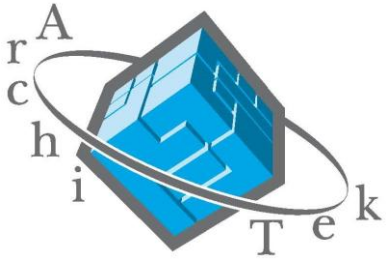


pss Specification

Pipeline Slice Scheduler

Revision 2.0

28 April 2025

English edition

1. Overview

1.1. Introduction

- The Pipeline Slice Scheduler (hereafter referred to as pss) is an embedded core that centrally controls multiple functional Pipelines. The Pipelines themselves are prepared by the user.

- Up to 256 logical channels (software interfaces) can be defined, which operate by time-division fragmentation to drive up to 16 physical channels (Pipelines). These numbers can be adjusted via implementation parameters.

- Arbitration of logical channels is performed to optimize the operation of physical channels. It can be controlled in such a way that up to 256 Pipelines appear to exist.

- Logical channel operation can be controlled by a simple statement such as DMA configuration. You can choose a dedicated register or an external memory such as main memory and set it up. In the case of external memory, pss automatically accesses it as needed.

- Logical channels can be linked by specifying a forward/backward relationship between them. By specifying the pre- and post-relationships, it is possible to control the order in which the logical channels are activated, etc. Many-to-one, one-to-many, and many-to-many relationships can be constructed.

- Index, an execution Index, can be expressed in up to four dimensions. This makes it possible to exchange data on a per-line basis, per-frame basis, and in different ratios.

- Logic channels are evaluated and executed sequentially, not simultaneously. Since the number of circuits operating simultaneously is limited, low cost and low power consumption are maintained even if the number of logic channels is increased.

Note:

- Pipelines require a specific interface that conforms to connection rules.
- The memory interface must be customized for each system.
- The number of logical and physical channels can be customized within a certain range.

1.2. Main Parameters

- Memory Bus – Logical Description Read: 64bit x 1
- throughput – Max. 0.5 schedule/cycle
- clock – Undefined (depends on implementation process)

1.3. Implementation Parameters

Parameter Name	Description	Default Value
CNR	• Radix of channel number	**8**(8or below)
PNR	• Radix of Pipeline number	**4**(4or below)
BLR	• Radix of burst length • Set burst unit for 64-bit memory access	**2**(4or below)

---

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	• Writ 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

2.2. Memory Interface

Signal Name	IO	Pol	Source	Description
memReq	O	+	clk	• Request signal
memGnt	I	+	clk	• Grant signal
memAddr[31:0]	O	+	clk	• Address signal
memStrb	I	+	clk	• Read strobe signal
memAck	I	+	clk	• Read acknowledge signal
memFlush	I	+	clk	• Read flush signal
memData[63:0]	I	+	clk	• Read data signal

2.3. Pipeline Input Interface

Signal Name	IO	Pol	Source	Description
pi_nVld	O	+	clk	• Valid signal
pi_nStall	I	+	clk	• Stall signal
pi_nRest	O	+	clk	• Restore signal • Indicates invalid frame and use previous frame
pi_nCID[**CNR**-1:0]	O	+	clk	• ID signall • Indicates the handling ID
pi_nEnd[3:0]	O	+	clk	• End signal • Indicates last fragment of each Index
pi_nAddr[31:0]	O	+	clk	• Address signal • Indicates an address of Pipeline contexts
pi_nDelta[15:0]	O	+	clk	• Delta signal • Indicates a length of Pipeline processing
pi_nIndex[64:0]	O	+	clk	• Index signal • Indicates the start of Indexes {C, W[15:0], Z[15:0], Y[15:0], X[15:0]}
pi_nOk	I	+	clk	• OK signal • Indicates buffer being able to push

Signal name subscript n is Pipeline number from 0 to $2^{\mathbf{PNR}}$-1

2.4. Pipeline Output Interface

Signal Name	IO	Pol	Source	Description
po_nVld	I	+	clk	• Valid signal
po_nStall	O	+	clk	• Stall signal

Signal name subscript n is Pipeline number from 0 to 2^{PNR}-1

2.5. Utility

Signal Name	IO	Pol	Source	Description
intVld[2^{CNR}-1:0]	I	+	clk	• Interrupt valid to activate logical channel [n] • Do not assert if parameter are not set
intStall[2^{CNR}-1:0]	O	+	clk	• Interrupt stall to activate logical channel [n] • Indicates the logical channel is busy
intOp[2^{CNR}-1:0]	I	+	clk	• Side signals of intSet inform restoring the Index
intClr[2^{CNR}-1:0]	I	+	clk	• Interrupt set to inactivate logical channel [n] • Do not assert if parameter is not set
fpReq[2^{PNR}-1:0]	I	+	clk	• 1 clock early request against the pi_nVld signal • Use to generate gate signal (for Pipeline)
ppReq[2^{PNR}-1:0]	O	+	clk	• 1 clock early request against the po_nVld signal • Use to generate gate signal (for Pipeline)
gate[2^{PNR}+12:0]	O	+	clk	• Gated clock control signal signifying condition of each internal block
gclk[2^{PNR}+12:0]	I	+	clk	• Gated clock
reset	I	+	clk	• Synchronous reset
clk	I	+	clk	• Clock
reset_n	I	-	clk	• Asynchronous reset

3. Architecture and Operation Description

3.1. Architecture Overview

➢ The PSS, as shown in Figure 1, consists of the following components: a control unit that

manages the input and output of logical channel states and parameters; an arbitration unit that selects an appropriate one from the 2CNR logical channels; each calculation unit that determines whether or not to execute 2PNR Pipelines according to logical channel parameters, each FIFO sections that queues the execution parameters of the Pipeline, and SRAM that stores the parameters.

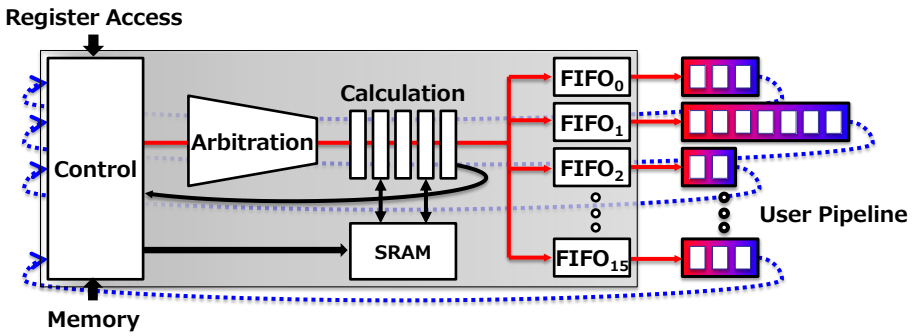


Figure 1 *pss* Block Diagram

- The process up to Pipeline activation involves selecting a logical channel (indicated by the red line) and issuing commands to each Pipeline.

