Revision History

Revision 3.0
Updated the contact information and added this revision history. Prior revision history, other than revision 1.0, is not available for this document.

Revision 1.0
First publication of this document.
Table of Contents

Introduction ............................................................................................................. 4

1 Functionality ........................................................................................................ 5
   Optimization for High Speed or Low Power ...................................................... 5
   Write Depth/Width and Read Depth/Width ...................................................... 5
   Single Clock (CLK) or Independent Write and Read Clocks (WCLK, RCLK) ....... 5
   Write Enable (WEN) ....................................................................................... 5
   Read Enable (REN) ....................................................................................... 5
   Pipeline for Read Data Output ........................................................................... 5
   Register Enable (RD_EN) ............................................................................... 6
   Synchronous Reset (RD_SRST_N) ................................................................... 6
   Asynchronous Reset (ARST_N) ....................................................................... 6
   Expose Write Byte Enables (WBYTE_EN) ..................................................... 6
   Error Correction Code (ECC) ......................................................................... 6
   RD Register Truth Table ................................................................................... 6

2 Internal Connections of the Configurator ............................................................ 7
   WEN Connections ......................................................................................... 7
   REN Connections .......................................................................................... 7
   WD Connections ............................................................................................ 8
   RD Logic ........................................................................................................ 8
   SB_CORRECT, DB_DETECT Logic .................................................................. 8

3 Implementation Rules .......................................................................................... 9
   Caveats for Two-Port Large SRAM generation ............................................... 9

4 RAM Content Manager ...................................................................................... 10
   Supported Formats ......................................................................................... 10
   RAM Content Manager Functionality ........................................................... 12
   MEMFILE (RAM Content Manager output file) .............................................. 13

5 Port Description .................................................................................................. 14

6 Parameters .......................................................................................................... 15
A Two-Port Large SRAM enables write access on one port and read access on the other port (Figure 1). The core configurator automatically cascades Large SRAM blocks to create wider and deeper memories by choosing the most efficient aspect ratio. It also handles the grounding of unused bits. The core configurator supports the generation of memories that have different Read and Write aspect ratios.

Two-Port Large SRAM is synchronous for read and write operations, setting up the addresses as well as writing and reading the data. The memory write and read operations are triggered at the rising edge of the clock. The address, data, write-enable, and read-enable inputs are registered.

An optional pipeline register is available at the read data port to improve the clock-to-out delay. When ECC is enabled, output flags are generated to indicate single-bit-correct and double-bit-detect.

In this document, we describe how you can configure a Two-Port Large SRAM instance and define how the signals are connected. For more details about the Two-Port Large SRAM, refer to the RTG4 User Guide.

Figure 1 • Two-Port Large SRAM Configurator
1 – Functionality

Optimization for High Speed or Low Power

Selecting High Speed results in a macro optimized for speed and area (width cascading).
Selecting Low Power results in a macro optimized for low power, but uses additional logic at the input and output (depth cascading). Performance for a low power optimized macro may be inferior to that of a macro optimized for speed.

Write Depth/Width and Read Depth/Width

The depth range for each port is 1-65536. The width range for each port is 1-7524.
The two ports can be independently configured for any depth and width. (Write Depth * Write Width) must equal (Read Depth * Read Width)

Single Clock (CLK) or Independent Write and Read Clocks (WCLK, RCLK)

The default configuration for Two-Port Large SRAM is a Single clock (CLK) to drive WCLK and RCLK with the same clock. Uncheck the Single clock checkbox to drive independent clocks (one each for Write and Read).
Click the waveform next to any of the clock signals to toggle its active edge.

Write Enable (WEN)

Asserting WEN writes the data WD into the RAM at the address WADDR on the next rising edge of WCLK. Unchecking the WEN option ties the signal to the active state and removes it from the generated macro. Click the signal arrow (when available) to toggle its polarity.

