SigComp Specification

Signal Composer

Revision 1.0

14 July 2025

English　ver.

Table of Contents (Translated)

1. Overview

1.1. Introduction

- Signal Composer (hereafter referred to as *sigComp*) is an embedded core that performs filtering, rate conversion, modulation, and mixing with other data on digital signals.
- All data manipulations, including parameters and coefficients, are performed through memory access. Multiple data streams can be handled via time division, allowing for flexible operations.
- The core supports N-order filtering using polyphase filters and resampling at integer ratios. The filter order can be set freely up to 65536, and arbitrary memory regions can be specified as coefficients.
- A single modulation signal can be used to manipulate the filtered data. Operations include multiplication, addition, and DDA (Digital Differential Analyzer). Like coefficients, arbitrary regions in memory can be specified.
- Input/output data can be either single-precision floating point (IEEE754) or fixed-point less than 1 (16-bit or 23-bit). Internal calculation precision is all in single-precision floating point.
- The filter consumes cycles proportional to the filter order. If the order is zero, the final throughput is one cycle.

Note: The memory interface must be customized depending on the system.

1.2. Main Parameters

- **Memory Bus**:
    - Data Read/Write: 32-bit × 4
    - Command List Read: 64-bit × 1
- **Throughput**: N+1 samples per cycle (N = filter order)
- **Signal Format**:
    - 32-bit floating point data (IEEE754 single precision)
    - 16-bit floating point data (IEEE754 half precision)
    - 32-bit fixed point data (MSB sign, LSB 23-bit valid)
    - 16-bit fixed point data (unsigned, 16-bit valid)
    - 32-bit integer (unsigned, for DDA output)

    (For reading, LR-packing is optional)

- **Coefficient Format**: 32-bit floating point data (IEEE754 single precision)
- **Modulation Format**: 32-bit floating point data (IEEE754 single precision)
- **Clock**: Undefined (depends on implementation process)

1.3. Implementation Parameters

Parameter Name	**Description**	**Default Value**
BLR	● Burst length radix for external (parameter) bus ● sets burst unit for 64-bit memory access	1 (≤ 4)
BSR	● Burst length radix for data bus ● sets burst unit for 64-bit memory access	1 (≤ 4)

2. Signal Lines

2.1. Control Bus Interface

Signal Name	IO	Pol	Source	Description
cntlReq	I	+	clk	• Request signal • Evaluate cntlGnt
cntlGnt	O	+	clk	• Grant signal
cntlRxw	I	+	clk	• R/W signal • Evaluate cntlReq & cntlGnt 0: Write 1: Read
cntlAddr[31:0]	I	+	clk	• Address signal • Evaluate cntlReq & cntlGnt
cntlWrAck	O	+	clk	• Write acknowledge signal
cntlWrData[31:0]	I	+	clk	• Write data signal • Evaluate cntlWrAck
cntlRdAck	O	+	clk	• Read acknowledge signal
cntlRdData[31:0]	O	+	clk	• Read data signal • Sync cntlRdAck

cntlIrq	O	+	clk	• Interrupt signal • Level hold type(Fix'0')

2.2. PSS Interface

Signal Name	IO	Pol	Source	Description
iVld	I	+	clk	• Pipeline start valid signal
iStall	O	+	clk	• Pipeline start stall signal
iEnd[3:0]	I	+	clk	• Information of end of indexes
iAddr[31:4]	I	+	clk	• Address to fetch context data • Evaluate iVld &! iStall
iDelta[15:0]	I	+	clk	• Transfer volume • Evaluate iVld &! iStall
iIndex[64:0]	I	+	clk	• Five coordinates to specify the processing • Evaluate iVld &! iStall
oVld	O	+	clk	• Pipeline end valid signal
oStall	I	+	clk	• Pipeline end stall signal

2.3. Memory Interface (Read Use)

Signal Name	IO	Pol	Source	Description
mc$_n$Req	O	+	clk	• Request signal
mc$_n$Gnt	I	+	clk	• Grant signal
mc$_n$Rxw	I	+	clk	• R/W signal • Write indicates cache flush
mc$_n$Addr[31:0]	O	+	clk	• Address signal
mc$_n$RdStrb	O	+	clk	• Read strobe
mc$_n$RdAck	I	+	clk	• Read acknowledge signal
mc$_n$RdData[31:0]	I	+	clk	• Read data signal

Note: The subscript "n" in signal names indicates channel number from 0 to 2.

2.4. Memory Interface (Write Use)

Signal Name	IO	Pol	Source	Description
mbWrReq	O	+	clk	• Request signal
mbWrGnt	I	+	clk	• Grant signal
mbWrNew	O	+	clk	• Transaction start signal
mbWrEnd	O	+	clk	• Transaction end signal
mbWrAddr[31:0]	O	+	clk	• Address signal
mbWrStrb	O	+	clk	• Write strobe
mbWrAck	I	+	clk	• Write acknowledge signal
mbWrData[31:0]	O	+	clk	• Write data signal
mbWrMask[3:0]	O	+	clk	• Write mask signal

2.5. Memory Interface (Parameter Read Use)

Signal Name	IO	Pol	Source	Description
meReq	O	+	clk	• Request signal
meGnt	I	+	clk	• Grant signal
meAddr[31:0]	O	+	clk	• Address signal
meStrb	O	+	clk	• Read strobe signal
meAck	I	+	clk	• Read acknowledge signal
meFlush	O	+	clk	• Read flush signal
meData[63:0]	I	+	clk	• Read data signal

2.6. Utility

Signal Name	IO	Pol	Source	Description
rstReq	O	+	clk	• Internal reset signal to reset the external system
rstAck	I	+	clk	• Acknowledge of rstReq
fReq	I	+	clk	• 1 clock early request against the miReq signal • Use to generate gate signal (for mc2)
pReq	O	+	clk	• 1 clock early request against the meReq signal

				• Use to generate gate signal (for mc2)
gate[8:0]	O	+	clk	• Gated clock control signal signifying condition of each internal block
gclk[8:0]	I	+	clk	• Gated clock
Clk	I	+	clk	• Clock
reset	I	+	clk	• Synchronous reset signal

3. Architecture and Operation Description

3.1. Architecture Overview

➢ The Pipeline Slice Scheduler (hereafter ***pss***) fetches the necessary context from memory, generates information fragments and coordinates, etc., and starts ***sigComp***. See also the specification on ***pss***. Note that since the connection interface is simple, using PSS is not mandatory. If not used, consider replacing the ***pss*** part with a custom core.