- The completion signals from each Pipeline are returned to the PSS as shown by the blue dotted lines. This triggers an update of the information necessary to activate the next Pipeline.

- The PSS essentially functions like a pump, driving the Pipelines, which can be thought of as individual organs.

3.2. Operational Overview

- The PSS drives the Pipelines based on the configuration of logical channels. Each logical channel corresponds to one Pipeline. It is acceptable for multiple logical channels to specify the same Pipeline.

- The logical channels referenced by the PSS consist of a transfer size (Width) and an Index indicating the progress, similar to Direct Memory Access (DMA). The transfer size is provided by the user, while the Index is generated by the PSS. Note that the Width should be set to the transfer size minus one.

- The fragment size (Delta), which determines the counting unit of the Index, is also specified by the user. It is acceptable to configure values that result in remainders or exceed the Width. For example, if Width is 10 and Delta is 4, the Index changes as 0, 4, 8, 10 (10 wraps around and effectively becomes 0). If Width is 10 and Delta is 16, the Index changes as 0, 10. Note that Delta should be set to the fragment size minus one.

- The Index is 4+1 dimensional and increments in order from the lowest to highest dimension: X, Y, Z, W, and C. Transfer sizes are configured for four dimensions. For example, using the dimension symbols as subscripts, if $Width_{X, Y, Z}$, and $_W$ are set to 4, 3, 2, and 1 respectively, then $Index_X$ resets to 0 just before reaching 4, and $Index_Y$ is incremented. The other Indexes similarly carry over in order. However, $Index_C$ is derived from the carry-over of $Index_W$ and does not have a transfer size setting. Additionally, when a carry occurs, the value is inverted.

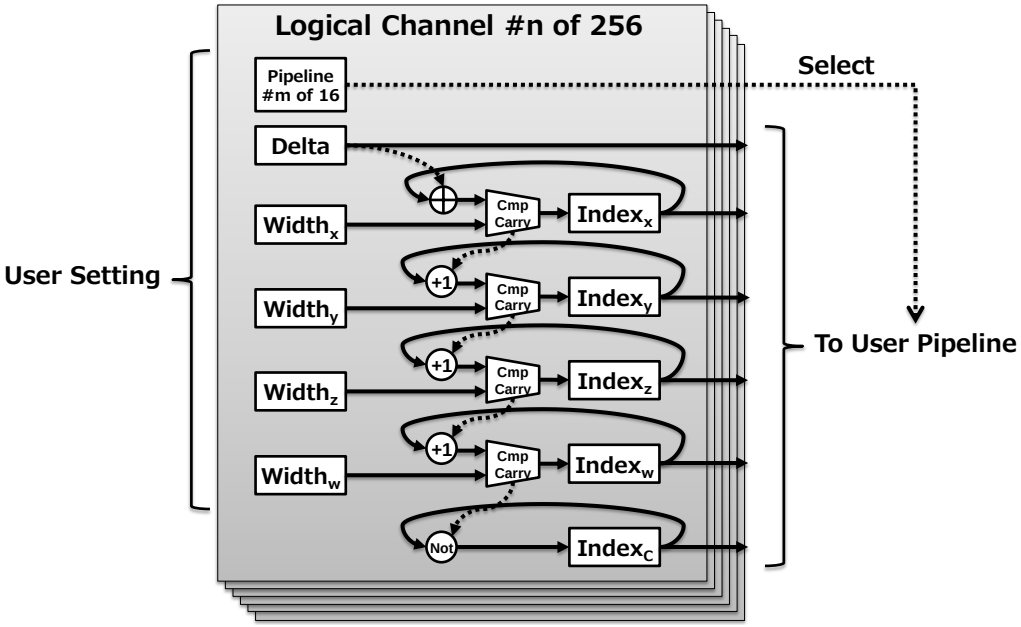


Figure 2 Index Count of each Logical Channel

- Each Index, except for $Index_C$, can count up to 16 bits. When combined, the Indexes can represent a count of up to 64 bits. If a transfer size is set to '0', that dimension is omitted. For example, if only $Width_X$ is non-zero, the transfer is treated as one-dimensional. These Indexes are automatically reset to '0' at the initial activation of the logical channel.

- $Index_C$ is 2 bits wide and is used to manage double buffering of frame processing, where a frame is defined by $Index_X$ through $Index_W$. It is not necessarily reset to '0' at the initial

activation of a logical channel. However, it can be optionally cleared during the automatic loading of parameters, which will be described later.