Read Enable (REN)

When there is no depth cascading, de-asserting REN holds the previous Read data (RD). When there is depth cascading, de-asserting REN will generate zeros on RD. The different behavior in these two scenarios stems from the fact that the component’s REN input is used to either drive the LS RAM block’s A_REN input or the read-port block select input (A_BLK) depending on the cascading configuration.
Asserting the REN reads the RAM at the read address RADDR onto the input of the RD register on the next rising edge of RCLK.
The default configuration for REN is unchecked, which ties the signal to the active state and removes it from the generated macro. Click the checkbox to insert that signal on the generated macro. Click the signal arrow (when available) to toggle its polarity.

Pipeline for Read Data Output

Click the Pipeline checkbox to enable pipelining for Read data (RD).
Turning off pipelining of Read data also disables the configuration options of the RD_EN, RD_SRST_N and ARST_N signals.
Register Enable (RD_EN)
The pipeline register for RD has an active high, enable input. The default configuration is to tie this signal to the active state and remove it from the generated macro. Click the signal’s checkbox to insert that signal on the generated macro. Click the signal arrow (when available) to toggle its polarity.

Synchronous Reset (RD_SRST_N)
The pipeline register for RD has an active low, synchronous reset input. The default configuration is to tie this signal to the inactive state and remove it from the generated macro. Click the signal's checkbox to insert that signal on the generated macro. Click the signal arrow (when available) to toggle its polarity.

Asynchronous Reset (ARST_N)
The pipeline register for RD has an active low, asynchronous reset input. The default configuration is to tie this signal to the inactive state and remove it from the generated macro. Click the signal's checkbox to insert that signal on the generated macro. Click the signal arrow (when available) to toggle its polarity.

Expose Write Byte Enables (WBYTE_EN)
When enabled, write byte enables (WBYTE_EN) are available as a top-level bus. Each bit of WBYTE_EN enables writing to an individual byte of data. When ECC is enabled, the state of the WBYTE_EN bits should be identical for each LSRAM block.

Error Correction Code (ECC)
Three options are available for ECC:
- Disabled
- Pipelined
- Non-Pipelined
When ECC is disabled, each port could be configured to either 36 bits, 18 bits or 9 bits width. Alternatively, both ports could be configured to 12 bits width.
When ECC is enabled (Pipelined or Non-Pipelined), both ports have word widths equal to 36 bits or 18 bits.

RD Register Truth Table
Table 1-1 describes the functionality of the control signals on the RD register.

<table>
<thead>
<tr>
<th>ARST_N</th>
<th>RCLK</th>
<th>RD_EN</th>
<th>RD_SRST_N</th>
<th>D</th>
<th>Qn+1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>Not rising</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Qn</td>
</tr>
<tr>
<td>1</td>
<td>↑</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>Qn</td>
</tr>
<tr>
<td>1</td>
<td>↑</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>↑</td>
<td>1</td>
<td>1</td>
<td>D</td>
<td>D</td>
</tr>
</tbody>
</table>
This chapter describes an example where a Two-Port LSRAM configuration generates a component with the following address widths.

- WADDR[WA_MSB:0]
- RADDR[RA_MSB:0]

Let $M$ be the width of the address on the write-port of each LSRAM block and $N$ be the width of the address on the read-port of each LSRAM block.

Let the decoder logic function be $\text{decode}(\text{addr}[j:k], i)$, where $0 \leq i < 2^{(j-k+1)}$.

Let $D$ be the depth of an LSRAM block in the array of blocks, starting at 0.

<table>
<thead>
<tr>
<th>LSRAM Block Port Depth x Width</th>
<th>$M$, $N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2Kx9</td>
<td>11</td>
</tr>
<tr>
<td>1Kx18</td>
<td>10</td>
</tr>
</tbody>
</table>

### WEN Connections

The WEN signal on the generated component is connected to the write port block select input (B_BLK) for each LSRAM block according to the block depth within the component and synchronized with WCLK.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$WA_{MSB} &lt; N$</td>
<td>WEN</td>
<td>1</td>
<td>1, $\text{decode}(\text{WADDR}[M:M], D%2)$</td>
</tr>
<tr>
<td>$WA_{MSB} = N$</td>
<td>WEN</td>
<td>1</td>
<td>decode(\text{WADDR}[M+1:M+1], (D/2)%2)</td>
</tr>
<tr>
<td>$WA_{MSB} = M+1$</td>
<td>WEN &amp; $\text{decode}(\text{WADDR}[WA_{MSB}:M+2], D/4)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$WA_{MSB} &gt; M+1$</td>
<td>WEN</td>
<td>$\text{decode}(\text{WADDR}[M+1:M+1], (D/2)%2)$</td>
<td></td>
</tr>
</tbody>
</table>