➢ The ***sigComp*** operates as a pipeline composed of four stages: Initiator, Filter, Modifier, and Mixer, as shown in Figure 1 .

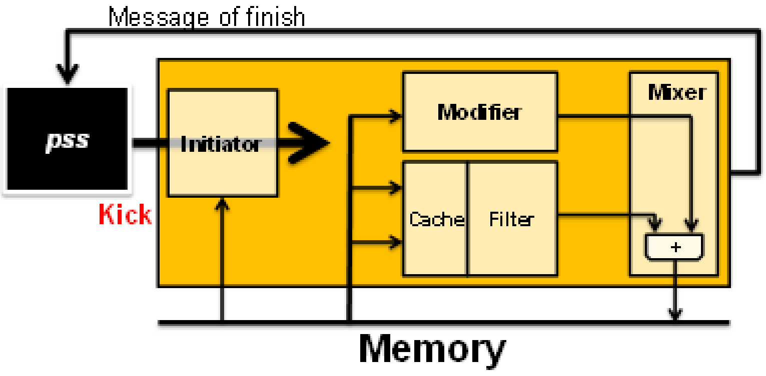


Figure 1 *sigComp* Block Diagram

3.2. Operational Overview

➢ The ***pss*** counts four-dimensional indices and sends the results to the Initiator. The settings for ***pss*** (voice information, processing units, etc.) are placed in memory in advance. ***pss*** manages up to 256 (depending on the implementation) multiple settings in a time-division manner, and drives ***sigComp*** after scheduling.

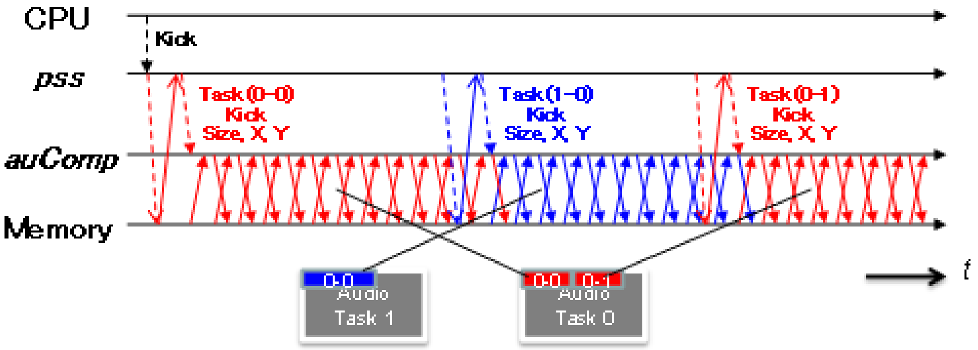


Figure 2 Resource sharing using pss

- The Initiator reads the context containing the processing information sent out by ***pss*** and performs the initial setup of the pipeline. Parameters extracted from the context are double buffer controlled, so there is no performance degradation unless the processing unit specified by ***pss*** is extremely short.

- It also converts the input 4-dimensional indices (16bit x4) into 1-dimensional coordinates. Assuming that the indicators are W, Z, Y, and X from the top to the bottom, the time axis information t is given by the following formula. ⊿ is given as a parameter. Normally, it should be the same as the Stride of each indicator specified in ***pss***. If ⊿ is all 0x10000, it is simply a 33-bit binary notation with W, Z, Y, and X in a row.

$$t = X + ⊿X (Y + ⊿Y(Z + ⊿Z \cdot W))$$

- Filter performs sequential type processing using multipliers and adders for order N. It takes N+1 cycles to obtain the filter result t() from the input data sig() and filter coefficients h(). The input data and filter coefficients are obtained from memory. The filter coefficients can be fixed to a single value.

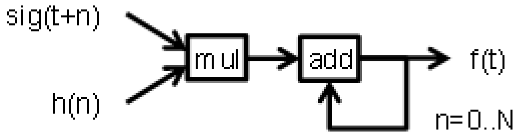


Figure 3 Convolution for filter

- The filter result is given by the general convolution formula:

$$f(t) = \sum_{n=0}^{N} h(n)sig(t+n)$$

Where

$$n = \mathrm{N} - \mathrm{n}'$$
$$t = t' - \mathrm{N}$$

then

$$f(t'-N) = \sum_{n'=0}^{N} \hat{h}(n')sig(t'-n')$$

This is a general convolution formula. It is important to note that the filter coefficients in the inverse order of h ˆ() are prepared as h(), and the output waveform is the result of N samples ahead.

- Depending on the configuration, it is possible to internally generate siĝ(), which inserts N zero-valued samples into the input signal.
  This allows the system to conform to the standard convolution expression.

$$f(t) = \sum_{n=0}^{N} h(n)\widehat{sig}(t+n)$$

Then、

$$\widehat{sig}(t-n) = \begin{cases} 0, & t-N+n<0 \\ sig(t-N+n), & t-N+n \geq 0 \end{cases}$$

If it is, then

$$f(t') = \sum_{n'=0}^{N} \hat{h}(n')\widehat{sig}(t'-n')$$

This means that standard convolution can be performed by using the input data as-is and preparing only the filter coefficients in reverse order.

- The data and coefficients input to the Filter are fetched from memory based on the sample position.
  The sample position t^ is calculated by applying a scaling ratio (numerator and denominator) to the time-axis information t.
  At the same time, the remainder R is used for interpolation.
  Both the numerator and denominator are 8-bit integers.

$$\hat{t} = t \cdot \frac{Numerator}{Denominator}$$

- From the sample position t^, the input data is retrieved, and based on the scaling numerator (Num) and remainder (R), the required number of coefficients (according to the filter order) are obtained.
  These are superimposed and output as the result for sample t^.
  Whether interpolation is needed (i.e., whether R is zero or not) can be determined at the configuration stage from the numerator and denominator, and the user should configure the coefficients accordingly.

- The superposition process described above is illustrated in Figure 4.
  As the time-axis information t increments, the remainder R changes accordingly.
  In response to this change, the coefficients are selected at intervals defined by the numerator of the scaling ratio.
  If both the numerator and denominator are 1, the remainder R is always 0, and the coefficients are selected sequentially without skipping — which corresponds to standard filtering.

