**Functional Design Document**

**For**

**NxtrIntrpn**

**VERSION: 1.0**

**DATE: 20-Feb-2015**

**Prepared By:**

**Nexteer Automotive,**

**Saginaw, MI, USA**

**Revision History**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version** | **Description** | **Author** | **Section Modified** | **Date** | **Approved By** |
| 1.0 | Initial Version | K. Smith | All | 20-Feb-2015 | Nexteer |

**Table of Contents**

[1 Abbrevations And Acronyms 4](#_Toc412195615)

[2 References 5](#_Toc412195616)

[3 Purpose 6](#_Toc412195617)

[4 Interpolation Design 7](#_Toc412195618)

[4.1 Linear Interpolation 7](#_Toc412195619)

[4.1.1 Linear Interpoliation Accuracy 8](#_Toc412195620)

[4.2 Fixed X-Axis Linear Interpolation Functions 9](#_Toc412195621)

[4.2.1 API 9](#_Toc412195622)

[4.2.1.1 Truncating Functions 9](#_Toc412195623)

[4.2.1.2 Rounding Functions 9](#_Toc412195624)

[4.3 Variable X-Axis Linear Interpolation Functions 10](#_Toc412195625)

[4.3.1 API 10](#_Toc412195626)

[4.3.1.1 Truncating Functions 10](#_Toc412195627)

[4.3.1.2 Rounding Functions 12](#_Toc412195628)

[4.4 Bilinear Interpolation 14](#_Toc412195629)

[4.4.1 Common X Axis Bilinear Interpolation Functions 16](#_Toc412195630)

[4.4.1.1 API 16](#_Toc412195631)

[4.4.2 Variable X Axis Bilinear Interpolation Functions 19](#_Toc412195632)

[4.4.2.1 API 19](#_Toc412195633)

[5 Know Limitations With Design 22](#_Toc412195634)

[6 Appendix A 23](#_Toc412195635)

[7 Appendix B 24](#_Toc412195636)

[7.1 Truncating Linear Interpolation Functions 24](#_Toc412195637)

[7.1.1 Fixed X-Axis Interpolation Function 24](#_Toc412195638)

[7.1.2 Variable X-Axis Interpolation Function 24](#_Toc412195639)

[7.2 Rounding Linear Interpolation Functions 24](#_Toc412195640)

[7.2.1 Fixed X-Axis Interpolation Function 24](#_Toc412195641)

[7.2.2 Variable X-Axis Interpolation Function 24](#_Toc412195642)

[8 Appendix C 25](#_Toc412195643)

# Abbrevations And Acronyms

|  |  |
| --- | --- |
| Abbreviation | Description |
| BS | Bilinear Selection |
| API | Application Program Interface |

# References

This section lists the title & version of all the documents that are referred for development of this document

|  |  |  |
| --- | --- | --- |
| Sr. No. | Title | Version |
| Appendix C | RH850/P1x Series User’s Manual: Software | 0.10 Jan, 2014 |

# Purpose

The purpose of this document is to describe the functions contained within the Nexteer interpolation library and as an API reference in designing functional requirements, models, and software components.

# Interpolation Design

## Linear Interpolation

Linear interpolation is used to determine an output value between a set of known data points for a given input. In the image below, the input point (x) is between the known points (x0,y0) and (x1,y1).



The linear interpolant is defined as the straight line between the two known points (x0,y0) and (x1,y1) and is defined in equation (1).

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

Solving equation (1) for the desired output (y) is described below in equation (2).

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

If the delta between neighboring points on the X-axis is the same, equation (2) can be modified with that delta as shown in equation (3).

|  |  |  |
| --- | --- | --- |
|  |  | (3) |

## Linear Interpoliation Accuracy

The operations required to perform equations (2) and (3) shall be implemented using fixed point math. This is to save on execution time to perform the interpolation compared to using floating point math functions (See Appendix C). This also means that there can be information loss because the operations can produce results that have more bits than the operands. In order to keep the same number of bits as the operands, the answer must be rounded or truncated.

The interpolation library shall support both truncation and rounding methods in all linear interpolation functions. These methods are described in Appendix B. The error produced by the truncated function results can be +/- one count. Depending on the resolution of the calibration and the resolution required by the design, this may be negligible. The function designer shall use the truncation library functions in all cases where this error is acceptable in order to save on execution time.

