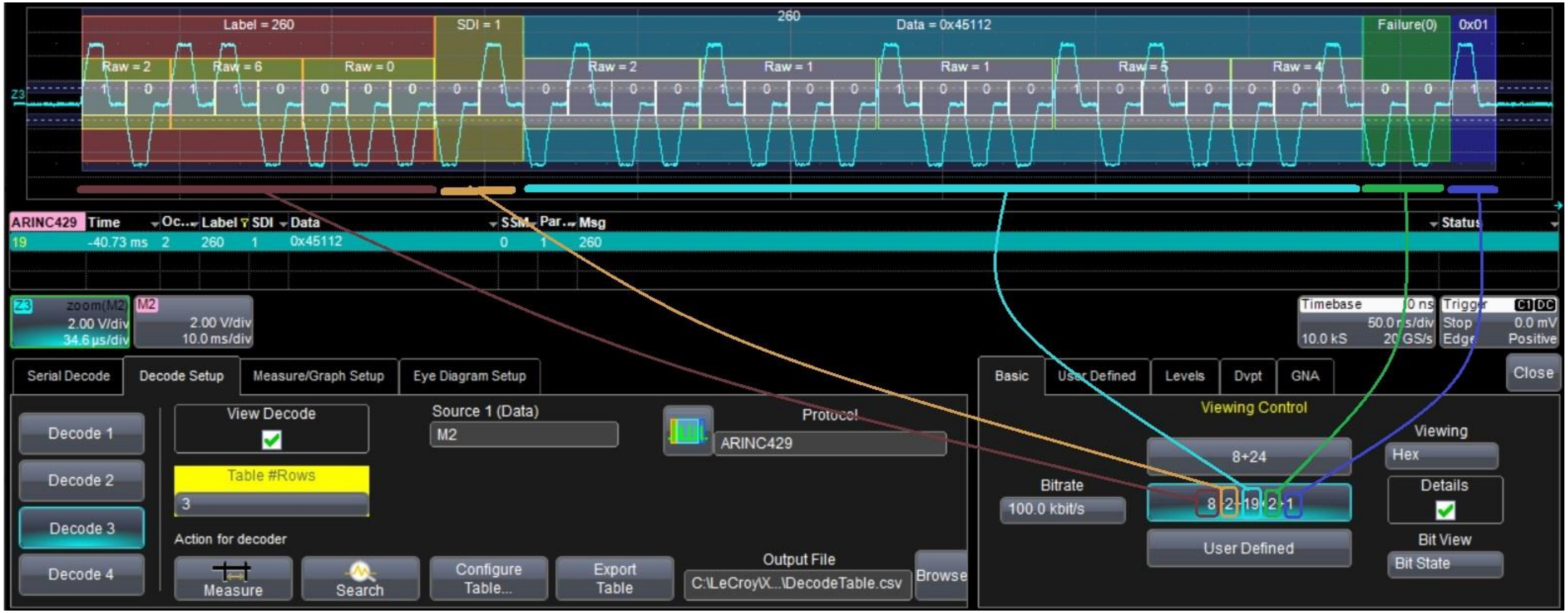


Figure 14 Signal decoded in the 8+2+19+2+1 bit mode

Both of the above modes have been chosen because there are frequently used by other tools in the ARINC 429 field and are reminiscent of the original specification. However, most examples that we have come across from the field can not be supported using these modes and require the Symbolic Decode Mode described in the next section

3.3 *The Symbolic Decode Mode*

This mode is the richest of all the 3 modes, is available on all instruments and it is described in the following sections. Unlike the simple modes described earlier, it requires the user to provide a definition file, containing the description of the signals emitted by the equipment.

When using the Symbolic mode (also known as User Defined or User Interpreted) the data field of each Label is decoded according to a file supplied by the user, for its own system and equipment. Each Label value is decoded independently. **The syntax of this file is described in detail in Appendix A**. Examples are also provided.

A text file drives the interpretation algorithm. The following image gives a feeling for these files, under a simple editor such as Notepad for MS Windows.

_cGPS_L101-102-110-111-260-273.txt - Notepad

File Edit Format View Help

```
//=====================================================================================
// Label 273,
273,-,SDI,,-,-,2, PosSen,1,Min,Max,Label_Discrete,8,Enum:Not Used|Unit1|Unit2|Unit3
// 5 individual bits
273,-,MSB Sat,,-,-,1, PosSen,1,Min,Max,Label_Discrete,10,Enum:<15 Vis|>15 Vis
273,-,FMS/DADS Stat,,-,-,1, PosSen,1,Min,Max,Label_Discrete,11,Enum:Present|Not Present
273,-,FMS/DADS Source,,-,-,1, PosSen,1,Min,Max,Label_Discrete,12,Enum:Prim|Sec
273,-,IRS/FMS Stat,,-,-,1, PosSen,1,Min,Max,Label_Discrete,13,Enum:Present|Not Present
273,-,IRS/FMS Source,,-,-,1, PosSen,1,Min,Max,Label_Discrete,14,Enum:Prim|Sec
273,-,Num Sat Vis,,-,-,4, PosSen,1,Min,Max,Label_Discrete,15,Binary
273,-,Num Sat Used,,-,-,4, PosSen,1,Min,Max,Label_Discrete,19,Binary
273,-,GLSSU Op Mode,,-,-,5, PosSen,1,Min,Max,Label_Discrete,23,Enum:Self-tst||||Init||||Acq||||Nav|SBAS Nav|SBAS PA|Alt
Aid|Res|Aided|Fault
273,-,MSB Sat Used,,-,-,1, PosSen,1,Min,Max,Label_Discrete,28,Enum:<=15|>15
273,-,SSM,,-,-,2, PosSen,1,Min,Max,Label_Discrete,29,Enum:Norm Op|NCD|FunctionalTest|FailureWarning
```

Figure 15 Example of ULDF file, with several USigD for Label 273 of a civilian GPS

4 Appendix A: Symbolic Decoding in More Details

4.1 Explanation of the User Label Definition File

4.1.1 Terminology, Conventions

The file is called ULDF (User Label Definition File)

It is expected that each Equipment (for example a radio Altimeter, or a GPS) requires one ULDF.