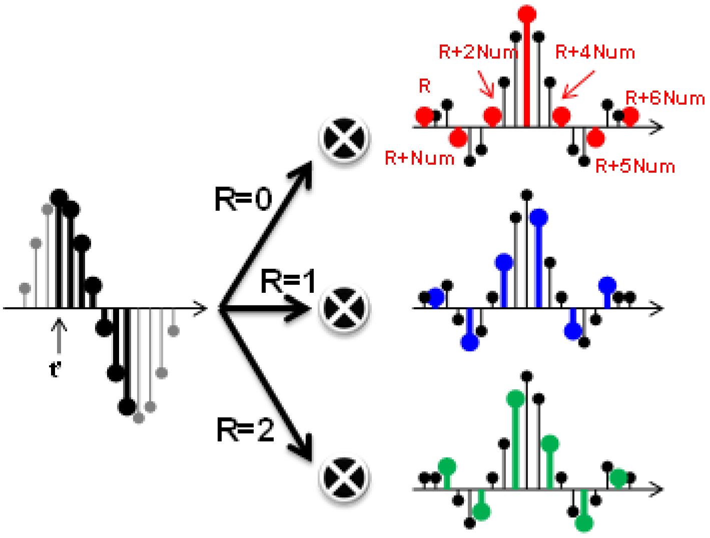


Figure 4 Convolution

- Since the input data and coefficients are reused repeatedly, they are cached to reduce unnecessary memory accesses.

  It is also possible to configure whether or not to synchronously flush the cache at the start of processing.

  However, if the processing is highly fragmented into small units, frequent flushing can diminish the effectiveness of caching.

  Please configure the auto-flush setting while considering the likelihood of updates to the input data or coefficients.

- The Filter outputs a new sample result, but it is also possible to accumulate the sample results sequentially.

  By setting the filter order N to 0, a simple accumulation result can be obtained.

$$f(t)' = \sum_{0}^{t} f(t)$$

- Modifier generates modulation signals used to manipulate the filtered signal. These modulation signals must be preloaded into memory.

- The modulation signal m(t) can be generated from any arbitrary sample position t for a specified duration and repeat count. The pre/post modulation values can be either 0.0 or 1.0 (default: 0.0).

- If repeating, take care with continuity between start and end. Without repetition, only continuity before and after the period matters. For example, when setting a sine wave, configure it to maintain waveform continuity.

  - Match modulation value to 1.0 or 0.0 before/after period.
  - Make it even-symmetric to begin and end at 0.0 or 1.0.

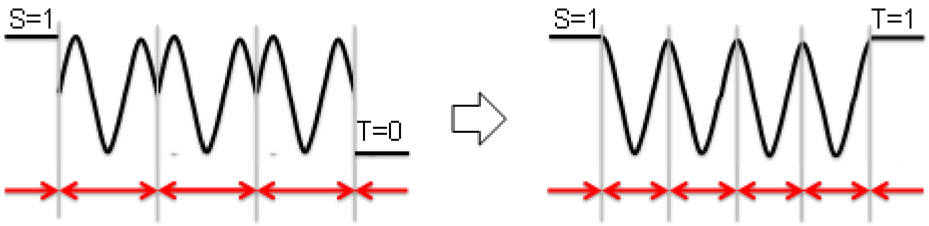


Figure 5 Continuous sin() Curve

- Mixer mixes the filtered result f(t) and two modulation signals m(t) using the following settings. Mag is the magnification constant, and Floor is the lower limit constant.

configuration			out(t)
Mix[3:0]			
0	0	00	f(t) + Mag * m(t)　　　（Synthesis）
0	0	01	m(t) + Mag * m(t)
0	0	10	Floor + Mag * m(t)
0	0	11	（refer to the last line and next section）
0	1	00	f(t) + f(t) * m(t)
0	1	01	m(t) + f(t) * m(t)
0	1	10	Floor + f(t) * m(t)　　　（modulation）
0	1	11	（refer to the last line and next section）
1	0	00	f(t) + Mag * Mag
1	0	01	m(t) + Mag * Mag
1	0	10	Floor + Mag * Mag
1	0	11	（refer to the last line and next section）
1	1	00	f(t) + f(t) * Mag
1	1	01	m(t) + f(t) * Mag
1	1	10	Floor + f(t) * Mag

x	x	11	DDA(f(t), Mag * t) （refer to next section） **Mix[2]** = Option A **Mix[3]** = Option B **→ A = B = 1 is prohibited**

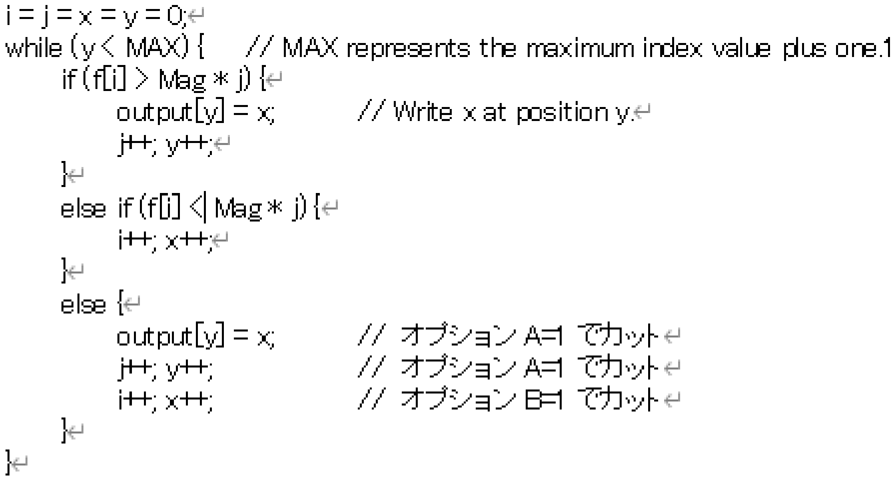
- The DDA process outputs a transposed result of the filter output f(t)f(t)f(t), based on the reference line Mag × t¥text{Mag} ¥times tMag × t.

  The function f(t)f(t)f(t) must be a monotonically increasing function.

  The transposed result is represented as a 32-bit index (integer).

  Below is a snippet of C code illustrating the conversion process.