## Fixed X-Axis Linear Interpolation Functions

The following functions are available for linear interpolation with a fixed X-axis and are based on equation (3). The table below defines each argument used in the API. Note that some functions require and/or return unsigned or signed values. It is up to the designer to pick the proper interpolation function for their design.

|  |  |
| --- | --- |
| Argument | Notes |
| DeltaX | Provides the delta between points in the X-axis for interpolation. A value of 0 shall return the first value in YTbl. |
| YTbl[] | Y table values |
| Size | Number of elements in YTbl |
| Inp | Input value (x). Values that are less than or equal to DeltaX or larger than DeltaX \* (Size – 1) will return the first or last value respectively from YTbl. |

## API

The input and output variables are defined as uint16 or sint16 for purposes of min and max ranges. However, calibrations with different resolutions, for example u8p8, can be used because they are still represented as a 16-bit value within these functions.

## Truncating Functions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | LnrIntrpn\_u16\_u16FixdXu16VariY | Type | Min | Max |
| **Arguments Passed** | DeltaX | uint16 | 0 | 65535 |
|  | YTbl[] | uint16[] | 0 | 65535 |
|  | Size | uint16 | 1 | 65535 |
|  | Inp | uint16 | 0 | 65535 |
| **Return Value** | | uint16 | 0 | 65535 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | LnrIntrpn\_s16\_u16FixdXs16VariY | Type | Min | Max |
| **Arguments Passed** | DeltaX | uint16 | 0 | 65535 |
|  | YTbl[] | sint16[] | -32768 | 32767 |
|  | Size | uint16 | 1 | 65535 |
|  | Inp | uint16 | 0 | 65535 |
| **Return Value** | | sint16 | -32768 | 32767 |

## Rounding Functions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | LnrIntrpnWithRound\_u16\_u16FixdXu16VariY | Type | Min | Max |
| **Arguments Passed** | DeltaX | uint16 | 0 | 65535 |
|  | YTbl[] | uint16[] | 0 | 65535 |
|  | Size | uint16 | 1 | 65535 |
|  | Inp | uint16 | 0 | 65535 |
| **Return Value** | | uint16 | 0 | 65535 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | LnrIntrpnWithRound\_s16\_u16FixdXs16VariY | Type | Min | Max |
| **Arguments Passed** | DeltaX | uint16 | 0 | 65535 |
|  | YTbl[] | sint16[] | -32768 | 32767 |
|  | Size | uint16 | 1 | 65535 |
|  | Inp | uint16 | 0 | 65535 |
| **Return Value** | | sint16 | -32768 | 32767 |

## Variable X-Axis Linear Interpolation Functions

The following functions are available for linear interpolation with a variable X-axis and are based on equation (2). The table below defines each argument used in the API. Note that some functions require and/or return unsigned or signed values. It is up to the designer to pick the proper interpolation function for their design.

|  |  |
| --- | --- |
| Argument | Notes |
| XTbl[] | X Table |
| YTbl[] | Y Table |
| Size | Number of elements in the XTbl and YTbl |
| Inp | Input value (x). Values that are less than or equal to XTbl[0] or larger than XTbl[Size – 1] will return the first or last value respectively from YTbl. |

## API

The input and output variables are defined as uint16 or sint16 for purposes of min and max ranges. However, calibrations with different resolutions, for example u8p8, can be used because they are still represented as a 16-bit value within these functions.