Variations of the same Equipment **manufactured over decades** could also require different ULDF files to accommodate differences of precision (Altitude precision to the meter or decimeter) or additional Labels (Support for GPS auxiliary results). However, there can be unused information in a ULDF file. If a piece of Equipment was emitting Labels 102,103,104,105 in its 1978 version and additionally Label 203,204,204 in its 1995 version, the same ULDF file could be used.

A line in the ULDF file is called a USigD (**User Signal Definition**). A USigD contains rules to interpret bits in an ARINC message. The Altitude, or the Distance To go, or the fuel Level in Tank 4 each require a USigD. Some parameters require more than a single USigD, for example the Date requires several BCD USigDs.

A ULDF is therefore made up of one or more USigD There is no limit for the number of USigDs in a Label.

USigDs beginning **with the same Label** will appear with all Labels of that value in the annotation and the table.

All Labels of the **same value** are decoded according to that **same rule**, as per ULDF

When the frequent (but not always used) fields **SDI and SSM** are desired, they will need a one line USigD per Label in which they are desired. Examples are given further in this document.

When the frequent (but not always used) **sign field** of a binary signal is desired, it will require a one line USigD.

Several fields (also called tokens) of the USigD are not used at present time but retained for compatibility with the Standard. They might become useful in the future.

4.1.2 Line Syntax

The text elements explain the syntax of the ULDF files are always shown in Helvetica.

The line structure one a single USigD in the Comma Separated File (CSV) contains 12 to 14 tokens, separated by commas as follows:

Label[1], EquipementID[2],Name[3],Units[4],Min[5],Max[6],SigBits[7],PosSense[8],Resolution[9],MinTransit[10],MaxTransit[11],LabelType1[12],Offset[13],DetailsList[14]

Each USigD's element has the following syntactic rules:

	Name of the field	Type of contents	Range	Used by algorithm
1	Label[1]	Decimal	0 to 377	Yes
2	EquipementID[2]	Text	Free text	No
3	Name[3]	Text	Free text	Yes
4	Units[4]	Text	Free text	Yes, for Binary and BCD USigD only
5	Min[5]	Text	Free text	No
6	Max[6]	Text	Free text	No
7	SigBits[7]	Decimal	1-32	Yes, by all subtypes
8	PosSense[8]	Text	Free text	No
9	Resolution[9]	Decimal	-max double to + max double	Yes, for Binary and BCD USigD only
10	MinTransit[10]	Text	Free text	No
11	MaxTransit[11]	Text	Free text	No
12	LabelType1[12]	Keyword	Label_Discrete	Yes
13	Offset[13]	Decimal	0-31	Yes
14	DetailsList[14]	Special format, see below		Yes

Figure 16 Listing of all Fields of the USigD line, with comments

ULDF files support commented lines starting with // as the normal C++ convention. Comment can also be added at the end of the line, **but not in the middle** of the token list.

4.1.3 Line Contents following ARINC 429

The structure of the line largely follows the ARINC 429 standards familiar to those working in the avionics domain. Deviations and additions are listed below and all cases are explained with examples in this document.

4.1.4 Deviations from ARINC 429

Token 2: Equipment ID is decimal rather than hexadecimal, and is not used by the annotation algorithm. This token can be used to document the USigD if desired.

Token 5/6: In the ARINC tables, the min and max are merged in one column, with several possible syntaxes, in human language, not readily parsable. In ULDF, min and max have dedicated columns, therefore dedicated tokens. Not used by algorithm.

4.1.5 Extensions to ARINC 429

Token 12: Label Types are always Discrete with subtypes Binary (signed), Unsigned Binary, Bcd and Enum

Token 13: Zero based Offset in bits from bit 0 at beginning of message

Token 14: DetailsList contains details for BCD and Enumerated Discretes.

For an enumerated type details would be i.e: Enum:On|Off, or Enum:Disengaged|Engaged or Enum:Off|Low|Medium|Full. In principle the number of enumerated keywords should match $2^{SigBits.}$However, shorter or longer lists will be accepted silently.

For a BCD type it could be i.e. BCD:3|4 or BCD:2|4|4|4. The sum of the BCD digits should be equal to SigBits, otherwise a warning will appear on the message line at the bottom of the screen.

4.1.6 Unused Tokens

Equipment ID, Min/Max Transit times, Min/Max values and PosSense and are parsed but not used at this stage.

4.2 Examples used in this document

The arbitrary labels and signals used for the examples in this document do not always have realistic contents. They are only meant to be examples of encoding/interpretations.

Any neutral text editor can be used to work on the ULDF file, as long as it does not add extraneous odd characters or remove characters.

In some cases, the BCD examples following contain values greater than 9 in a 4 bit BCD digit. This is incorrect but due to the difficulty to generate or gather syntactically corrects samples.

4.3 Evolution from 1978 specification concepts

The initial 1978 specification define BCD, Binaries and Discrete Labels, heavily linked to specific bit positions within the 32-bit message. As this tool was released and experienced was gained, it was noticed that over its 40+ life the protocol had been used in very many ways, departing from the original specification. It rapidly became clear that some of the 1978 concepts had been retained, while some had been evolved.

The 4 different signal storage type (BCD, signed and unsigned binary, Enumerated) were pretty much adhered to by the community.