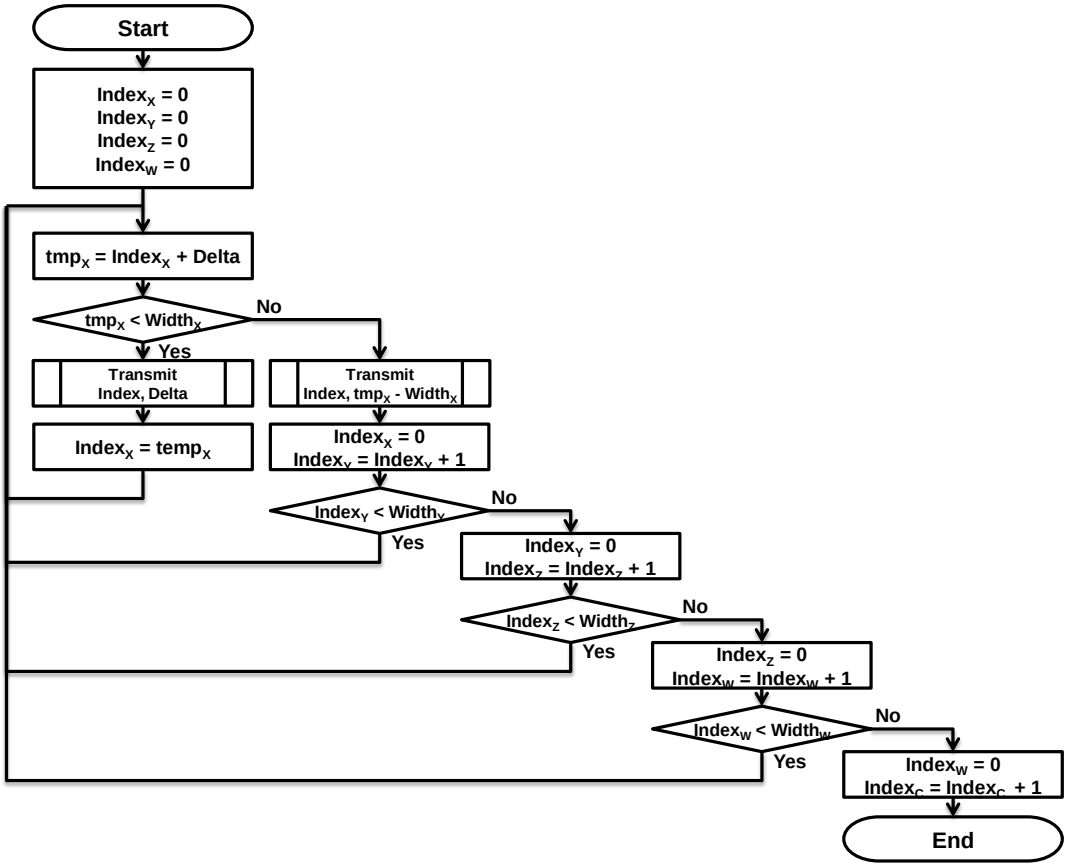


Figure 3 Index Count Flowchart

- Each logical channel must be activated to become effective. Activation can be performed in one of three ways: by register configuration, by external interrupt, or by automatic parameter loading. Details of each method will be described in the register descriptions.

- It is also possible to inactivate a logical channel. This can only be done through register configuration. The values of parameters such as the Index are preserved, allowing the channel to be resumed later.

3.3. Arbitration of Logical Channels

- pss performs arbitration to select one of the 2^{CNR} logical channels. This arbitration occurs in **two stages** to ensure that execution timing across Pipelines remains balanced. Arbitration

takes **two cycles**, which defines the maximum driving capacity of pss (i.e., in a system with 256 logical channels, each DMA can be scheduled approximately once every 512 cycles).

- **First Arbitration Stage**

  Among the activated logical channels (CID: 0 to $2^{CNR} - 1$), those with matching Pipeline ID (PID: 0 to $2^{PNR} - 1$) are selected as candidates. And finaly a round-robin selection is performed from the candidates. A pointer is incremented. Logical channels without a PID assignment are skipped.

- **Second Arbitration Stage**

  In the next arbitration, the CID selected above is determined in a round robin prepared for each PID. The selection is made by choosing the CID closest to the pointer in ascending order. The pointer is automatically incremented. However, it is possible to disable this increment for each Pipeline. If not incremented, the logical channel will always be executed as long as the corresponding logical channel is Active. This behavior is useful when a Pipeline should not share resources, such as a display controller.

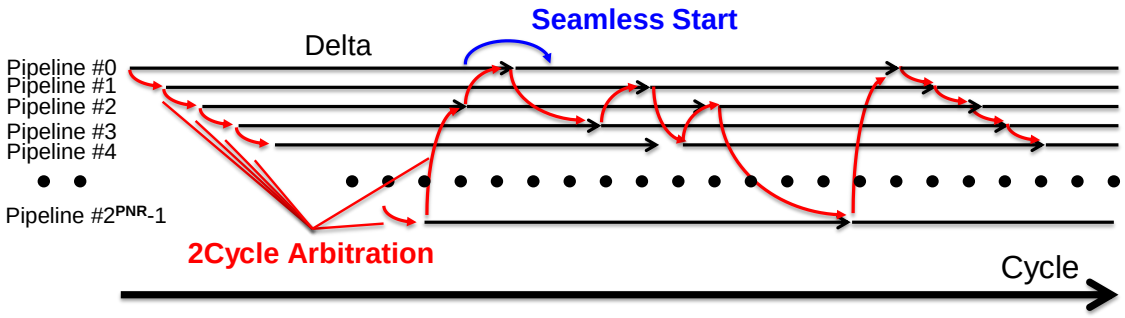


Figure 4 Arbitration of Logical Channel

- Using this arbitration method, it is possible to assign and drive one valid logical channel per each of the 2^{PNR} Pipelines. It takes $2^{PNR} \times 2$ cycles to complete one round of Pipeline numbers, but as long as the fragment size is sufficiently long, the Pipelines can be driven continuously without interruption.

- Through priority control, it is possible to prioritize the same CID that was previously executed on the same Pipeline. This allows for intentional execution of consecutive operations on the same Pipeline.

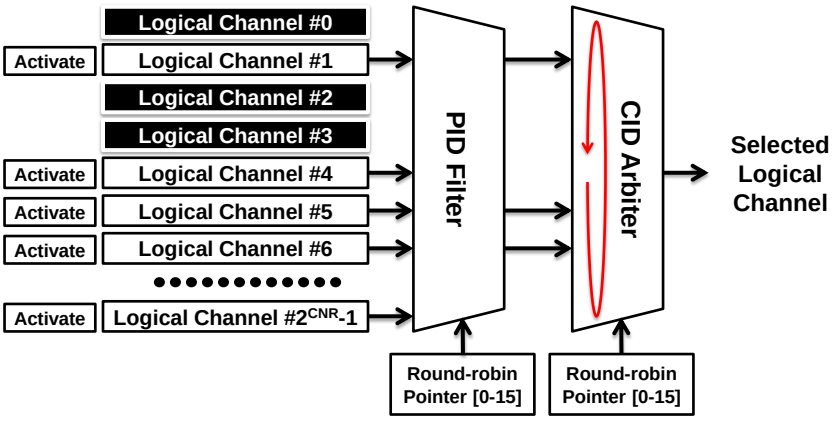


Figure 4 Seamless Pipeline Start

3.4. Pipeline Control

- The PSS handles the Index, while the Pipeline operates on the actual entity pointed to by the Index. For example, the entity could be data stored in main memory or other memory, and the Index serves as a pointer to it.

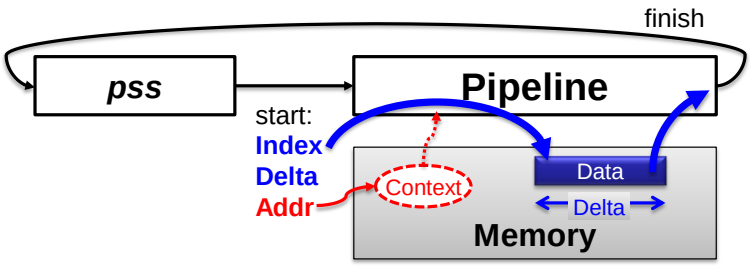


Figure 5 Example of Pipeline Processing

- Pipeline is driven by fragment volume units. Accordingly, the Pipeline is required to process the amount of fragments from the incoming Index. It is not necessary to refer to the Index and the amount of fragments. If it is not inconvenient, some Indexes can be ignored.