  If both Option A and Option B are set to 1, and the condition f[i]==Mag × if[i] == ¥text{Mag} ¥times if[i]==Mag × i occurs, the process cannot proceed—therefore, this configuration is prohibited.



```
i = j = x = y = 0;↵
while (y < MAX) {    // MAX represents the maximum index value plus one.1
    if (f[i] > Mag * j) {↵
        output[y] = x;        // Write x at position y.↵
        j++; y++;↵
    }↵
    else if (f[i] < Mag * j) {↵
        i++; x++;↵
    }↵
    else {↵
        output[y] = x;        // オプション A=1 でカット↵
        j++; y++;             // オプション A=1 でカット↵
        i++; x++;             // オプション B=1 でカット↵
    }↵
}↵
```

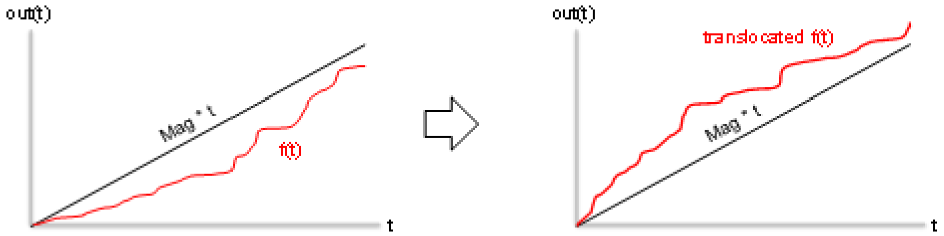


Figure 5 Translocated Signal↵

- You can apply clipping using constants Floor and Ceil during memory writeout of out(t).

3.3. Format

- The data stored in memory is supported in the following formats. Both single-channel and dual-channel (e.g., stereo audio: left-right packed) data are supported. However, for packed dual-channel data, only one channel (e.g., only L from LR) can be processed at a time, and write operations are not supported for packed data.

Bit/Word	**Type**	**Description**
16	Fixed Point	Value / 65536 (i.e., 16-bit unsigned fixed-point number)
32	Fixed Point	MSB is the sign (0: positive, 1: negative) LSB 23 bits represent the fractional value / 2^23 Unused bits are reserved. Equivalent to non-normalized IEEE754 Binary32.
16	Floating Point	IEEE754 Binary16 (half-precision). Does not support denormals, NaN, or Infinity
32	Floating Point	IEEE754 Binary32 (single-precision). Does not support denormals, NaN, or Infinity
32	Int	Result of DDA (Digital Differential Analyzer) in the Mixer stage

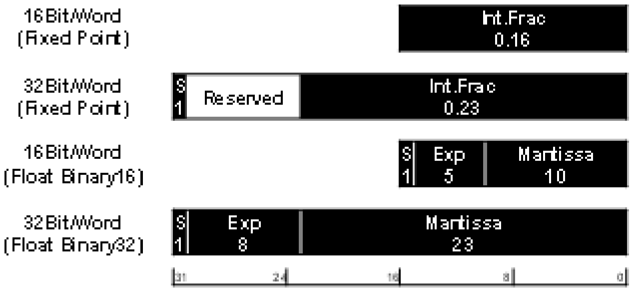


Figure 6 Signal Data Format

- In processing with two data streams packed together, the selection of the stream is determined by the base address configuration.
  This is based on the value (0 or 1) of the bit corresponding to the data word length.
  For 16-bit word length, this corresponds to bit number 1 (the second bit from the least significant bit).

3.4. Internal Computation

- All internal operations related to data processing within *sigComp* are performed in single-precision floating-point format.

  However, note that the rounding method differs from the IEEE754 standard's ″round to nearest, ties to even.″ Instead, *sigComp* adopts a round-half-up method (i.e., round up when ≥ 0.5), which deviates from the standard.

3.5. Connection with *PSS*

- The *sigComp* fetches the command list from memory based on the address (iAddr) output by the *PSS*.

  For details on the command list format, refer to the "Command List Description" section.

  If the *PSS* is not used, you can directly access the *sigComp* via the PSS interface signals.

- Processing for the length specified by the *PSS*' s transfer size (iDelta) is performed using the index values X, Y, Z, W, and C. Essentially, iDelta determines the processing unit granularity controlled by the *PSS*. The end of a line will naturally involve fractional processing.

- Smaller transfer sizes result in faster task switching, which enables more uniform service across tasks.

  However, if the filter order is high, this increases the likelihood that the internal cache (1K words) will be exceeded, which can lead to performance degradation.

  Frequent cache flushing due to task switching also contributes to this issue.

3.6. Performance

- The Filter stage requires N + 1 cycles for a filter order of N.

- Memory access wait times can stall the pipeline described above and cause performance degradation.

  There are two main causes of such wait-induced stalls:

  - Waits due to the ratio between absolute memory bus bandwidth and required bandwidth (i.e., contention on the memory bus being used)
  - Waits due to latency that cannot be absorbed—primarily the delay between a read Request

and its corresponding Acknowledge

4. Register Description

4.1. Overview

- All registers are accessed via the control bus.

- Some registers can affect the operation and performance of the pipeline, so care must be taken regarding when they are configured.

- In the detailed register descriptions, the following access types are used:

  - R – Read Only (writes have no effect)
  - R/W – Read and Write
  - R/WC – Read / Write-Clear

- Do not access reserved registers.
  For reserved fields, always write '0'.

- In address and data notations, values marked with 'x' indicate "don't care".

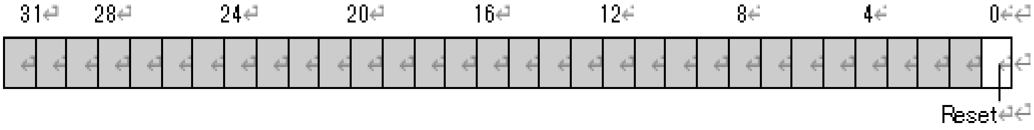
4.2. Definition

Address	**Register Name**	**Description**
0x0000_0000	Reset	Reset control
0x0000_0004	System	System control

4.3. Details

4.3.1.1. Reset Register

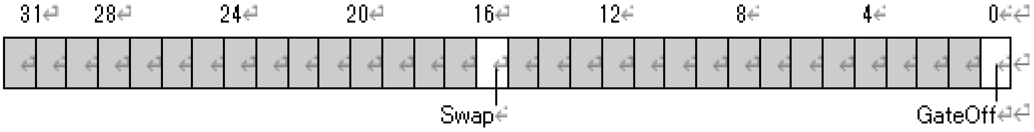
[Address: 0x0000_0000]



Name	Type	Default	Description
Reset	R/W	0	Synchronous reset. When set to '1', the module enters a reset state and automatically clears itself afterward. Unlike the reset_n signal, the contents of other registers are preserved. When set to '1', the internal signal rstReq is asserted immediately, notifying the external system that *sigComp* is in a reset state and requesting handling. Once the external handling is complete, it must assert the rstAck signal (if no special handling is required, assert '1' constantly). After the procedure is complete, Reset automatically returns to '0

4.3.1.2. System Register

[Address: 0x0000_0004]



Name	Type	Default	Description
Swap	R/W	0	Configures endian-ness for 32-bit words on a 64-bit memory bus. '0' = HL order, '1' = LH order. (This does not apply to the bus used for reading the Initiator's command list, which is always HL.)
GateOff	R/W	0	Gated Clock OFF mode. When set to '1', all bits of the gate signal are fixed to '1'.