The 8-bit BCD encoding of the Label as well as the Parity bit seem to have been widely respected by all implementation, probably because it is rooted in hardware.

However, the static use of the remaining 23 data bits was violated over time, leading to implementations in which **the native storage types are above are still used**, **but with position and span anywhere** within the 23 data bits.

It was therefore assumed that the ARINC 429 message could contain (within the same label) **any combination of binary, BCD and enumerated signals.** In this case the user is required to supply the offset, in bits, at which a signal starts within the label, and the bit span of the signal. This method not only supports any conceivable use of the data bits, but also allows backward compatibility with the 1978 definition of the data bits assignment.

We will now go through the 4 possible type of signal that can be embedded within the data bits of the ARNC 429 message.

4.3.1 Enumerated type (Signal value corresponds to text)

The enumerated type is a list of strings indexed by the value contained in the signal. For this type of signal, units and resolution supplied in the line are disregarded. In the following Figure, each of the relevant token in the USigD is visually connected to its annotation.

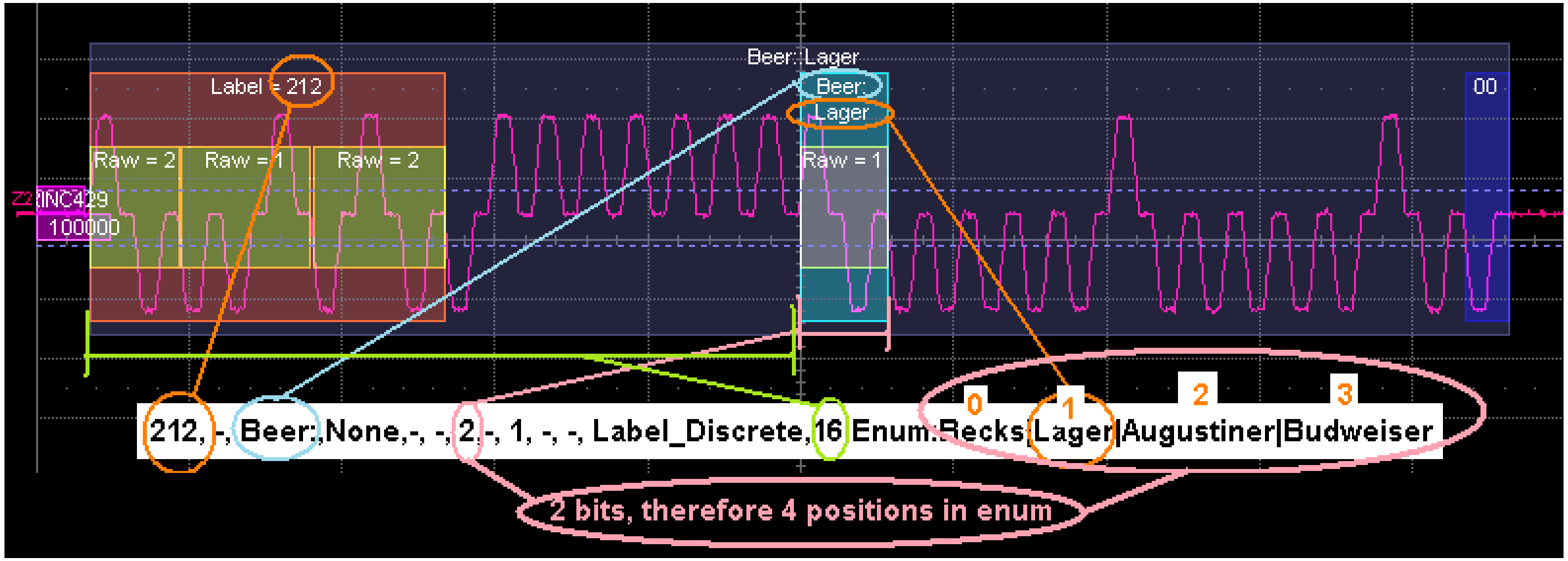


Figure 17 Visual explanation of each relevant token of an enumerated type

4.3.2 BCD Signal

The BCD format used for the BCD signals is flexible and allows the positioning and parametrizing of the BCD fields. In the following Figure, each of the relevant token in the USigD is visually connected to its annotation.