## Truncating Functions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | LnrIntrpn\_u16\_u16VariXu16VariY | Type | Min | Max |
| **Arguments Passed** | XTbl[] | uint16[] | 0 | 65535 |
|  | YTbl[] | uint16[] | 0 | 65535 |
|  | Size | uint16 | 1 | 65535 |
|  | Inp | uint16 | 0 | 65535 |
| **Return Value** | | uint16 | 0 | 65535 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | LnrIntrpn\_u16\_s16VariXu16VariY | Type | Min | Max |
| **Arguments Passed** | XTbl[] | sint16[] | -32768 | 32767 |
|  | YTbl[] | uint16[] | 0 | 65535 |
|  | Size | uint16 | 1 | 65535 |
|  | Inp | sint16 | -32768 | 32767 |
| **Return Value** | | uint16 | 0 | 65535 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | LnrIntrpn\_s16\_s16VariXs16VariY | Type | Min | Max |
| **Arguments Passed** | XTbl[] | sint16[] | -32768 | 32767 |
|  | YTbl[] | sint16[] | -32768 | 32767 |
|  | Size | uint16 | 1 | 65535 |
|  | Inp | sint16 | -32768 | 32767 |
| **Return Value** | | sint16 | -32768 | 32767 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | LnrIntrpn\_s16\_u16VariXs16VariY | Type | Min | Max |
| **Arguments Passed** | XTbl[] | uint16[] | 0 | 65535 |
|  | YTbl[] | sint16[] | -32768 | 32767 |
|  | Size | uint16 | 1 | 65535 |
|  | Inp | uint16 | 0 | 65535 |
| **Return Value** | | sint16 | -32768 | 32767 |

## Rounding Functions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | LnrIntrpnWithRound\_u16\_u16VariXu16VariY | Type | Min | Max |
| **Arguments Passed** | XTbl[] | uint16[] | 0 | 65535 |
|  | YTbl[] | uint16[] | 0 | 65535 |
|  | Size | uint16 | 1 | 65535 |
|  | Inp | uint16 | 0 | 65535 |
| **Return Value** | | uint16 | 0 | 65535 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | LnrIntrpnWithRound\_u16\_s16VariXu16VariY | Type | Min | Max |
| **Arguments Passed** | XTbl[] | sint16[] | -32768 | 32767 |
|  | YTbl[] | uint16[] | 0 | 65535 |
|  | Size | uint16 | 1 | 65535 |
|  | Inp | sint16 | -32768 | 32767 |
| **Return Value** | | uint16 | 0 | 65535 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | LnrIntrpnWithRound\_s16\_s16VariXs16VariY | Type | Min | Max |
| **Arguments Passed** | XTbl[] | sint16[] | -32768 | 32767 |
|  | YTbl[] | sint16[] | -32768 | 32767 |
|  | Size | uint16 | 1 | 65535 |
|  | Inp | sint16 | -32768 | 32767 |
| **Return Value** | | sint16 | -32768 | 32767 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | LnrIntrpnWithRound\_s16\_u16VariXs16VariY | Type | Min | Max |
| **Arguments Passed** | XTbl[] | uint16[] | 0 | 65535 |
|  | YTbl[] | sint16[] | -32768 | 32767 |
|  | Size | uint16 | 1 | 65535 |
|  | Inp | uint16 | 0 | 65535 |
| **Return Value** | | sint16 | -32768 | 32767 |

## Bilinear Interpolation

Bilinear interpolation is linear interpolation with expanded coverage into three dimensions. To calculate the result, three linear interpolations are required to be performed.



The first two interpolations are between the two data sets (shown in red and green in the image) and are described with the equations (4) and (5).

|  |  |  |
| --- | --- | --- |
|  |  | (4) |

|  |  |  |
| --- | --- | --- |
|  |  | (5) |

Equations (4) and (5) provide the endpoints for the interpolant between the two sets of data. The third interpolation determines how close the desired output is to each of the data sets. This is described in equation (6).

|  |  |  |
| --- | --- | --- |
|  |  | (6) |

Since divisions are throughput intensive operations, equation (6) is not efficient for an embedded environment because it contains three division steps when substitutions are made for Y1 and Y2. However, the terms can be rearranged to reduce the amount of divisions for additional multiplications and additions, which typically are much less in throughput consumption. Staring with equations (4) and (5), the terms can be reordered into equations (7) and (8).

|  |  |  |
| --- | --- | --- |
|  |  | (7) |

|  |  |  |
| --- | --- | --- |
|  |  | (8) |

Equations (7) and (8) can be substituted back in equation (6) and the simplified result is shown in equation (9).

|  |  |  |
| --- | --- | --- |
|  |  | (9) |

Finally, the numerator in equation (9) can also be rearranged and split into multiple terms. The steps taken between equations (9) and (10) are provided in Appendix A. The final equation, requiring only one division step, is shown in equation (10).