- Since the Pipeline performs fragment-based processing, context switching is necessary. The PSS sends the context information (Addr) together with the Index, which the Pipeline can utilize. Whether the Addr is used directly as context or as an address to fetch the context from memory is up to the implementation. Note that if the fragment size is small and context switching occurs frequently, and if there is significant Pipeline hazard (such as locking inputs

until processing completes), it can negatively impact performance.

- Completion notification to the PSS is mandatory, as it has a one-to-one relationship with Pipeline execution. Upon receiving the completion notification, the PSS updates the Index. As will be described later, link control requires that the referenced Index has already been processed. Therefore, the completion notification must be sent at the timing when the final data has been written to memory.

- Since PSS arbitration precedes Pipeline execution, drive commands are internally buffered for several stages. Buffering is necessary for performance, it also means the Pipeline may receive consecutive activations up to the maximum buffering capacity. If you need to suppress new activations during Pipeline execution, you can negate the piOk signal, which directly controls arbitration. During this period, no arbitration will be performed for that Pipeline, and drive commands will not be buffered.

3.5. Connection to Pipeline

- The connection between the PSS and each individual Pipeline can be physically connected or disconnected. For example, when resetting a Pipeline during operation, the Pipeline can be temporarily disconnected, and the states of the logical channels associated with that Pipeline in the PSS can be cleared.

- Changing the connection state during operation affects the logical channels associated with the corresponding physical channel, as well as the logical channels linked to them.

- The following connection modes are available: "Ground" results in a forced termination, "Short" causes a termination (with already issued commands temporarily suspended), and "Open" results in a temporary suspension.

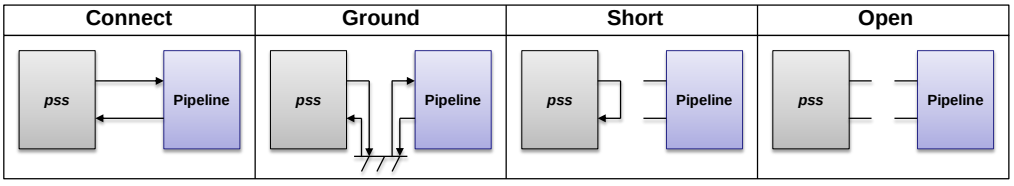


Figure 6 Link between *pss* and Pipeline

3.6. Link Control (Source/Destination)

- After arbitration, the Index of the selected logical channel is checked against the Indexes of other logical channels. Only those channels that have been designated for linking in their configuration are subject to this check.

- The check is similar to FIFO pointer management. When the data processed by a Pipeline is divided into input (Source) and output (Destination), the relationship appears as shown in Figure 8. In this case, Logical Channel A drives Pipeline X and generates data used by Logical Channel B. Logical Channel B, in turn, drives Pipeline Y and uses the data produced by Logical Channel A.

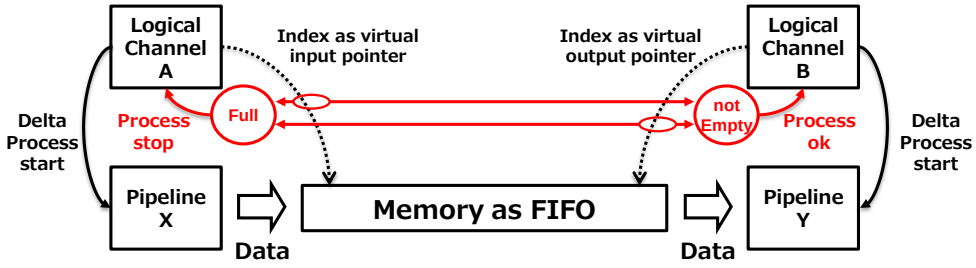


Figure 7 Link Control using Index as FIFO

- Two Indexes are managed to serve as pointers. One is updated immediately after being sent to the Pipeline (Tmp), and the other is updated after the Pipeline sends a completion notification (Cur). Tmp and Cur hold the same value but differ in update timing. By referencing Tmp for its own Index and Cur for the other party's Index, safe overtake control is ensured.

- Each logical channel can be configured with up to two links. These are generally divided into Source-side and Destination-side links (it is also possible to have two Source or two Destination links). Source-side links are used when the Pipeline should be driven only if the linked counterpart has data ready. Destination-side links are used when the Pipeline should not be driven if the linked counterpart is still processing data.

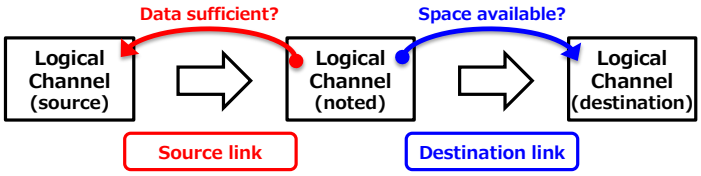


Figure 8 Source and Destination Link

- You can select any one dimension of the Index to reference. This selection allows the linking unit to be extended from one to four dimensions. For example, in the case of images, Pipeline execution can be controlled at the level of a two-dimensional framebuffer. In general, to manage FIFO-like pointer control, the transfer sizes of the referenced dimensions must match between the linked logical channels. There are no strict constraints on fragment size, but for efficiency, they are usually set the same.

- Settings for dimensions other than the referenced one are flexible. Typically, they are set the same, but different values can be used for operations such as rate conversion. For example, when converting audio from 44.1kHz to 48kHz (with the conversion handled by the Pipeline), you can set Width_X to 441 for the source and 480 for the destination, with the same Width_Y. This maintains consistency in the 480/441 data rate control.

- An offset can be applied in Index comparison. The offset introduces a delay in the link control for the referenced dimension. For **source links**, Pipeline execution waits until at least (fragment size + offset) worth of data is available. For **destination links**, Pipeline execution waits until at least (fragment size + offset) worth of space is available.

- The permission to drive the Pipeline for each source/destination link (srcOk / dstOk) is determined using the specified dimension and the following formulas: Here, srcCur / dstCur is the Index of the linked partner, carry is 1 if there's a dimensional overflow (wrap-around), srcTmp / dstTmp is the channel's own Index, delta is the fragment size, offset is the offset mentioned above, and width is either the transfer size or buffer size.

$$\text{srcOk} = (\text{srcCur} + \text{carry} \cdot \text{width}) > (\text{srcTmp} + \text{delta} + \text{offset})$$
$$\text{dstOk} = (\text{dstCur} + \overline{\text{carry}} \cdot \text{width}) > (\text{dstTmp} + \text{delta} + \text{offset})$$

- You may freely choose whether width represents transfer size or buffer size. The buffer size must be specified as a power of 2 (2^n), where n is between 1 and 7. If n = 0, the transfer size is used. If the transfer size is 0, the buffer size is treated as 1.