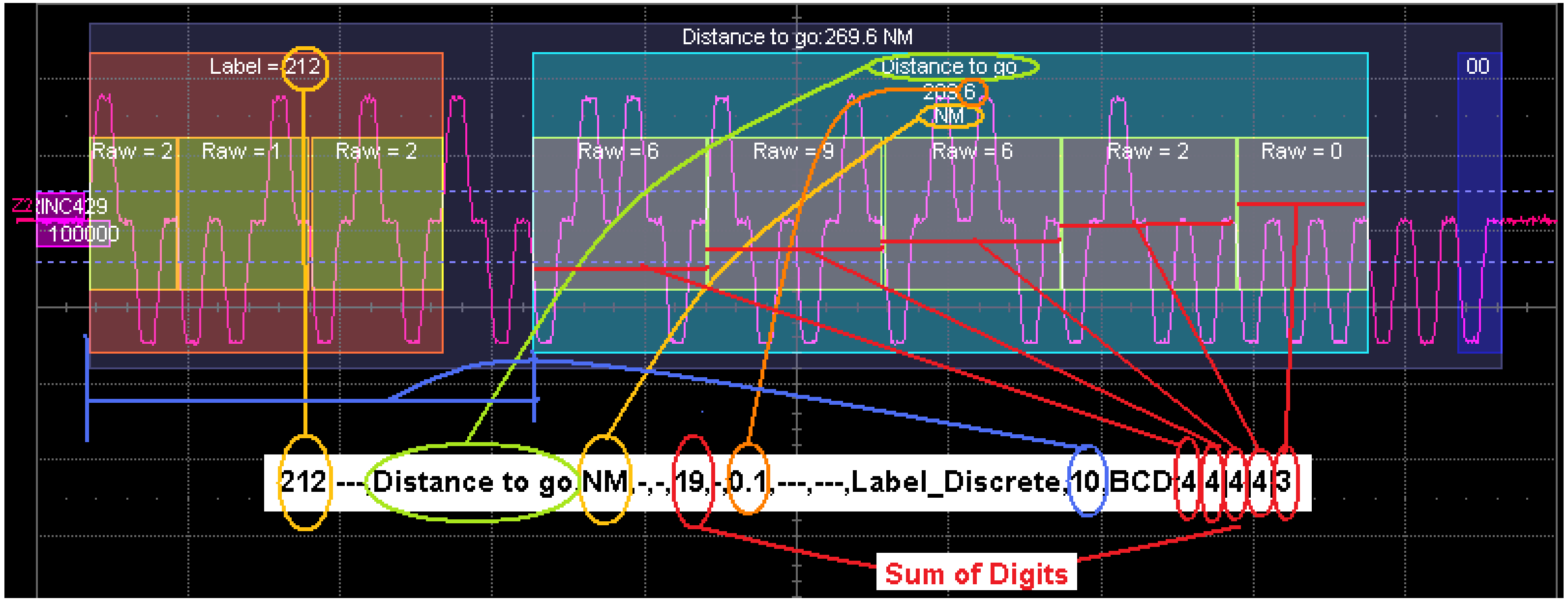


Figure 18 Visual explanation of each relevant token of a BCD type

4.3.3 Unsigned Binary Signal

The Binary format used for the binary signals only very slightly differs from the standard binary labels and allows the positioning of the Binary fields inside the message. We will explore a few examples here.

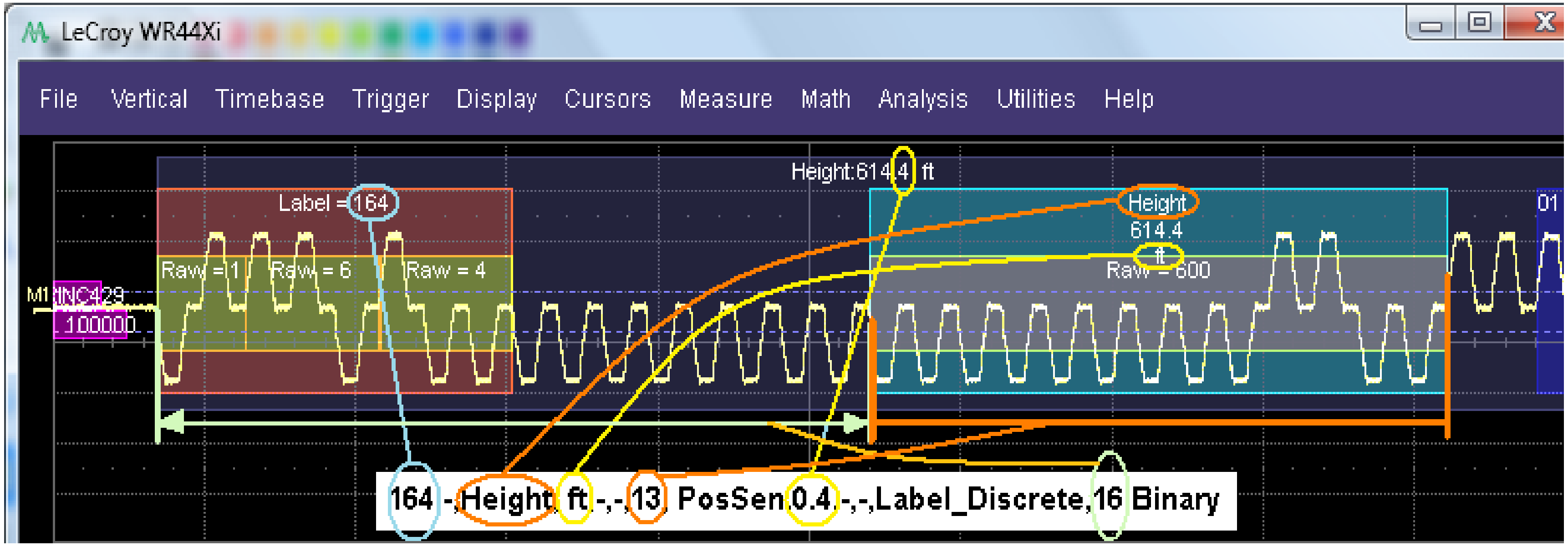


Figure 19 Visual explanation of each relevant token of an binary type

4.3.4 Signed Binary Signal

The Signed Binary format used for the Signed Binary signals only very slightly differs from the signed binary signals and allows the positioning of the Signed Binary fields inside the data bits of the ARINC 429 message.

The format is defined to exceed the ARINC 429 specification, which states that bit 29 is the sign bit. However, given the countless deviations from the standard since its inception in 1978, this implementation does not assume that the sign bit is necessarily located in bit 29. Its position result of the sum of **SigBits**[7] (in bits) and Offset[13] (in bits) specified by the user in the ULDF file.

Therefore, the position of the Sign Bit in the ARINC word is controllable via **SigBits**[7] and Offset[13]. The sum of these 2 values yields the position of the Sign Bit

In order to clearly distinguish Signed Binaries from Unsigned Binaries the ULDF syntax is extended as follows

110,-, Signal Name,,-,-,21,PosSen,0.000171661376953125,--,--,Label_Discrete,8,**SignedBinary**

The **SignedBinary** keyword in the DetailsList instructs the algorithm to interpret the bits spanned by **SigBits**[7] and Offset[13] as a 2's Binary complement

The following examples explain the behavior.