|  |  |  |
| --- | --- | --- |
|  |  | (10) |

## Common X Axis Bilinear Interpolation Functions

The following functions are available for bilinear interpolation with a common X-axis and are based on equation (10). The table below defines each argument used in the API. Note that some functions require and/or return unsigned or signed values. It is up to the designer to pick the proper interpolation function for their design.

|  |  |
| --- | --- |
| Argument | Notes |
| BilnrSeln | Bilinear Selection (BS-axis).  *If BilnrSeln is less than or equal to BilnrSelnTbl[0], BilnrSeln will be assigned BilnrSelnTbl[0] and the BS index will be set to 0. If BilnrSeln is greater or equal to BilnrSerlnTbl[BilnrSelnSize – 1], then the BilnrSeln will be assigned BilnrSelnTbl[BilnrSelnSize – 1] and the BS index will be set to BilnrSelnSize – 2.* |
| Inp | Input (X-axis).  *If Inp is less than or equal to XTbl[0], the value will be assigned XTbl[0] and the X index will be set to 0. If it is greater than or equal to XTbl[XSize – 1], Inp will be assigned XTbl[XSize – 1] and the X index will be set to XSize – 2.* |
| BilnrSelnTbl[] | Bilinear selection table |
| BilnrSelnSize | Number of elements in Bilinear selection table |
| XTbl[] | X table |
| MplYTbl[] | Y table (Number of elements = BilnrSelnSize \* XSize)  *In most cases, Y table will be a 2-D table, and is declared as MplYTbl[BilnrSelnSize][XSize]* |
| XSize | Size of X table |

## API

The input and output variables are defined as uint16 or sint16 for purposes of min and max ranges. However, calibrations with different resolutions, for example u8p8, can be used because they are still represented as a 16-bit value within these functions.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | BilnrIntrpnWithRound\_u16\_u16CmnXu16MplY | Type | Min | Max |
| **Arguments Passed** | BilnrSeln | uint16 | 0 | 65535 |
|  | Inp | uint16 | 0 | 65535 |
|  | BilnrSelnTbl[] | uint16[] | 0 | 65535 |
|  | BilnrSelnSize | uint16 | 1 | 65535 |
|  | XTbl[] | uint16[] | 0 | 65535 |
|  | MplYTbl[] | uint16[] | 0 | 65535 |
|  | XSize | uint16 | 1 | 65535 |
| **Return Value** | | uint16 | 0 | 65535 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | BilnrIntrpnWithRound\_s16\_u16CmnXs16MplY | Type | Min | Max |
| **Arguments Passed** | BilnrSeln | uint16 | 0 | 65535 |
|  | Inp | uint16 | 0 | 65535 |
|  | BilnrSelnTbl[] | uint16[] | 0 | 65535 |
|  | BilnrSelnSize | uint16 | 1 | 65535 |
|  | XTbl[] | uint16[] | 0 | 65535 |
|  | MplYTbl[] | sint16[] | -32768 | 32767 |
|  | XSize | uint16 | 1 | 65535 |
| **Return Value** | | sint16 | -32768 | 32767 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | BilnrIntrpnWithRound\_s16\_s16CmnXs16MplY | Type | Min | Max |
| **Arguments Passed** | BilnrSeln | uint16 | 0 | 65535 |
|  | Inp | sint16 | -32768 | 32767 |
|  | BilnrSelnTbl[] | uint16[] | 0 | 65535 |
|  | BilnrSelnSize | uint16 | 1 | 65535 |
|  | XTbl[] | sint16[] | -32768 | 32767 |
|  | MplYTbl[] | sint16[] | -32768 | 32767 |
|  | XSize | uint16 | 1 | 65535 |
| **Return Value** | | sint16 | -32768 | 32767 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | BilnrIntrpnWithRound\_u16\_s16CmnXu16MplY | Type | Min | Max |
| **Arguments Passed** | BilnrSeln | uint16 | 0 | 65535 |
|  | Inp | sint16 | -32768 | 32767 |
|  | BilnrSelnTbl[] | uint16[] | 0 | 65535 |
|  | BilnrSelnSize | uint16 | 1 | 65535 |
|  | XTbl[] | sint16[] | -32768 | 32767 |
|  | MplYTbl[] | uint16[] | 0 | 65535 |
|  | XSize | uint16 | 1 | 65535 |
| **Return Value** | | uint16 | 0 | 65535 |