### REN Connections

The REN signal on the generated component is connected to the write port block select input (A_BLK) for each LSRAM block according to the block depth within the component and synchronized with RCLK.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$RA_{MSB} &lt; N$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>REN</td>
<td>Hold</td>
</tr>
<tr>
<td>$RA_{MSB} = N$</td>
<td>REN</td>
<td>1</td>
<td></td>
<td>decode(\text{RADDR}[N:N], D%2)</td>
<td>1</td>
</tr>
</tbody>
</table>
Note the different behavior when \( REN \) is de-asserted on the top-level generated component. When there is no depth cascading (\( RA_{MSB} < N \)), de-asserting \( REN \) will hold the previous read-data. When there is depth cascading (\( RA_{MSB} \geq N \)), de-asserting \( REN \) will generate zeros on the read-data. The different behavior in these two scenarios stems from the fact that the component’s \( REN \) input is used to either drive the LSRAM block’s \( A_{REN} \) input or the read-port block select input (\( A_{BLK} \)) depending on the cascading configuration.

### WD Connections

The \( WD \) bits on the generated component is partitioned into slices based on the width of the data on the write port of each LSRAM block. Each bit of \( WD \) is connected to all blocks in a slice at every depth and synchronized with \( WCLK \).

### RD Logic

The \( RD \) bits on the generated component is partitioned into slices based on the width of the data on the read port of each LSRAM block. Each bit of read-data from all blocks in a slice at every depth is OR’d together to generate a bit of \( RD \). The \( RD \) bits are synchronized with \( RCLK \) according to the following latency.

<table>
<thead>
<tr>
<th>ECC Pipeline</th>
<th>ECC</th>
<th>RD Pipeline</th>
<th>RD Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>1</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>1</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>2</td>
</tr>
</tbody>
</table>

### SB_CORRECT, DB_DETECT Logic

The \( SB\_CORRECT \) and \( DB\_DETECT \) outputs are synchronized with \( RCLK \) according to the above \( RD \) latency. The \( SB\_CORRECT \) flags of each LSRAM block are gated by its \( A\_BLK \) signals pipelined one more than the \( RD \) latency value and then OR’d together to generate the \( SB\_CORRECT \) output of the component. Similarly, the \( DB\_DETECT \) flags of each LSRAM block are gated by its \( A\_BLK \) signals pipelined one more than the \( RD \) latency value and then OR’d together to generate the \( DB\_DETECT \) output of the component. As a result, both the outputs are zero whenever \( REN \) is de-asserted.
3 – Implementation Rules

Caveats for Two-Port Large SRAM generation

- The core configurator only supports depth cascading up to 32 blocks.
- The software returns a configuration error for unsupported configurations.

Note

- All unused inputs must be grounded.
- ARST_N does not reset the memory contents. It resets only the pipeline register for RD.
- Writing to and reading from the same address is undefined and should be avoided. There is no collision prevention or detection. However, correct data is expected to be written into the memory.
The RAM Content Manager enables you to specify the contents of your memory so that you can avoid the simulation cycles required for initializing the memory, which reduces simulation runtime.

The RAM core generator takes away much of the complexity required in the generation of large memory that utilize one or more RAM blocks on the device. The configurator uses one or more memory blocks to generate a RAM matching your configuration. In addition, it also creates the surrounding cascading logic.

The configurator cascades RAM blocks in three different ways:

- Cascaded deep (e.g. 2 blocks of 1024x18 to create a 2048x18)
- Cascaded wide (e.g. 2 blocks of 1024x18 to create a 1024x36)
- Cascaded wide and deep (e.g. 4 blocks of 1024x18 to create a 2048x36, in a 2 blocks width-wise by 2 blocks depth-wise configuration)

Specify memory content in terms of your total memory size. The configurator must partition your memory file appropriately such that the right content goes to the right block RAM when multiple blocks are cascaded.