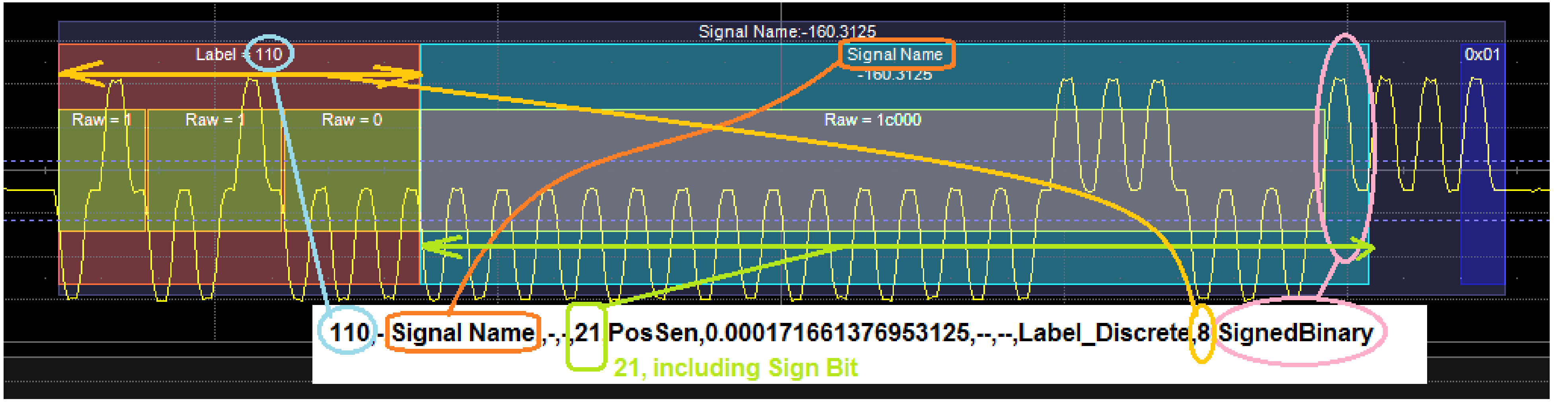


Figure 20 Visual explanation of each relevant token of a Signed Binary type, using Binary 2 complement

4.3.5 Combining Signals into a single Label

Several USigD can easily be combined to interpret the same Label. The following picture shows how the 3 USigDs used above can be assigned to the same label, in this case 212. Each of the USigD used in previous example has been copied into the single ULDF file. In order to avoid superposing the fields, the offset and length have been slightly changed.

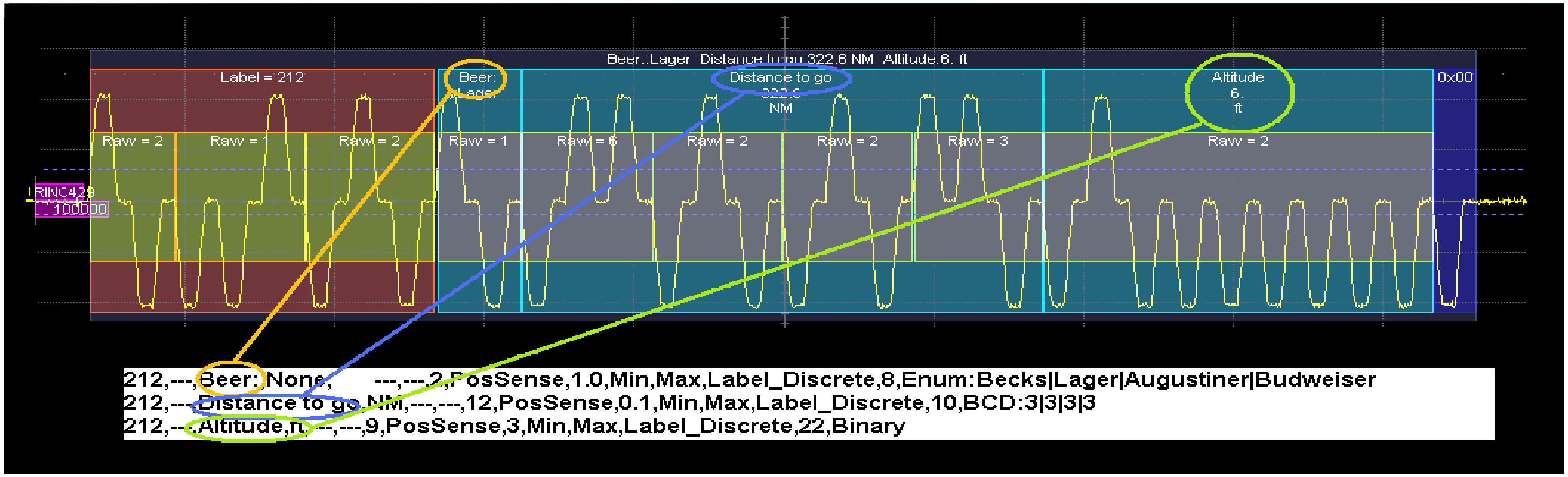


Figure 21 Combining 3 User Signals into single Label 212

Also note that if the USigDs are listed in increasing offset order in the file, they will appear in that order both in the Decode Table and the Label annotation.