- If the transfer size of the referenced link is 0, $Index_C$ is used as the reference. In this case, the buffer size (2^n) can only be selected as n = 0 or n = 1, and the offset is not applied. n = 0 corresponds to a FIFO of depth 1, and n = 1 corresponds to a FIFO of depth 2.

- If there is a carry (cycle difference), the offset is ignored. This is because once the referenced Index crosses a boundary (e.g., at the edge of an image), the condition including the offset may no longer be valid. However, for control methods like data rate regulation that still require the offset even across Index boundaries, a mode that always considers the offset is also available.

- For each Source and Destination link, the Index of the referring (linked) channel can be added as a condition. In addition to the completion-based condition, an issuance-based condition can also be applied, enabling processing to wait for the issuance of other channels. This is effective when you want to prioritize another channel. However, if the channels are not assigned to the same Pipeline, in-order execution is not guaranteed, and the other channel may not necessarily be executed first.

$$\text{srcOk} \,\&= (\text{srcTmp} + \text{carry} \cdot \text{width} + 1) > (\text{srcTmp} + \text{delta} + \text{offset})$$
$$\text{dstOk} \,\&= (\text{dstTmp} + \overline{\text{carry}} \cdot \text{width} + 1) > (\text{dstTmp} + \text{delta} + \text{offset})$$

- It is possible to control link behavior unconditionally—either blocking or freeing it—based on control flags from the referenced (linked) channel rather than from the channel itself. When executing multiple commands consecutively on the same channel, you can choose which command should be associated with link control involving other channels. In Figure 10, for example, Link Command 3 of Logical Channel A is blocked by Link Commands ① and ② of Logical Channel B, preventing it from executing. Meanwhile, Link Commands ① and ③ of Logical Channel B are freed from the influence of Command 2 of Logical Channel A. Note that a command with no link control settings behaves as if it is free-controlling itself.

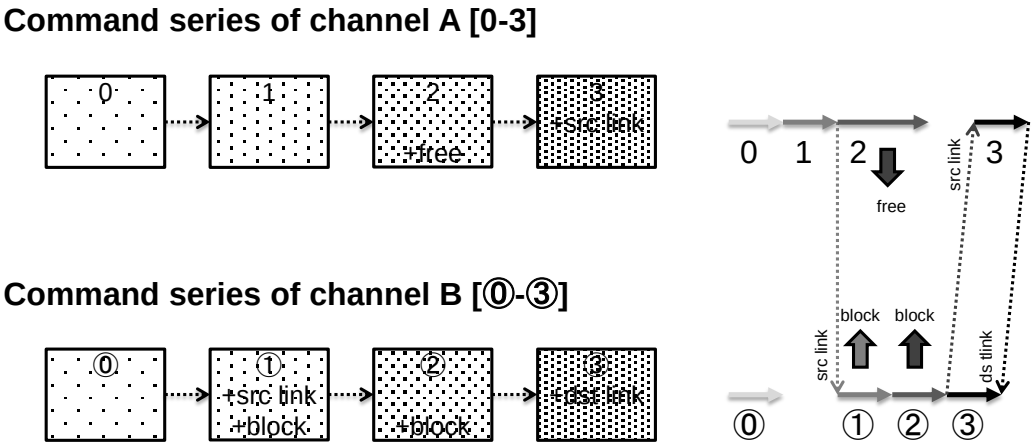


Figure 9 Link Block and Free Control

3.7. Link Control (Group)

- In addition to one-to-one link control, many-to-many link control may be required in some cases. For example, when multiple Pipelines perform overlapping writes to a single memory region. Many-to-one can be handled by having multiple logical channels establish Source/Destination links to a single logical channel. On the other hand, one-to-many cannot be directly controlled, as only two links can be configured per logical channel.

- To address this, a cascade link configuration, as shown in Figure 11, is used. When channels are connected in a cascade, no overtaking occurs between the preceding and following logical channels, and the Index difference remains within the transfer size or buffer size. Therefore, by preparing buffers equivalent to the number of chained links, the configuration appears as one-to-many from the entrance of the chain, and many-to-one from the exit.

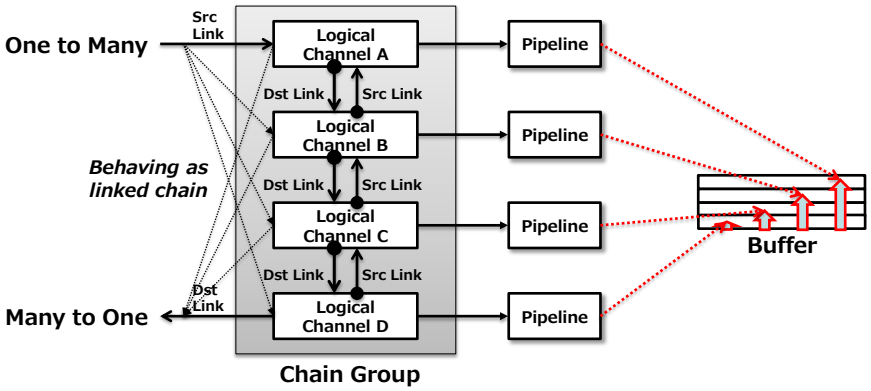


Figure 10 Cascade Link

➢ Multiple required buffers can be consolidated into a single buffer. To ensure that the Index of the first logical channel does not advance beyond the Index of the last logical channel plus the transfer size or buffer size, all Destination links should be configured to reference the last logical channel, as shown in Figure 12.

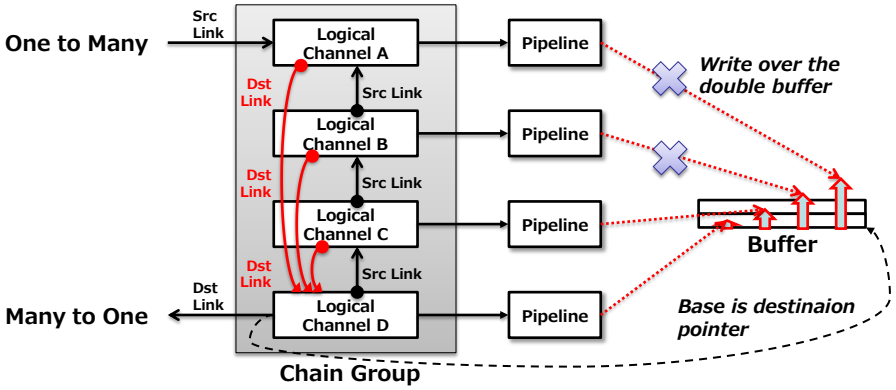


Figure 11 Cascade Link for Minimal Buffer

3.8. Index Mask

➢ Pipeline execution can be masked based on the value of the Index in a specific dimension. It is also possible to subtract a lower limit value to output a relative Index.