5. Command List Description

5.1. Overview

- The command list is located in memory at the address output by the *PSS*. After *sigComp* starts up, it fetches this command list and stores it into internal registers.

- Each stage of the pipeline manages its own command list independently, allowing different stages to operate with different command configurations even while the pipeline is running. As a result, there is no need for synchronization commands.

- Reserved registers and fields must always be set to '0'.

- The addresses shown are relative to the address output by the *PSS*.

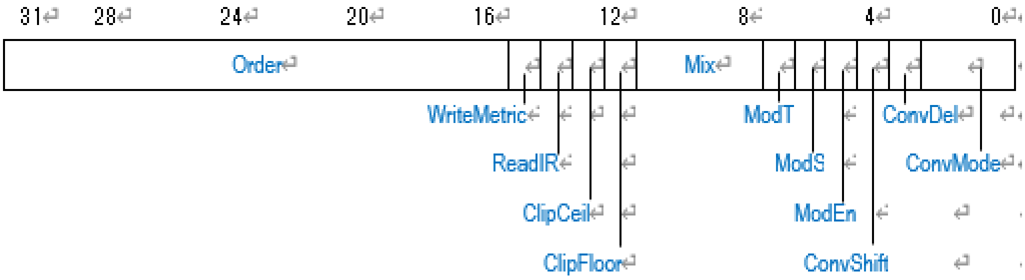
5.2. Definition

Address	**Command Name**	**Description**
00	MasterCntl	Master control
04	SamplingCntl	Sampling control
08	IndexWidth0	Δ for indices X and Y
0C	IndexWidth1	Δ for indices Z and W
10	MixMag	Mixer multiplication constant
14	MixFloor	Mixer lower limit constant
18	MixCeil	Mixer upper limit constant
1C	IRBase	Base address for filter coefficients
20	SrcInfo	Source data info
24	SrcBase	Source data base address
28	DstInfo	Destination data info
2C	DstBase	Destination data base address
30	ModInfo	Modulation data info
34	ModBase	Modulation data base address
38	ModPeriod	Modulation period and repeat count
3C	ModOffset	Modulation data offset

5.3. Details

5.3.1.1. MasterCntl Command

[Address: 0x00]



Name	**Description**
Order[15:0]	Sets the filter order N. The actual number of overlapping operations is N + 1. Set to 0 to disable filtering.
WriteMetric	If set to 1, the write results' minimum and maximum values are written sequentially using the address in MixCeil. The values of MixFloor and MixCeil are used as 0 in this mode.
ReadIR	If 1, filter coefficients are read from memory. If 0, the coefficient is taken from IRBase as a fixed value.
ClipCeil	If 1, output values are clipped to not exceed the upper limit defined in MixCeil.
ClipFloor	If 1, output values are clipped to not fall below the lower limit defined in MixFloor.
Mix[3:0]	Sets the Mixer mode (see detailed table below).

Mix[3:0]			**Mixer Output (out(t))**
0	0	00	f(t) + MixMag ∗ m(t)
0	0	01	m(t) ∗ (MixMag + 1)
0	0	10	MixFloor + MixMag ∗ m(t)
0	0	11	(refer to the last line)
0	1	00	f(t) ∗ (m(t) + 1)
0	1	01	m(t) ∗ (f(t) + 1)
0	1	10	MixFloor + f(t) ∗ m(t)
0	1	11	(refer to the last line)
1	0	00	f(t) + MixMag^2
1	0	01	m(t) + MixMag^2

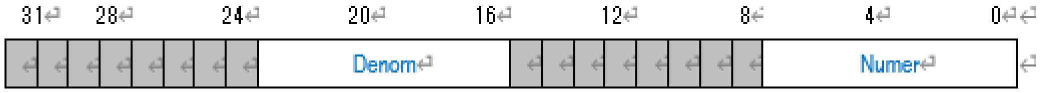
1	0	10	MixFloor + MixMag^2
1	0	11	Lust line reference
1	1	00	f(t) * (MixMag + 1)
1	1	01	m(t) + f(t) * MixMag
1	1	10	MixFloor + f(t) * MixMag
x	x	11	DDA(f(t), Mag * t) Mix[2] → Option A Mix[3] → Option B Setting both A and B to 1 is prohibited. If both are set to 1, the system forces both bits to 0

ModT	Sets the modulation value after the modulation period ends: 0 → 0.0, 1 → 1.0
ModS	Sets the modulation value before the modulation period starts: 0 → 0.0, 1 → 1.0
ModEn	If 1, modulation data is read from memory. If 0, modulation is based on the time variable t.
ConvShift	If 1, inserts Order − 1 zero samples before processing the input data.
ConvDel	If 1, delays the filter result by one sample. The first output becomes 0. (i.e., {a,b,c,d…} → {0,a,b,c,d…})
ConvMode[2:0]	Sets the **convolution mode** (see table below).