**Supported Formats**

The Microsemi implementation of these formats interprets data sets in bytes. This means that if the memory width is 7 bits, every 8th bit in the data set is ignored. Or, if the data width is 9, two bytes are assigned to each memory address and the upper 7 bits of each 2-byte pair are ignored.

The following examples illustrate how the data is interpreted for various word sizes:

For the given data: FF 11 EE 22 DD 33 CC 44 BB 55 (where 55 is the MSB and FF is the LSB)

For 32-bit word size:

- 0x22EE11FF (address 0)
- 0x44CC33DD (address 1)
- 0x000055BB (address 2)

For 16-bit word size:

- 0x11FF (address 0)
- 0x22EE (address 1)
- 0x33DD (address 2)
- 0x44CC (address 3)
- 0x55BB (address 4)

For 8-bit word size:

- 0xFF (address 0)
- 0x11 (address 1)
- 0xEE (address 2)
- 0x22 (address 3)
- 0xDD (address 4)
- 0x33 (address 5)
- 0xCC (address 6)
- 0x44 (address 7)
- 0xBB (address 8)
- 0x55 (address 9)

For 9-bit word size:

- 0x11FF -> 0x01FF (address 0)
- 0x22EE -> 0x00EE (address 1)
- 0x33DD -> 0x01DD (address 2)
- 0x44CC -> 0x00CC (address 3)
- 0x55BB -> 0x01BB (address 4)
Notice that for 9-bit, that the upper 7-bits of the 2-bytes are ignored.

**INTEL-HEX**

Industry standard file. Extensions are HEX and IHX. For example, file2.hex or file3.ihx.
A standard format created by Intel. Memory contents are stored in ASCII files using hexadecimal characters. Each file contains a series of records (lines of text) delimited by new line, ‘
’, characters and each record starts with a ‘:’ character. For more information regarding this format, refer to the Intel-Hex Record Format Specification document available on the web (search Intel Hexadecimal Object File for several examples).

The Intel Hex Record is composed of five fields and arranged as follows:

```
:llaaaaatt[dd...][cc]
```

Where:

- : is the start code of every Intel Hex record
- ll is the byte count of the data field
- aaaa is the 16-bit address of the beginning of the memory position for the data. Address is big endian.
- tt is record type, defines the data field:
  - 00 data record
  - 01 end of file record
  - 02 extended segment address record
  - 03 start segment address record (ignored by Microsemi SoC tools)
  - 04 extended linear address record
  - 05 start linear address record (ignored by Microsemi SoC tools)
- [dd...] is a sequence of n bytes of the data; n is equivalent to what was specified in the ll field
- cc is a checksum of count, address, and data

Example Intel Hex Record:

```
:030000002337A1E
```

**MOTOROLA S-record**

Industry standard file. File extension is S, such as file4.s
This format uses ASCII files, hex characters, and records to specify memory content in much the same way that Intel-Hex does. Refer to the Motorola S-record description document for more information on this format (search Motorola S-record description for several examples). The RAM Content Manager uses only the S1 through S3 record types; the others are ignored.

The major difference between Intel-Hex and Motorola S-record is the record formats, and some extra error checking features that are incorporated into Motorola S.

In both formats, memory content is specified by providing a starting address and a data set. The upper bits of the data set are loaded into the starting address and leftovers overflow into the adjacent addresses until the entire data set has been used.

The Motorola S-record is composed of 6 fields and arranged as follows:

```
Stllaaaaat[dd...][cc]
```

Where:

- S is the start code of every Motorola S-record
- t is record type, defines the data field
- ll is the byte count of the data field
- aaaa is a 16-bit address of the beginning of the memory position for the data. Address is big endian.
• [dd...] is a sequence of n bytes of the data; n is equivalent to what was specified in the ll field
• cc is the checksum of count, address, and data

Example Motorola S-Record:
S10a0000112233445566778899FFFA

RAM Content Manager Functionality

To open the RAM Content Manager, after specifying your RAM configuration (set your Read and Write Depth and Width), select the Initialize RAM for Simulation checkbox, and then click Customize RAM Content. The RAM Content Manager appears (Figure 4-1).