➢ Index masking is effective in link control (Group) when accessing different portions of data. Figure 13 shows an example where Logical Channels B, C, and D, while applying overtake control with the same transfer size, partially drive the Pipeline. Note that the PSS continues to update the Index even in the masked regions.

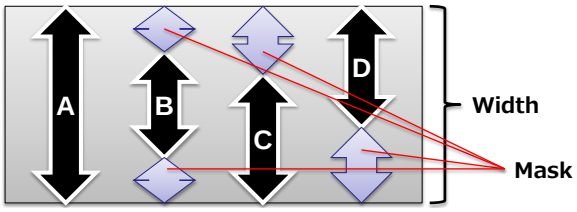


Figure 12 Partial Pipeline Activation

3.9. Restore Function

- In some cases, such as with raw devices, link control may not be possible. For example, video output must continue refreshing even if there is no data available. To handle such situations, the PSS includes a Restore function that can drive the Pipeline even when the Index conditions on both sides of the link are not met.

- The Restore function allows Pipeline execution to proceed even when link control conditions are violated, and it outputs a Restore flag to indicate this state. The Pipeline can use this flag to respond appropriately, such as by referencing the previous frame's Index.

- When the system enters a Restore state, it locks Index updates and command list switching for the specified dimension and above. This lock remains in effect until the current frame is completed.
  For example, in Figure 14, transfer from frame B to frame X proceeds normally, but if the switch to frame X occurs while frame A is being processed, the system enters a Restore state (the Pipeline should re-reference the data from frame B). The Restore state is cleared after the second transfer to frame X completes and frame A becomes valid again.

- Note that the Restore function does not operate if link control is not configured. Also, it is limited to cases where only one link side (either Source or Destination) is configured.

- A forced Restore can be triggered from an external terminal using intOp, which is paired with intSet. This allows for intentional transfers initiated by the Pipeline without updating the Index.

- For these Restore functions to operate, prior permission must be granted through the command list. A flag must be set at the time of activation.

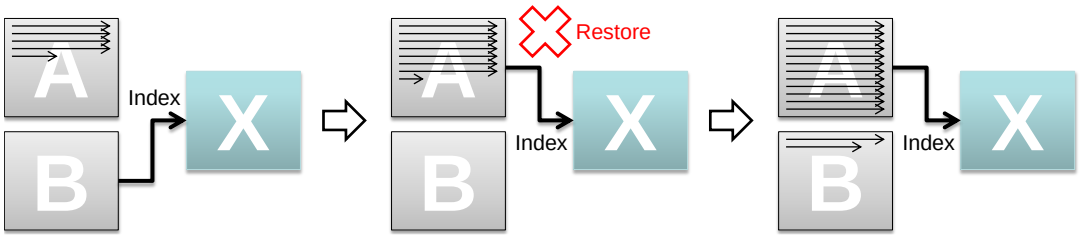


Figure 13 Example of Restore State Transition

3.10. Command List

- The parameters of a logical channel can be automatically swapped each time processing is completed by referencing a replacement address. Therefore, by creating a command list and linking it via the replacement address, arbitrary sequential processing becomes possible.

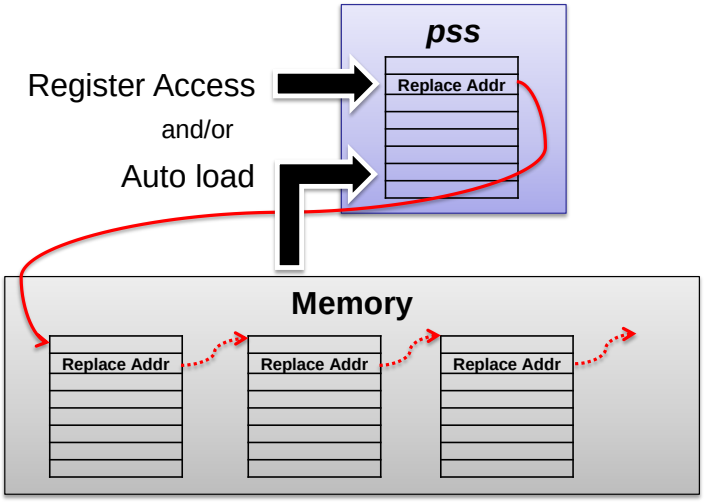


Figure 14 Command List Serial Processing

- The swapped-in command list also controls the activation register. If the new command list indicates a stop, the process ends after simply replacing the parameters.

3.11. Index Clearing and Continuation

- The 0 clearing at the end condition of Index and the 1 addition at the time of update are not executed by setting the command (Cntl.Incr), and the current Index value can be succeeded to the next command.

- For example, if different processing is to be performed on any given section of the Index, in each command, set the Index not to clear 0 and the Index not to carry digits up. The upper part of Fig. 16 shows the normal command transition, where the set ⊿ minutes are processed sequentially. The lower arrow line in Figure. 16 is the transition that takes over the Index value and processes the set ⊿ minus the Index value.

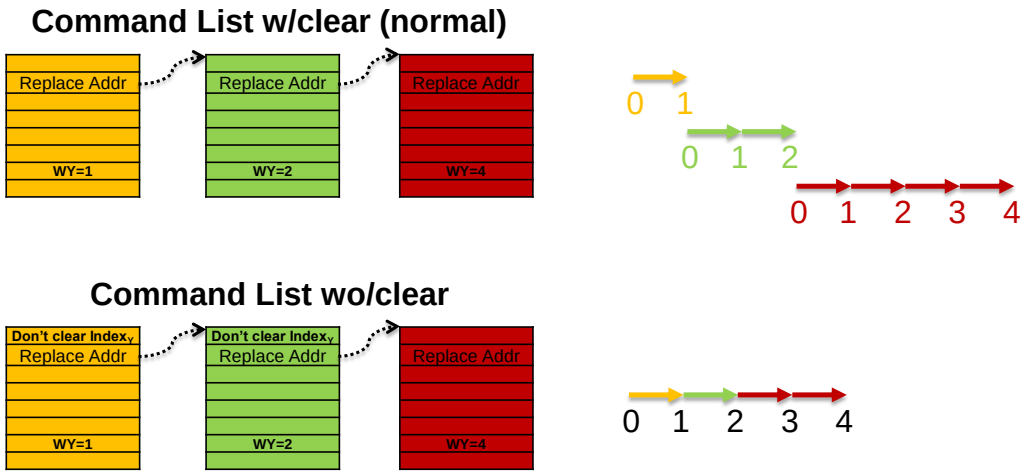


Figure 15 Index Clear and Non-Clear

➢ In addition, it is possible to implement programmatic processing that stacks multiple commands on the same Index (for example, $Index_Y$ during screen scanning). Figure 17 shows an example where a process with $\Delta Y = 64$ is divided into three areas, with three commands assigned to each area. Each group of three commands is configured to loop among themselves until the corresponding Y Index range is completed: [CL0 → CL1 → CL2] × 16 → [CL3 → CL4 → CL5] × 32 → [CL6 → CL7 → CL8] × 16.