ConvMode[2:0]			**Operation Type**
0	0	0	Sum over filter order (standard conv)
0	0	1	Product over filter order
0	1	0	Minimum over filter order
0	1	1	Maximum over filter order
1	0	0	Cumulative sum over all samples
1	0	1	Cumulative product over all samples
1	1	0	Minimum over all samples
1	1	1	Maximum over all samples

5.3.1.2. SamplingCntl Command

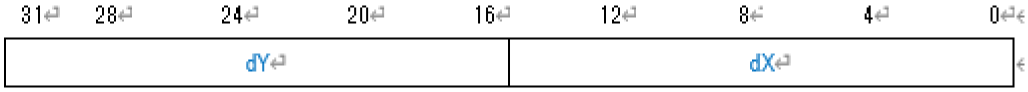
[Address: 0x04]



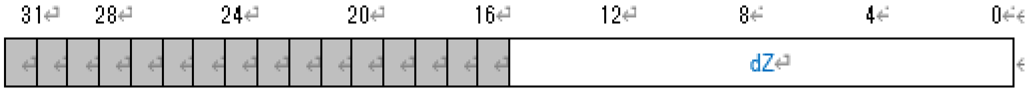
Name	Description
Denom[7:0]	Sets the denominator minus 1 for the sampling ratio. After oversampling using Numer, Denom + 1 samples are generated and Denom of them are selected. In practice, only the necessary values are computed based on Numer and Denom.
Numer[7:0]	Sets the numerator minus 1 for the sampling ratio. A single sample is expanded into Numer + 1 samples.

5.3.1.3. IndexWidth0,1 Command

[Address: 0x08]



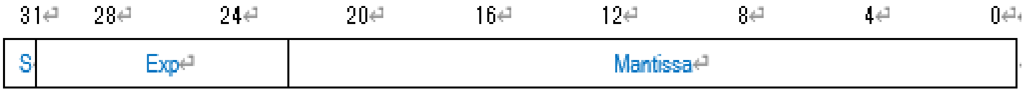
[Address: 0x0c]



Name	Description
dX[15:0], dY[15:0], dZ[15:0]	When the iIndex[64:0] signal is set to MSB as indicators W[15:0], Z[15:0], Y[15:0], and X[15:0], the update range for each is −1. The one−dimensional indicator T used internally is calculated using the following formula. T = X+(dX+1)(Y+(dY+1)(Z+(dZ+1)W))

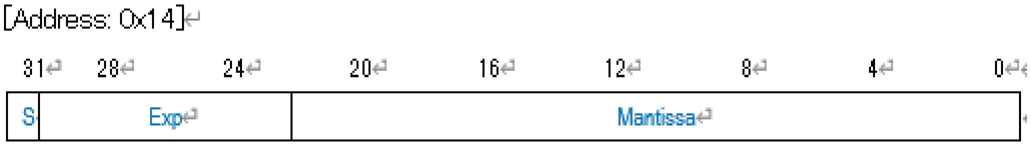
5.3.1.4. MixMag Command

[Address: 0x10]



Name	**Description**
S,Exp,Mantissa	Multiplier coefficient for single-precision floating-point representation used in the mixer.
S	Sets the sign of the single-precision floating-point format.
Exp[7:0]	Sets the exponential of the single-precision floating-point format. 0x00 and 0xff are not supported.
Mantissa[22:0]	Sets the mantissa of the single-precision floating-point format.

5.3.1.5. MixFloor Command

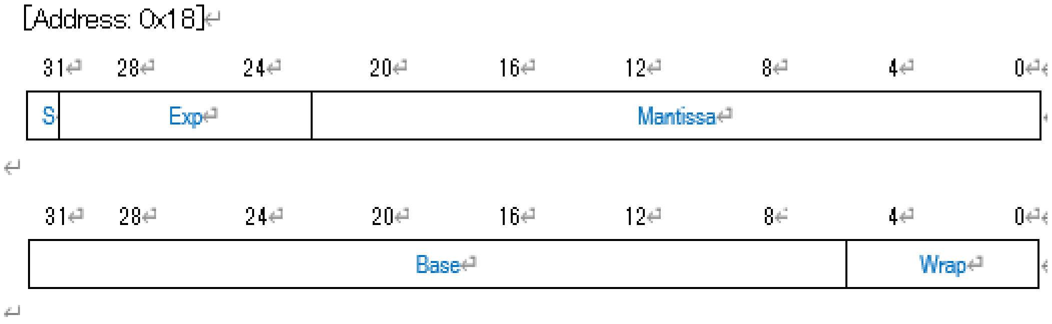


[Address: 0x14]

31	28	24	20	16	12	8	4	0
S	Exp		Mantissa					

Name	Description
S,Exp,Mantissa	Offset value and lower limit value for clipping of single-precision floating-point representation used in the mixer. When MasterContl.Mix[1:0] is ‘3’ for DDA processing, this becomes the lower limit value for clipping of 32-bit unsigned integers (in this case, set the value as an integer rather than a floating-point number).
S	Set the Sign in single-precision floating-point format.
Exp[7:0]	Sets the Exponential for single precision floating point format; 0x00, 0xff are not supported.
Mantissa[22:0]	Sets the mantissa of the single-precision floating-point format.

5.3.1.6. MixCeil Command



[Address: 0x18]

31	28	24	20	16	12	8	4	0
S	Exp		Mantissa					

31	28	24	20	16	12	8	4	0
Base							Wrap	

Name	Description

MasterCntl.WriteMetric = '0':

S,Exp,Mantissa	Upper limit for clipping of single-precision floating-point representation used in Mixer; for DDA processing with MasterContl.Mix[1:0] set to '3', this is the upper limit for clipping of 32-bit unsigned integers. In this case, it should be set in integer notation, not floating point.
S	Set the Sign in single precision floating point format
Exp[7:0]	Sets the Exponential for single precision floating point format; 0x00, 0xff are not supported.
Mantissa[22:0]	Set the Mantissa in the single precision floating point format

MasterCntl.WriteMetric=' 1' :

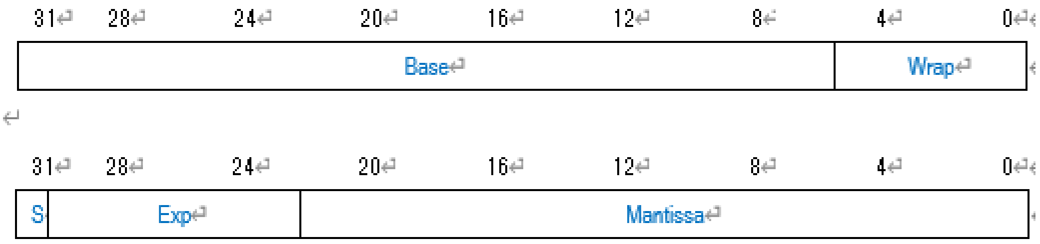
The upper limit is 0, which replaces the following meaning

Base[31:6]	Sets the base address of the input data in 64Byte boundary units.
Wrap[5:0]	By setting MSB to '1', the mask position becomes valid for the remaining 5 bits. The mask is masked up to the specified position of +1 counting from the MSB of the address. For example, if the address is 37, the upper 6 bits of the generated address are preserved as the upper 6 bits of Base, and only the remaining lower 26 bits are changed.