RAM Configuration

Write Depth and Write Width - As specified in the RAM core generator dialog box (not editable).
Read Depth and Read Width - As specified in the RAM core generator dialog box (not editable).
**Write Port View / Read Port View**

**Go To Address** - Enables you to go to a specific address in the manager. Each memory block has many addresses; it is often difficult to scroll through and find a specific one. This task is simplified by enabling you to type in a specific address. The number display format (Hex, Bin, Dec) is controlled by the value you set in the drop-down menu above the Address column.

**Address** - The Address column lists the address of a memory location. The drop-down menu specifies the number format for your address list (hexadecimal, binary, or decimal).

**Data** - Enables you to control the data format and data value in the manager. Click the value to change it. Note that the dialogs show all data with the MSB down to LSB. For example, if the row showed 0xAA_BB for a 16-bit word size, the AA would be the MSB and BB would be the LSB.

**Default Data Value** - The value given to memory addresses that have not been explicitly initialized (by importing content or editing manually). When changed, all default values in the manager are updated to match the new value. The number display format (Hex, Bin, Dec) is controlled by the value you set in the drop-down menu above the Data column.

**Reset All Values** - Resets the Data values.

**Import from File** - Opens the Import Memory Content dialog box; enables you to select a memory content file (Intel-Hex) to load. Intel-Hex file extensions are set to *.hex during import.

**OK** - Closes the manager and saves all the changes made to the memory and its contents.

**Cancel** - Closes the manager, cancels all your changes in this instance of the manager, and returns the memory back to the state it held before the manager was opened.

---

**MEMFILE (RAM Content Manager output file)**

Transfer of RAM data (from the RAM Content Manager) to test equipment is accomplished via MEM files. The contents of your RAM is first organized into the logical layer and then reorganized to fit the hardware layer. Then it is stored in MEM files that are read by other systems and used for testing.

The MEM files are named according to the logical structure of RAM elements created by the configurator. In this scheme the highest order RAM blocks are named CORE_R0C0.mem, where “R” stands for row and “C” stands for column. For multiple RAM blocks, the naming continues with CORE_R0C1, CORE_R0C2, CORE_R1C0, etc.

The data intended for the RAM is stored as ASCII 1s and 0s within the file. Each memory address occupies one line. Words from logical layer blocks are concatenated or split in order to make them fit efficiently within the hardware blocks. If the logical layer width is less than the hardware layer, two or more logical layer words are concatenated to form one hardware layer word. In this case, the lowest bits of the hardware word are made up of the lower address data bits from the logical layer. If the logical layer width is more than the hardware layer, the words are split, placing the lower bits in lower addresses. If the logical layer words do not fit cleanly into the hardware layer words, the most significant bit of the hardware layer words is not used and defaulted to zero. This is also done when the logical layer width is 1 in order to avoid having leftover memory at the end of the hardware block.
Table 5-1 lists the Two-Port Large SRAM signals in the generated macro.