## Variable X Axis Bilinear Interpolation Functions

The following functions are available for bilinear interpolation with a variable X-axis and are based on equation (10). The table below defines each argument used in the API. Note that some functions require and/or return unsigned or signed values. It is up to the designer to pick the proper interpolation function for their design.

|  |  |
| --- | --- |
| Argument | Notes |
| BilnrSeln | Bilinear Selection (BS-axis).  *If BilnrSeln is less than or equal to BilnrSelnTbl[0], BilnrSeln will be assigned BilnrSelnTbl[0] and the BS index will be set to 0. If BilnrSeln is greater or equal to BilnrSerlnTbl[BilnrSelnSize – 1], then the BilnrSeln will be assigned BilnrSelnTbl[BilnrSelnSize – 1] and the BS index will be set to BilnrSelnSize – 2.* |
| Inp | Input (X-axis).  *If Inp is less than or equal to XTbl[0], the value will be assigned XTbl[0] and the X index will be set to 0. If it is greater than or equal to XTbl[XSize – 1], Inp will be assigned XTbl[XSize – 1] and the X index will be set to XSize – 2.* |
| BilnrSelnTbl[] | Bilinear selection table |
| BilnrSelnSize | Number of elements in Bilinear selection table |
| MplXTbl [] | X table (Number of elements = BilnrSelnSize \* XSize)  *In most cases, X table will be a 2-D table, and is declared as MplXTbl[BilnrSelnSize][XSize]* |
| MplYTbl[] | Y table (Number of elements = BilnrSelnSize \* XSize)  *In most cases, Y table will be a 2-D table, and is declared as MplYTbl[BilnrSelnSize][XSize]* |
| XSize | Size of X table |

## API

The input and output variables are defined as uint16 or sint16 for purposes of min and max ranges. However, calibrations with different resolutions, for example u8p8, can be used because they are still represented as a 16-bit value within these functions.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | BilnrIntrpnWithRound\_u16\_u16MplXu16MplY | Type | Min | Max |
| **Arguments Passed** | BilnrSeln | uint16 | 0 | 65535 |
|  | Inp | uint16 | 0 | 65535 |
|  | BilnrSelnTbl[] | uint16[] | 0 | 65535 |
|  | BilnrSelnSize | uint16 | 1 | 65535 |
|  | MplXTbl[] | uint16[] | 0 | 65535 |
|  | MplYTbl[] | uint16[] | 0 | 65535 |
|  | XSize | uint16 | 1 | 65535 |
| **Return Value** | | uint16 | 0 | 65535 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | BilnrIntrpnWithRound\_u16\_s16MplXu16MplY | Type | Min | Max |
| **Arguments Passed** | BilnrSeln | uint16 | 0 | 65535 |
|  | Inp | sint16 | -32768 | 32767 |
|  | BilnrSelnTbl[] | uint16[] | 0 | 65535 |
|  | BilnrSelnSize | uint16 | 1 | 65535 |
|  | MplXTbl[] | sint16[] | -32768 | 32767 |
|  | MplYTbl[] | uint16[] | 0 | 65535 |
|  | XSize | uint16 | 1 | 65535 |
| **Return Value** | | uint16 | 0 | 65535 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | BilnrIntrpnWithRound\_s16\_s16MplXs16MplY | Type | Min | Max |
| **Arguments Passed** | BilnrSeln | uint16 | 0 | 65535 |
|  | Inp | sint16 | -32768 | 32767 |
|  | BilnrSelnTbl[] | uint16[] | 0 | 65535 |
|  | BilnrSelnSize | uint16 | 1 | 65535 |
|  | MplXTbl[] | sint16[] | -32768 | 32767 |
|  | MplYTbl[] | sint16[] | -32768 | 32767 |
|  | XSize | uint16 | 1 | 65535 |
| **Return Value** | | sint16 | -32768 | 32767 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | BilnrIntrpnWithRound\_s16\_u16MplXs16MplY | Type | Min | Max |
| **Arguments Passed** | BilnrSeln | uint16 | 0 | 65535 |
|  | Inp | uint16 | 0 | 65535 |
|  | BilnrSelnTbl[] | uint16[] | 0 | 65535 |
|  | BilnrSelnSize | uint16 | 1 | 65535 |
|  | MplXTbl[] | uint16[] | 0 | 65535 |
|  | MplYTbl[] | sint16[] | -32768 | 32767 |
|  | XSize | uint16 | 1 | 65535 |
| **Return Value** | | sint16 | -32768 | 32767 |