Wrap	Description
0	No mask
1-31	Reserved
32-62	Generated address [62-n:0] is valid
63	Address value is always the same as Base

5.3.1.7. IRBase Command

[Address: 0x1 c]



Name	Description

MasterCntl.ConvIR=' 0' :

Base[31:6]	Sets the base address of the coefficient data in 64Byte boundary units.
Wrap[5:0]	By setting MSB to '1', the mask position becomes valid for the remaining 5 bits. The mask is masked up to the specified position of +1 counting from the MSB of the address. For example, if the address is 37, the upper 6 bits of the generated address are preserved as the upper 6 bits of Base, and only the remaining lower 26 bits are changed.

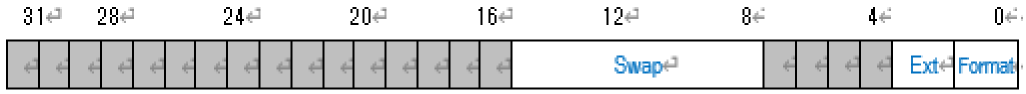
Wrap	Description
0	No mask
1-31	Reserved
32-62	The generated address [62-n:0] is valid.
63	Address value is always the same as Base

MasterCntl.ConvIR=' 1' :

S,Exp,Mantissa	Fixed filter coefficients in single-precision floating-point representation.
S	Set the Sign in single-precision floating-point format.
Exp[7:0]	Sets the Exponential for single precision floating point format; 0x00, 0xff are not supported.
Mantissa[22:0]	Set Mantissa in single precision floating point format.

5.3.1.8. SrcInfo Command

[Address: 0x20]



Name	Description
Swap[7:0]	Configures the byte swap setting for the input data. Maps input data In[31:0] to output data Out[31:0]. Each byte of Out[31:0] is controlled using 2-bit fields in Swap[7:0]. In principle, a one-to-one mapping should be used — otherwise, undefined or overlapping lanes may occur.

Swap 2bit Value	Swap[7:6]	Swap[5:4]	Swap[3:2]	Swap[1:0]
	Out[31:24]	Out[23:16]	Out[15:8]	Out[7:0]
0	In[31:24]	In[23:16]	In[15:8]	In[7:0]
1	In[7:0]	In[7:0]	In[23:16]	In[15:8]

2	In[15:8]	In[15:8]	In[31:24]	In[23:16]
3	In[23:16]	In[23:16]	In[7:0]	In[31:24]

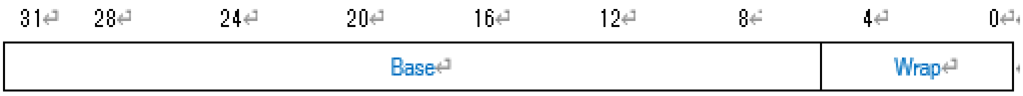
Exp[1:0] Set the data format details of the input data.

Format[1:0] Set the data format Bpw (Bit/Word) of the input data.

Format	Exp	Description	Note
0 8Bpp	–	Reserved	–
1 16Bpp	0	16bit Fixed Point Data	0.16
	1	16bit Fixed Point Data x2	Packed
	2	16bit Floating Point Data	IEEE754 Binary16
	3	16bit Floating Point Data x2	Packed
2 24Bpp	–	Reserved	–
3 32Bpp	0	32bit Fixed Point Data	Sign+0.23
	1	32bit Fixed Point Data x2	Packed
	2	32bit Floating Point Data	IEEE754 Binary32
	3	32bit Floating Point Data x2	Packed

5.3.1.9. SrcBase Command

[Address: 0x24]



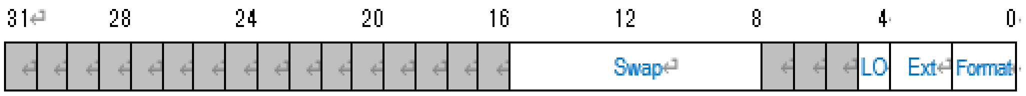
Name	Description
Base[31:6]	Sets the base address of the input data. Must be configured in units of 64-byte alignment.
Wrap[5:0]	By setting the MSB to '1', the remaining 5 bits become valid as the mask position. The mask applies from the MSB of the address down to the specified position (inclusive). For example, if the value is 37, the upper 6 bits of the generated address remain the same as those of Base, and only the lower 26 bits are allowed to change.

Wrap	Description
0	No mask

1–31	Reserved
32–62	The generated address bits [62－n：0] are valid.
63	Address value is always the same as Base

5.3.1.10. DstInfo Command

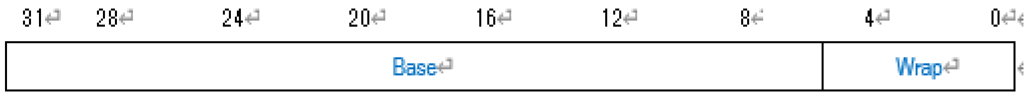
[Address: 0x28]



Name	Description
Swap[7:0]	Configures the byte swap setting for the output data (see SrcInfo).
LO	When set to '1', only the final data will be written.
Exp[1:0]	Configures the detailed format of the output data (see SrcInfo).
Format[1:0]	Configures the Bits per Word (Bpw) format of the output data (see SrcInfo).

5.3.1.11. DstBase Command

[Address: 0x2c]

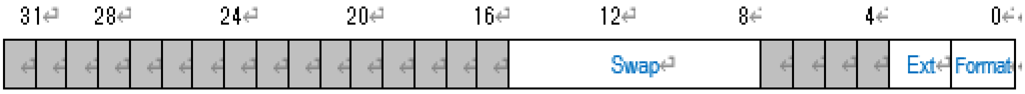


Name	Description
Base[31:6]	Sets the base address of the output data. Must be configured in units of 64-byte alignment.
Wrap[5:0]	By setting the MSB to '1', the remaining 5 bits become valid as the mask position. The mask applies from the MSB of the address down to (and including) the specified bit position. For example, if the value is 37, the upper 6 bits of the generated address remain the same as those of Base, and only the lower 26 bits can vary.