This is a second example of a combination of fields within a label. Here Label 260 is interpreted as Date, in this case November 22nd, 2012

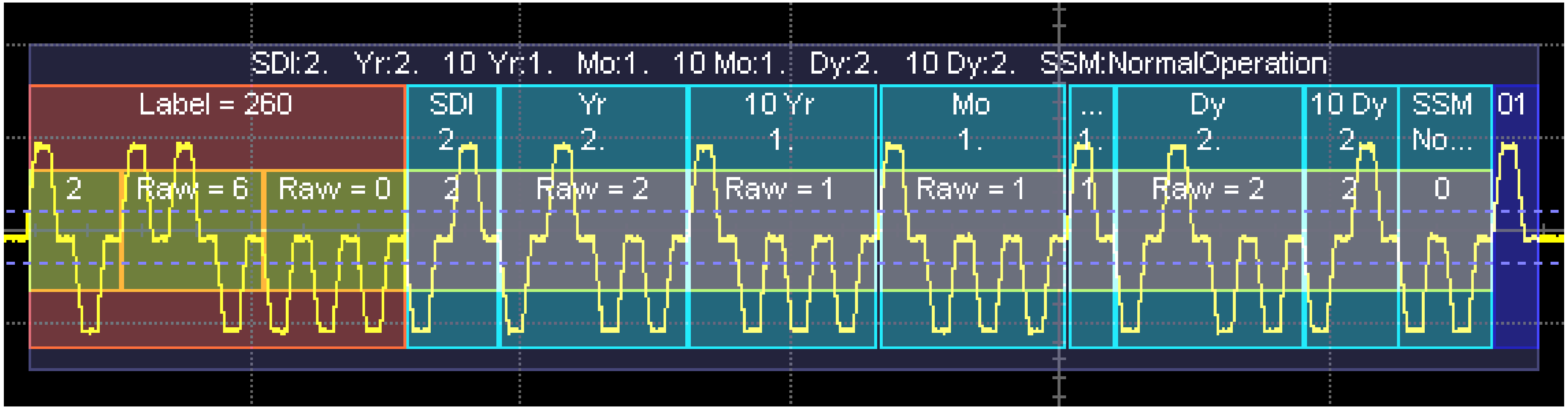


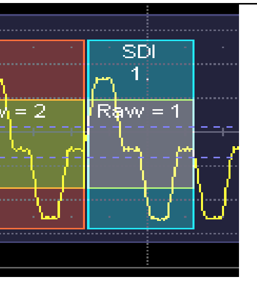
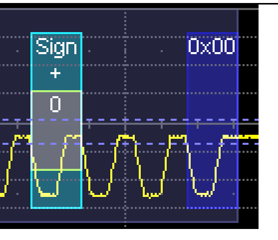
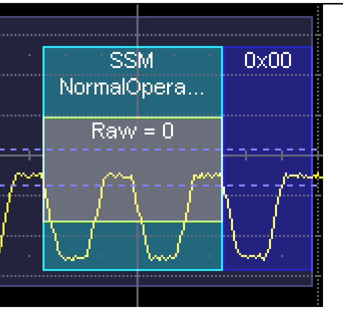
Figure 22 User defined interpretation of the Date Label 260

//===

// Label 260, Date
260,-,SDI,,-,-,2, PosSen,1,Min,Max,Label_Discrete,8,Binary
260,-,Yr,,-,-,4, PosSen,1,Min,Max,Label_Discrete,10,BCD:4
260,-,10 Yr ,,-,-,4, PosSen,1,Min,Max,Label_Discrete,14,BCD:4
260,-,Mo,,-,-,4, PosSen,1,Min,Max,Label_Discrete,18,BCD:4
260,-,10 Mo,,-,-,1, PosSen,1,Min,Max,Label_Discrete,22,BCD:1
260,-,Dy ,,-,-,4, PosSen,1,Min,Max,Label_Discrete,23,BCD:4
260,-,10 Dy ,,-,-,2, PosSen,1,Min,Max,Label_Discrete,27,BCD:2
260,-,SSM,,-,-,2, PosSen,1,Min,Max,Label_Discrete,29,Enum:NormalOperation|NoComputedData|FunctionalTest|FailureWarning

4.3.6 Common Signals found in many Labels

Some common User Signals have been used for decades in many Labels. In an attempt to ease the constitution of ULDF files, we list some of the common ones, with their screen renderings on an arbitrary signal. These lines can therefore be cut and paste into any ULDF file.

USigD coding example. **(The line can be cut and paste into the ULDF file)**	**Rendering on trace**	**Comment**
212,-,SDI,,-,-,2, -,1,-,-, Label_Discrete, 8, Binary		The Source Destination ID (SDI)used in some labels can be coded as a binary Signal In this case the Unit's field is left blank
212,-,Sign,,-,-,1, -,1,-,-,Label_Discrete,28,Enum:+|-		The sign fields used by some labels in bit 29 (or 28 when 0 based) can be coded as a 1 bit enumerated Signal.
212,-,SSM,,-,-,2, PosSen,1,Min,Max,Label_Discrete,29,Enum:NormalOperation|NoComputedData|FunctionalTest|FailureWarning		The SSM field for BCD data

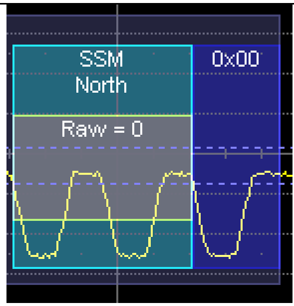
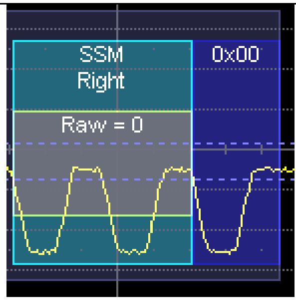
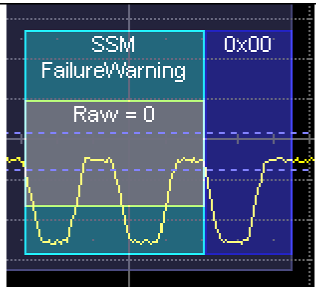
212,-,SSM,,-,-,2, PosSen,1,Min,Max,Label_Discrete,29,Enum:North|NoComputedData|FunctionalTest|South		The SSM field when used for BCD Latitude
212,-,SSM,,-,-,2, PosSen,1,Min,Max,Label_Discrete,29,Enum:Right|NoComputedData|FunctionalTest|Left		The SSM field when used for BCD Distance Right and Left
212,-,SSM,,-,-,2, PosSen,1,Min,Max,Label_Discrete,29,Enum:FailureWarning|NoComputedData|FunctionalTest|NormalOperation		The SSM field when used in conjunction with Binary Data