**Table 5-1 • Two-Port Large SRAM Signals**

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Default Polarity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>In</td>
<td>Rising Edge</td>
<td>Single clock to drive both WCLK and RCLK</td>
</tr>
<tr>
<td>WD[]</td>
<td>In</td>
<td></td>
<td>Write data</td>
</tr>
<tr>
<td>WADDR[]</td>
<td>In</td>
<td></td>
<td>Write address</td>
</tr>
<tr>
<td>WEN</td>
<td>In</td>
<td>Active high</td>
<td>Write port enable</td>
</tr>
<tr>
<td>WCLK</td>
<td>In</td>
<td>Rising edge</td>
<td>Write clock</td>
</tr>
<tr>
<td>RCLK</td>
<td>In</td>
<td>Rising edge</td>
<td>Read clock</td>
</tr>
<tr>
<td>REN</td>
<td>In</td>
<td>Active high</td>
<td>Read data enable</td>
</tr>
<tr>
<td>RADDR[]</td>
<td>In</td>
<td></td>
<td>Read address</td>
</tr>
<tr>
<td>RD[]</td>
<td>Out</td>
<td></td>
<td>Read data</td>
</tr>
<tr>
<td>RD_EN</td>
<td>In</td>
<td>Active high</td>
<td>Read data register enable</td>
</tr>
<tr>
<td>RD_SRST_N</td>
<td>In</td>
<td>Active low</td>
<td>Read data register Synchronous reset</td>
</tr>
<tr>
<td>ARST_N</td>
<td>In</td>
<td>Active low</td>
<td>Read data register Asynchronous reset</td>
</tr>
<tr>
<td>SB_CORRECT</td>
<td>Out</td>
<td>Active High</td>
<td>Single-bit correct flag</td>
</tr>
<tr>
<td>DB_DETECT</td>
<td>Out</td>
<td>Active High</td>
<td>Double-bit detect flag</td>
</tr>
<tr>
<td>WBYTE_EN[]</td>
<td>In</td>
<td>Active High</td>
<td>Write Byte Enables (per byte)</td>
</tr>
</tbody>
</table>
6 – Parameters

Table 6-1 lists the Micro SRAM parameters in the generated macro.