Wrap	Description
0	No mask
1–31	Reserved
32–62	The generated address bits [62－n：0] are valid.
63	Address value is always the same as Base

5.3.1.12. ModInfo Command

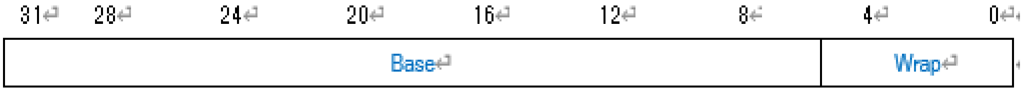
[Address: 0x30]



Name	Description
Swap[7:0]	Configures the byte swap setting for the modulation data (see SrcInfo).
Exp[1:0]	Configures the detailed data format of the modulation data (see SrcInfo).
Format[1:0]	Configures the Bits per Word (Bpw) format of the modulation data (see SrcInfo).

5.3.1.13. ModBase Command

[Address: 0x34]

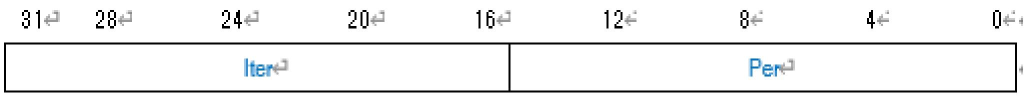


Name	Description
Base[31:6]	Sets the base address of the output data. Must be configured in units of 64-byte alignment.
Wrap[5:0]	By setting the MSB to '1', the remaining 5 bits become valid as the mask position. The mask applies from the MSB of the address down to (and including) the specified bit position. For example, if the value is 37, the upper 6 bits of the generated address remain the same as those of Base, and only the lower 26 bits can vary.

Wrap	Description
0	No mask
1-31	Reserved
32-62	The generated address bits [62－n：0] are valid.
63	Address value is always the same as Base

5.3.1.14. ModPeriod Command

[Address: 0x38]



Name	Description
Iter[15:0]	Specifies the number of repetitions for the modulation data. For example, if set to '2', the period defined by Per will repeat twice starting from the position indicated by ModOffset. A value of '0' indicates an infinite duration.
Per[15:0]	Specifies the period of the modulation data in number of samples. A value of '0' indicates an infinite period.

5.3.1.15. ModOffset Command

[Address: 0x3c]

Name	Description
Offset[31:0]	Specifies the starting position of the modulation data, based on the 1-dimensional index t.

6. Application Notes

6.1. Handling Data

6.1.1. Packed Data

- When using formats that pack two words per entry (e.g., stereo audio: L/R), only one of the two channels can be processed at a time. Therefore, such data must be processed in two separate passes.
- To select which word in the two-word packed data is processed, you must adjust the lower address bits of the Base address:
  - For example, with 16-bit LR packed data (L = upper word, R = lower word), set:
    - SrcBase[1] = 0 → process Right
    - SrcBase[1] = 1 → process Left

6.2. Filter operation

6.2.1. Superposition with Arbitrary Coefficients

Please refer to the following for the basic settings.

Number of operations driving sigComp	N	For PSS, set ΔX to N－1.
MasterCntl.Order	M	Order M (the number of coefficients is M－1)
MasterCntl.ReadIR	1	Using coefficients
MasterCntl.ClipCeil	0/1	Set to 1 if you want to clip the result at the upper limit.
MasterCntl.ClipFloor	0/1	Set to 1 if you want to clip the result at the lower limit.
MasterCntl.Mix	0	Filter result + MixMag × modulation data
MasterCntl.ModEn	0	No modulation data access
MasterCntl.ConvDel	0	No delayed output
MasterCntl.ConvMode	0	Sum (standard convolution)
IRBase	Address	Starting address of the coefficients
other	Set appropriately	Set all non-memory-related settings to 0.

6.3. Scalar Processing of a Sequence

6.3.1. Sum (∑), Product (Π), Minimum, and Maximum

- Please refer to the following for the basic settings

Number of operations driving sigComp	N	For PSS, set **ΔX to N－1**.
MasterCntl.Order	0	No filter
MasterCntl.ReadIR	0	No coefficients

MasterCntl.ClipCeil	0	No upper limit
MasterCntl.ClipFloor	0	No lower limit
MasterCntl.Mix	0	Filter result + MixMag × modulation data
MasterCntl.ModEn	0	No modulation data access
MasterCntl.ConvDel	0	No delayed output
MasterCntl.ConvMode	4–7	Use 4 for Σ, 5 for Π, 6 for minimum, and 7 for maximum.
IRBase	0x3f800000	ed coefficient of 1.0
DstInfo.LO	1	Output only the final result
other	Set appropriately	Set all non-memory-related settings to 0.

6.4. Generating an Inverse Lookup Table from a Cumulative Distribution Function (CDF)

6.4.1. Creating a Cumulative Distribution Function (CDF)

- It can be obtained by sequentially adding N probability distribution function (PDF) values stored in memory and outputting the result at each step.

- Please refer to the following for the basic settings

Number of processing steps to drive sigComp	N	For PSS, set **ΔX to N－1**.
MasterCntl.Order	0	No filter
MasterCntl.ReadIR	0/1	Multiply the value of IRBase by the probability density function (PDF). If no correction is needed, set IRBase = 1.0.
MasterCntl.ClipCeil	1	The cumulative value of a probability distribution is basically 1.0, but since it may exceed that, it should be configured accordingly.
MasterCntl.ClipFloor	0	No lower limit
MasterCntl.Mix	0	Filter result + MixMag × modulation

		data
MasterCntl.ModEn	0	No modulation data access
MasterCntl.ConvDel	0/1	Set to 1 to start the cumulative distribution function (CDF) from 0, or set to 0 to start from the first value of the probability density function (PDF).
MasterCntl.ConvMode	4	Perform cumulative summation
MixMag	0	Exclude the influence of modulation data
MixCeil	1.0	Set the upper limit to 1.0
IRBase	1/ΣPDF	Configure the settings so that the cumulative value reaches 1.0.
other	Set appropriately	Set all non-memory-related settings to 0.

6.4.2. Creating an Inverse Lookup Table

- It is possible to create a cumulative distribution function and a reverse lookup table at the same time.Internally, the two are processed at the same time, but the two cannot be written to memory at the same time.

- Please refer to the following for the basic settings. Only the areas that differ from the creation of the cumulative distribution function are described.

MasterCntl.Mix	0	Set MixMag = 3 to enable DDA mode.
MixCeil	N-1	Set the maximum index value to N－1.