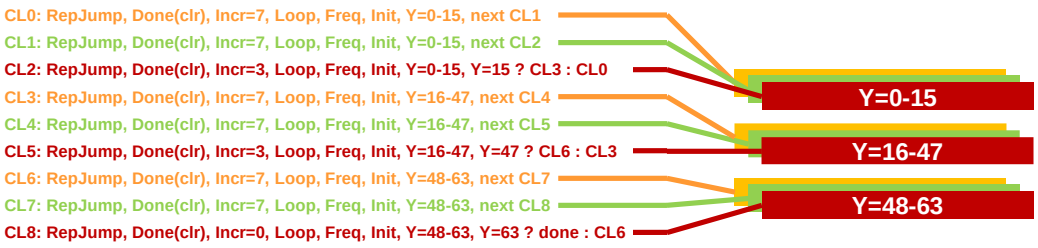


Figure 16 Command List Program

Cntl register	Description
RepJump	At the end of Y, execute the command at the CL address + 0x20 When Y is not completed, execute the command of the address written in the Rep register. (Loop CLn→CLn+1→CLn+2 until each setting ΔY)
StatDone	Clear '1' so that the exit flag does not continue to be set.

Incr	CL0,1,3,4,6,7 are set to '7' to repeat the same Y (no update) CL2,5 is set to '3' so that only Y is incremented (only Y is updated). CL8 is set to '0' to terminate normally
Loop	CL0-7 are set to be continuous, CL8 is not set for termination
Freq/Init	Update command each time ΔX ends and processing begins

3.12. Activation and Termination

- There are four ways to activate a logical channel:
  - Set parameters and operate the logical channel's activation register
  - Set parameters, then operate the startup vector and group activation register
  - Set parameters and activate via an external terminal
  - Set only the replacement address for parameters and operate the logical channel's activation register

- If the command list replacement flag (Init) is set at activation, new parameters will be fetched from memory according to the replacement address. The activation register will also be updated with new values.

- If the continuous execution flag (Loop) is set, the next execution will occur even after reaching the termination condition. If the Init flag is also set, the command list will be replaced with a new one. If not, the same command will be executed again under the same conditions (only the highest Index of dimension 4 will be incremented).

- The behavior of command replacement differs depending on whether it occurs at the time of activation or after activation. If Init = 1 at activation, all commands are replaced, and the initial parameters specified during activation are ignored. Since the activation register is also replaced, if the activation flag (Act) in the new activation register is 0, the operation will terminate immediately. Refer to Figure 18 for the flow diagram.

- The value of the Index is preserved even when the logical channel is stopped, allowing for interruption and resumption. To start from zero again, either perform a dummy stop-and-clear activation before starting, or set the clear flag during command list replacement.

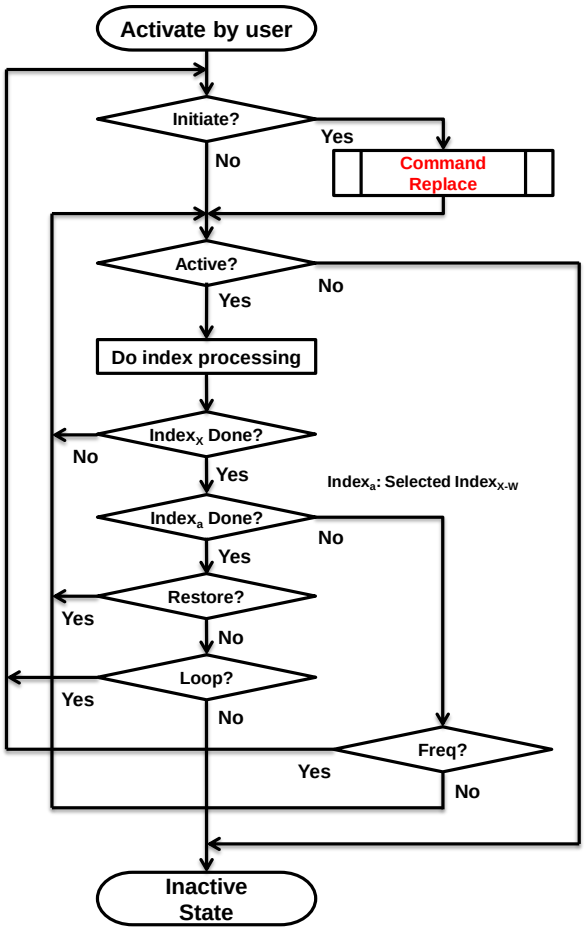


Figure 17 Command Replace Sequence

The state of each logical channel is managed individually. A simplified state transition diagram is shown in Figure 19. IDLE indicates a stopped state, LOAD indicates command list loading, DO represents execution, and DONE indicates completion. (Note: This diagram is simplified for explanatory purposes.)

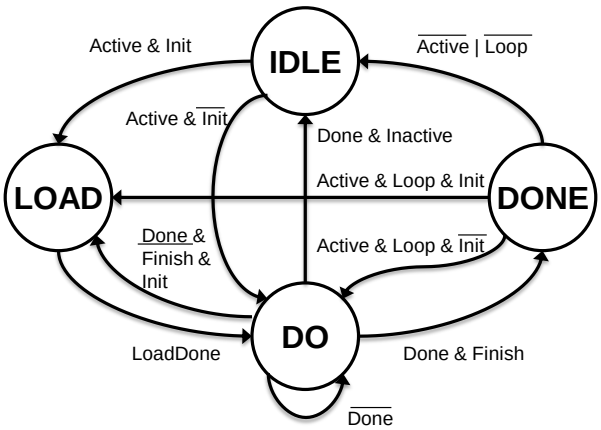


Figure 19 State Transition

- The activation register must be operated individually for each logical channel, whereas the activation vector is used to activate multiple logical channels at once. By setting flags for the desired channels in the activation vector and then operating the group activation register just once, simultaneous activation can be achieved.

- The external **intSet** signal for activation corresponds to setting Bit 0 of the activation register. Since it is evaluated in one cycle, the pulse must be one clock width wide or it may be continuously activated. Multiple assertions at the same time are allowed. This method takes priority over register access.