Figure 23 Table showing commonly found signals

5 Appendix B: Using the Table Filter

5.1 Concept

The Decode Table Filter mimics the filtering paradigm used by Microsoft Excel and is used across the decoders. First a column is selected, then a filter is applied to the values in the column.

In this example we will filter ARINC 429 messages based on their Label value. As long as no filter is active, the column header shows the white filter icon 

. As soon as a filter is defined the yellow filter icon appears 

.

Several columns can be filtered at the same time, therefore creating an AND condition over these columns. In this case several little yellow labels would appear, in this case on the columns "Label", "SSM" and "Parity"

Figure 24 Table Header with several columns filtered

5.2 Examples, using ARINC 429

Figure 25 Table Filter disabled, all messages are displayed

The observation of the lables in the table yields the impression that some lables repeat over time. We verify this hypothesus using a List Filter on the lables below and observe that several repeated block of 57-65 labels appear.

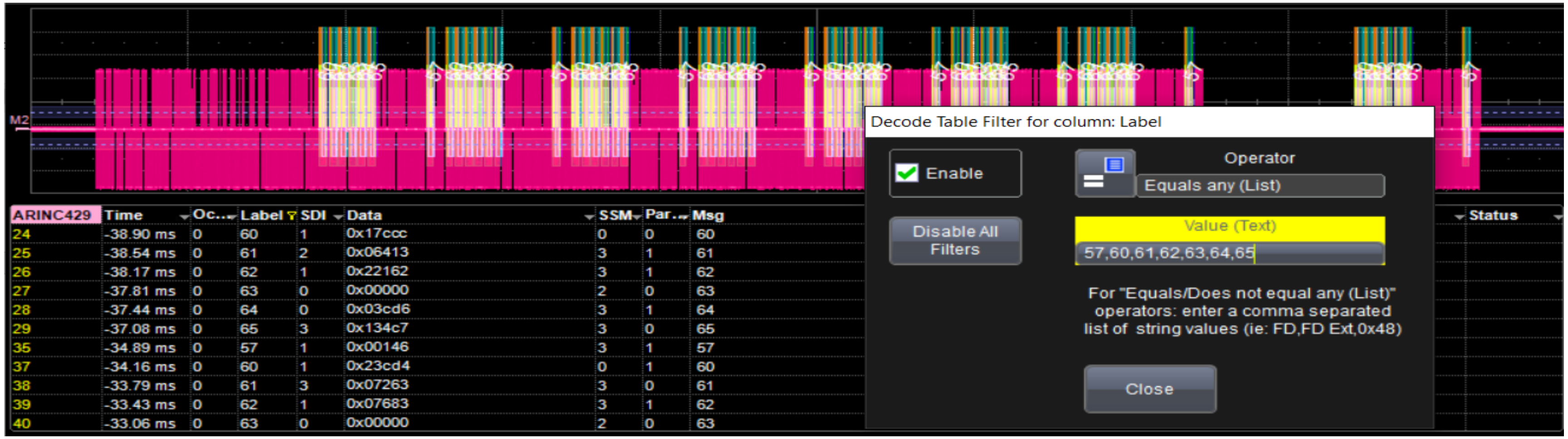


Figure 26 Table Filter Enabled, List of Labels

Following this initial intuition, we add another range of repeated labels to the filter list, becoming 57,60,61,62,63,64,65,66,70,71,72,73,74. Consequently we capture all the messages showed in the image below.