# Known Limitations With Design

1. The linear interpolation functions are implemented using signed 32-bit values for the calculation of the interpolation. If the product of the X and Y terms in the numerator are greater than or equal to 2,147,483,648 or 0x80000000 the rounding functions will apply the rounding incorrectly resulting in a 1 count error due to an overflow of the sign-bit. The truncating functions should be used in these situations.
2. The bilinear functions are calculated using single-precision floating point math.
3. Values of tables for linear interpolation X-axis and each row in the x-axis in a bilinear interpolation are assumed to be increasing in value.
4. Consider the array of values below:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| X-Axis | 0 | 1 | 2 | 2 | 2 | 2 | 2 | 2 |
| Y-Axis | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 |

These values can be used in the interpolation library. If the input value is less than 2, then the interpolation will be performed as described in this document. If the value is greater than or equal to 2, then index 7 will be returned. This also is true for the bilinear selection table.

1. Consider the array of values below:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| X-Axis | 0 | 1 | 2 | 2 | 2 | 2 | 2 | 4 |
| Y-Axis | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 |

These values can also be used in the interpolation library. If the input value is less than 2, then the interpolation will be performed as described in this document. If the value is greater than or equal to 2 and less than 4, the function will interpolate between index 6 and 7 as described in this document. If the value is greater than or equal to 4, index 7 is returned. This also is true for the bilinear selection table.

1. All division operations in linear interpolation functions are not protected for division by zero to save on execution time. The designer shall take all efforts to ensure that the values selected for the tables will not result in a divide by zero operation.

# Appendix A

The following section will demonstrate how equation (9) is recorded into the equation (10) described in the section Bilinear Interpolation.

Using equation (9) from section 5.2, the numerator and denominator can be subsitutituted for ynum and yden as shown in equation (A1).

|  |  |  |
| --- | --- | --- |
|  |  | (A1) |

Substituting back in all the terms in ynum, the numerator can be simplified as shown in equations A2 to A4.

Expanding the Y1Num and Y2Num term:

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  | (A2) |

Simplifying (A2)

|  |  |  |
| --- | --- | --- |
|  |  | (A3) |

Now equation (A3) can be grouped into smaller terms and simplified.

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  | (A4) |

# Appendix B

The equations below can be used by a designer to see the impact of the truncating and rounding functions on their design.

## Truncating Linear Interpolation Functions

From equations (2) and (3), the truncated result can be characterized by equations (B1) and (B2).

## Fixed X-Axis Interpolation Function

|  |  |  |
| --- | --- | --- |
|  |  | (B1) |

## Variable X-Axis Interpolation Function

|  |  |  |
| --- | --- | --- |
|  |  | (B2) |

## Rounding Linear Interpolation Functions

From equations (2) and (3), the rounding result can be characterized by equations (B3) and (B4).

## Fixed X-Axis Interpolation Function

|  |  |  |
| --- | --- | --- |
|  |  | (B3) |

## Variable X-Axis Interpolation Function

|  |  |  |
| --- | --- | --- |
|  |  | (B4) |

# Appendix C

The chart below shows the estimated number of clock cycles that are required for the fixed x-axis linear interpolation for various calculation methods. The source file was converted to assembly code for the Renesas RH850 microcontroller targeted for EA4 applications. The number of cycles for each instruction was determined based on the software user’s manual [1]. The best/worst case values are based on all predictive branching passing or failing and all other optimizations are ignored.

Based on the information, use of floating point calculations is too costly in terms of clock cycles for use throughout the project. Where the truncating and rounding functions offer the same results for almost half the execution time.