**Table 6-1 • Two-Port Large SRAM Parameters**

<table>
<thead>
<tr>
<th>GENFILE Parameter</th>
<th>Configurator Parameter</th>
<th>Valid Range</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESIGN</td>
<td></td>
<td></td>
<td></td>
<td>Name of the generated macro</td>
</tr>
<tr>
<td>FAM</td>
<td></td>
<td></td>
<td></td>
<td>Target family</td>
</tr>
<tr>
<td>OUTFORMAT</td>
<td></td>
<td></td>
<td></td>
<td>Netlist format</td>
</tr>
<tr>
<td>LPMTYPE</td>
<td></td>
<td></td>
<td></td>
<td>Macro category</td>
</tr>
<tr>
<td>DEVICE</td>
<td></td>
<td></td>
<td></td>
<td>Target device: RT4G150</td>
</tr>
<tr>
<td>PTYPE</td>
<td></td>
<td>1, 2</td>
<td>1</td>
<td>1: Two-port</td>
</tr>
<tr>
<td>INIT_RAM</td>
<td>INIT_RAM</td>
<td>F, T</td>
<td>F</td>
<td>Initialize RAM for simulation</td>
</tr>
<tr>
<td>CASCADE</td>
<td>CASCADE</td>
<td>0, 1</td>
<td>0</td>
<td>0: Cascading for WIDTH or Speed 1: Cascading for DEPTH or Power</td>
</tr>
<tr>
<td>CLKS</td>
<td>CLKS</td>
<td>1, 2</td>
<td>1</td>
<td>1: Single Read/Write Clock 2: Independent Read and Write Clocks</td>
</tr>
<tr>
<td>WCLK_EDGE</td>
<td>CLK_EDGE</td>
<td>RISE, FALL</td>
<td>RISE</td>
<td>RISE: Rising edge Single clock FALL: Falling edge Single clock</td>
</tr>
<tr>
<td>WWIDTH</td>
<td>WWIDTH</td>
<td>1-7524</td>
<td>18</td>
<td>Write data width</td>
</tr>
<tr>
<td>WDEPTH</td>
<td>WDEPTH</td>
<td>1-65536</td>
<td>1024</td>
<td>Write address depth</td>
</tr>
<tr>
<td>RWIDTH</td>
<td>RWIDTH</td>
<td>1-7524</td>
<td>18</td>
<td>Read data output width</td>
</tr>
<tr>
<td>RDEPTH</td>
<td>RDEPTH</td>
<td>1-65536</td>
<td>1024</td>
<td>Read address depth</td>
</tr>
<tr>
<td>WE_POLARITY</td>
<td>WE_POLARITY</td>
<td>0, 1, 2</td>
<td>1</td>
<td>0: Active-low Write port enable 1: Active-high Write port enable 2: Write port enable tied off to be always active</td>
</tr>
<tr>
<td>WCLK_EDGE</td>
<td>WCLK_EDGE</td>
<td>RISE, FALL</td>
<td>RISE</td>
<td>RISE: Rising edge Write clock FALL: Falling edge Write clock</td>
</tr>
<tr>
<td>RCLK_EDGE</td>
<td>RCLK_EDGE</td>
<td>RISE, FALL</td>
<td>RISE</td>
<td>RISE: Rising edge Read clock FALL: Falling edge Read clock</td>
</tr>
<tr>
<td>RE_POLARITY</td>
<td>RE_POLARITY</td>
<td>0, 1, 2</td>
<td>2</td>
<td>0: Active-low Read port enable 1: Active-high Read port enable 2: Read port enable tied off to be always active</td>
</tr>
<tr>
<td>PMODE2</td>
<td>RPMODE</td>
<td>0, 1</td>
<td>0</td>
<td>0: Bypass Read data register 1: Pipeline Read data</td>
</tr>
<tr>
<td>A_DOUT_EN_POLARITY</td>
<td>A_DOUT_EN_POLARITY</td>
<td>PMODE2=1, 0, 1, 2</td>
<td>2</td>
<td>0: Active-low Read data register enable 1: Active-high Read data register enable 2: Read data register enable tied off to be always active</td>
</tr>
<tr>
<td>GENFILE Parameter</td>
<td>Configurator Parameter</td>
<td>Valid Range</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>------------------------</td>
<td>-------------</td>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>A_DOUT_SRST_POLARITY</td>
<td>A_DOUT_SRST_POLARITY</td>
<td>PMODE2=1</td>
<td>2</td>
<td>0: Active-low Read data register Sync-reset 1: Active-high Read data register Sync-reset 2: Read data register Sync-reset tied off to be always inactive</td>
</tr>
<tr>
<td>RESET_POLARITY</td>
<td>ARST_N_POLARITY</td>
<td>PMODE2=1</td>
<td>2</td>
<td>0: Active-low Read data register Async-reset 1: Active-high Read data register Async-reset 2: Read data register Async-reset tied off to be always inactive</td>
</tr>
<tr>
<td>ECC</td>
<td>ECC</td>
<td>0, 1, 2</td>
<td>0</td>
<td>0: ECC Disabled 1: ECC Pipelined 2: ECC Non-pipelined</td>
</tr>
<tr>
<td>CLOCK_PN</td>
<td>CLOCK_PN</td>
<td>CLKS=1</td>
<td>CLK</td>
<td>Single clock Port name</td>
</tr>
<tr>
<td>DATA_IN_PN</td>
<td>DATA_IN_PN</td>
<td>WD</td>
<td>WADDR</td>
<td>Write data Port name</td>
</tr>
<tr>
<td>WADDRESS_PN</td>
<td>WADDRESS_PN</td>
<td>WADDR</td>
<td>WEN</td>
<td>Write address Port name</td>
</tr>
<tr>
<td>WE_PN</td>
<td>WE_PN</td>
<td>WE_POLARITY=2</td>
<td>WEN</td>
<td>Write port enable Port name</td>
</tr>
<tr>
<td>WCLOCK_PN</td>
<td>WCLOCK_PN</td>
<td>CLKS=2</td>
<td>WCLK</td>
<td>Write clock Port name</td>
</tr>
<tr>
<td>RCLOCK_PN</td>
<td>RCLOCK_PN</td>
<td>CLKS=2</td>
<td>RCLK</td>
<td>Read clock Port name</td>
</tr>
<tr>
<td>RE_PN</td>
<td>RE_PN</td>
<td>RE_POLARITY=2</td>
<td>REN</td>
<td>Read port enable Port name</td>
</tr>
<tr>
<td>RADDRESS_PN</td>
<td>RADDRESS_PN</td>
<td>RADDR</td>
<td>Read address Port name</td>
<td></td>
</tr>
<tr>
<td>DATA_OUT_PN</td>
<td>DATA_OUT_PN</td>
<td>RD</td>
<td>RD</td>
<td>Read data Port name</td>
</tr>
<tr>
<td>A_DOUT_EN_PN</td>
<td>A_DOUT_EN_PN</td>
<td>PMODE2=1</td>
<td>RD_EN</td>
<td>Read data register enable Port name</td>
</tr>
<tr>
<td>A_DOUT_SRST_PN</td>
<td>A_DOUT_SRST_PN</td>
<td>PMODE2=1</td>
<td>RD_SRST_N</td>
<td>Read data register Sync-reset Port name</td>
</tr>
<tr>
<td>RESET_PN</td>
<td>RESET_PN</td>
<td>PMODE2=1</td>
<td>ARST_N</td>
<td>Read data register Async-reset Port name</td>
</tr>
<tr>
<td>BYTEENABLES</td>
<td>BYTEENABLES</td>
<td>0, 1</td>
<td>0</td>
<td>0: Don’t generate WBYTE_EN 1: Generate WBYTE_EN</td>
</tr>
<tr>
<td>COLLISION_WARN_MSGS</td>
<td>COLLISION_WARN_MSGS</td>
<td>-1, 0, &gt;0</td>
<td>-1</td>
<td>-1: All warning messages related to collisions will appear in the simulation log 0: One warning message related to collisions will appear in the simulation log &gt;0: More than one warning message equal to the integer value passed to the parameter related to collisions will appear in the simulation log.</td>
</tr>
</tbody>
</table>
A – Product Support