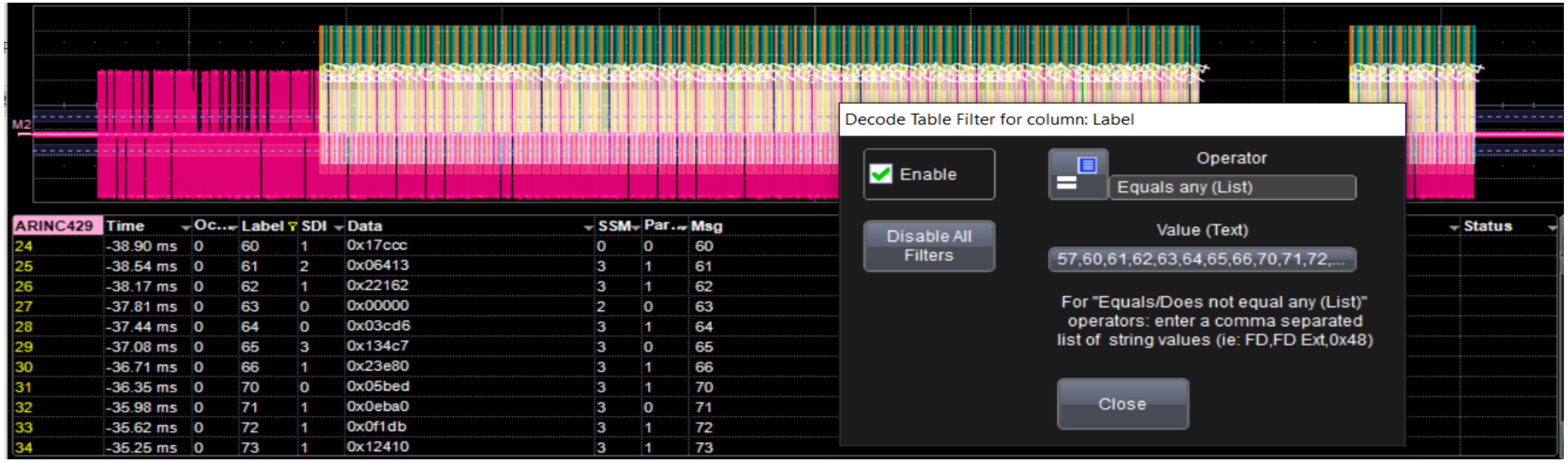


Figure 27 Table Filter on extended list of Labels

We can now turn our attention to the labels at the beginning of the record. With a bit of experimenting we discover that they all reside within the InRange of 101 to 176, which is proven by this experiment.

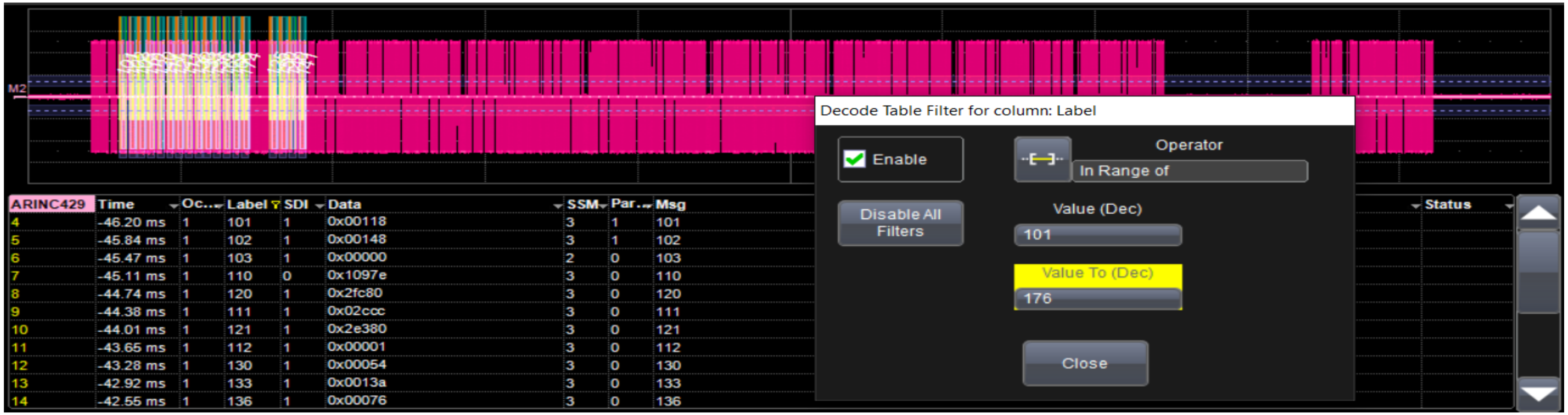


Figure 28 Table Filter using InRange operator

We are left with a few unaccounted labels at the beginning of the record. A bit more poking will show that they are labels 247,260,273,370, which again is proven by using the Equal Filter on these values.

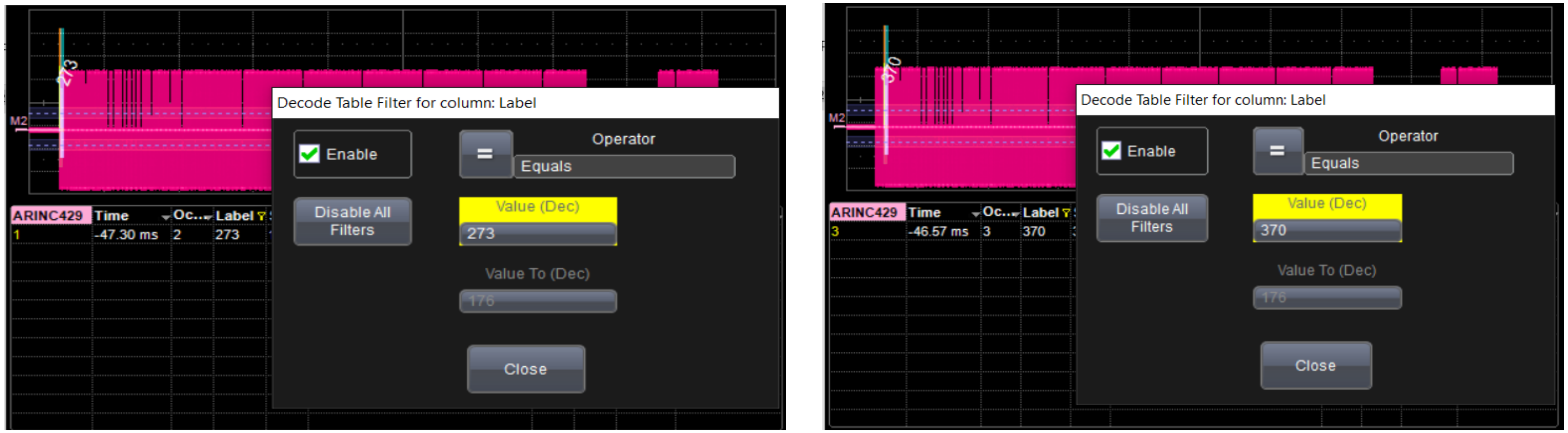


Figure 29 Table Filter using Equal operator on Label 273 and 370