Microsemi SoC Products Group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, electronic mail, and worldwide sales offices. This appendix contains information about contacting Microsemi SoC Products Group and using these support services.

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From North America, call 800.262.1060
From the rest of the world, call 650.318.4460
Fax, from anywhere in the world, 650.318.8044

Customer Technical Support Center

Microsemi SoC Products Group staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions about Microsemi SoC Products. The Customer Technical Support Center spends a great deal of time creating application notes, answers to common design cycle questions, documentation of known issues, and various FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Technical Support


Website

You can browse a variety of technical and non-technical information on the Microsemi SoC Products Group home page, at www.microsemi.com/soc.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center. The Technical Support Center can be contacted by email or through the Microsemi SoC Products Group website.

Email

You can communicate your technical questions through our Web portal and receive answers back by email, fax, or phone. Also, if you have design problems, you can upload your design files to receive assistance. We constantly monitor the cases created from the web portal throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

For technical support on RH and RT FPGAs that are regulated by International Traffic in Arms Regulations (ITAR), log in at https://soc.microsemi.com/Portal/Default.aspx, go to the My Cases tab, and select Yes in the ITAR drop-down list when creating a new case. For a complete list of ITAR-regulated Microsemi FPGAs, visit the ITAR web page.

**My Cases**
Microsemi SoC Products Group customers may submit and track technical cases online by going to My Cases.

**Outside the U.S.**
Customers needing assistance outside the US time zones can either contact technical support at https://soc.microsemi.com/Portal/Default.aspx or contact a local sales office. Visit About Us for sales office listings and corporate contacts.

Sales office listings can be found at www.microsemi.com/soc/company/contact/default.aspx.

---

**About Microsemi**
Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for communications, defense & security, aerospace and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; Enterprise Storage and Communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif. and has approximately 4,800 employees globally. Learn more at www.microsemi.com.

Microsemi makes no warranty, representation, or guarantee regarding the information contained herein or the suitability of its products and services for any particular purpose, nor does Microsemi assume any liability whatsoever arising out of the application or use of any product or circuit. The products sold hereunder and any other products sold by Microsemi have been subject to limited testing and should not be used in conjunction with mission-critical equipment or applications. Any performance specifications are believed to be reliable but are not verified, and Buyer must conduct and complete all performance and other testing of the products, alone and together with, or installed in, any end-products. Buyer shall not rely on any data and performance specifications or parameters provided by Microsemi. It is the Buyer's responsibility to independently determine suitability of any products and to test and verify the same. The information provided by Microsemi hereunder is provided "as is, where is" and with all faults, and the entire risk associated with such information is entirely with the Buyer. Microsemi does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other IP rights, whether with regard to such information itself or anything described by such information. Information provided in this document is proprietary to Microsemi, and Microsemi reserves the right to make any changes to the information in this document or to any products and services at any time without notice.