- The external **intClr** signal for stopping behaves oppositely to **intSet**, stopping the specified logical channel. Since it is evaluated in one cycle, the pulse must be one clock width wide or it may be continuously activated. Multiple assertions at the same time are allowed. It also takes priority over register access, but **intSet** has higher priority than **intClr**.

- In the stopped state, the Index is preserved, so if the channel is restarted without clearing it via register access or other means, it will resume from where it left off.

3.13. Termination Conditions and Interrupts

- logical channel is considered to have terminated when the configured Index has completely scanned all transfer dimensions. This means that the $Index_C$ of dimension 4 is inverted. All other Indexes reset to 0, even if intermediate dimensions have a transfer size of 0.

- The state of a logical channel can be checked using the StatMain field of the Cntl register. A value of all zeros indicates the **IDLE** state. Register-based activation is prohibited when the channel is not in **IDLE** state.

- Even if a stop to the activation register is performed and the state of the logical channel is **IDLE**, a startup (piVld assertion) may be applied to the pipeline if a command is queued in the command FIFO between the logical and physical channels. You can check the status of the command FIFO using the Stat field of the PI Register.

- The command FIFO not Busy state allows detection of the complete **IDLE** of the relevant logical channel in pss.

- If the Pipeline becomes locked for any reason and the command FIFO remains busy without naturally clearing, the logical channels associated with that physical channel will also be locked. Setting those logical channels to **IDLE** will not resolve that issue. In this case, you must reset the corresponding Pipeline and also explicitly clear the command FIFO using the Reset in the PC Register.

- Interrupts can be triggered upon termination of a logical channel. To enable this, set the interrupt flag (Int) in the activation register.

- When a logical channel terminates, the termination condition flag (StatDone) is set to 1. This condition can also trigger an interrupt, so take care with the interrupt timing. In continuous processing scenarios, StatDone becomes 1 at the end of each execution. If you intend to trigger an interrupt only at the end or at specific points in a continuous process, be sure to clear the termination condition beforehand.

3.14. Interrupt-Driven Execution

- A logical channel n is activated when a handshake is established between the external signals intVld[n] and intStall[n] (i.e., intVld is asserted and intStall is not). The intStall signal indicates that the logical channel is currently active. To use intVld for activation, the corresponding logical channel must be preconfigured—for example, by setting the Rep Register to automatically update the command list upon interrupt-based activation.

- It is also possible to activate a specific logical channel from an interrupt triggered by another logical channel. In this case, one or two Link Registers are used to designate the interrupt target. Link Registers configured for interrupt purposes cannot be used for link control. It is also possible to set the interrupt to be allowed only to specific destinations.

- If the interrupt drive is a pulse signal and the intStall signal at the interrupt destination is standing, or if the interrupt is rejected in the IntInDis register, any interrupt input is ignored.

- Even if a logic channel interrupt cannot be accepted, if BufEn in the Int register is set to '1', the interrupt state is held for one interrupt, and interrupt driving is performed when the intStall signal is released.
- Note that if the logical channel interrupt destination returns to the interrupt source, an infinite loop is started.

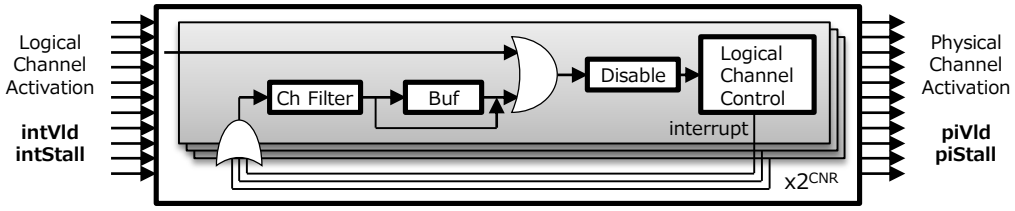


Figure 18 Outer and Inner Interrupt Activation

3.15. Priority Control

- Each logical channel can be assigned a priority level. It has a register which reduces the priority value (Prior) by 1 every 2^{PNR} cycles, and when a request is accepted, 16 is added to the register. The resulting value is added as a weight in the arbitration process. This mechanism effectively records the difference between the number of activations and the target activation rate per cycle (Prior / 2^{PNR+4}). By maintaining this difference in a steady state, activation occurs in a way that is weighted according to the priority, leading to an average activation rate of approximately (Prior / 2^{PNR+4}) activations per cycle.

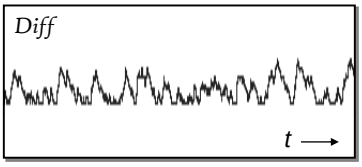


Figure 19 Difference between activation and sum of priority per cycle

- Finally, priority control is performed on the physical channel. That is, the priority Prior is set for the physical channel that is set in the logical channel.

- If it is ready to be activated, it is activated regardless of the priority. On the other hand, if the total amount of priority settings for logical channels exceeds the physical capacity, the logical channels that exceed the physical capacity are equally allocated to each other.

- If Prior = 0, the channel will always have the lowest priority (weight = 0). If Prior = 0xF, the channel will always have the highest priority (weight = 1).

---

4. Register Description

4.1. Overview

- All registers are accessed through the control bus interface.

- Some settings need to be carefully timed as they may affect the operation and performance of the Pipeline.

- The following access types are used in register definitions:
  - **R** — Read only
  - **R/W** — Read and write
  - **R/WC** — Read / Write-to-clear

- Do not access registers marked as Reserved. When writing to Reserved fields, be sure to set the value to '0'.

- In address and data descriptions, the symbol 'x' indicates a Don't Care value.

- Asynchronous reset initializes all registers, while synchronous reset initializes only certain fields in the Command register. In the detailed register descriptions, fields that are initialized by synchronous reset are marked with a specific symbol (described below).

  † Registers to be reset synchronously

4.2. Definition（Command）

Address	Register Name	Description
0000_0000 + 64n	Cntl[n]	Activation Register
0000_0004 + 64n	Rep[n]	Replacement Register
0000_0008 + 64n	Pipe[n]	Pipeline Register
0000_000c + 64n	Mask[n]	Mask Register
0000_0010 + 64n	Link0[n]	link Register0
0000_0014 + 64n	Link1[n]	Link Register1
0000_0018 + 64n	Width0[n]	Transfer Size Register0
0000_001c + 64n	Width1[n]	Transfer Size Register1
0000_0020 + 64n	Tmp0[n]	Pre-Update Index Register0
0000_0024 + 64n	Tmp1[n]	Pre-Update Index Register1
0000_0028 + 64n	Cur0[n]	Post-Update Index Register0
0000_002c + 64n	Cur1[n]	Post-Update Index Register1

$n=0 \sim 2^{CNR}-1$

The address space from 0x0000_000C onward is assigned per logical channel.

4.3. Definition (General)

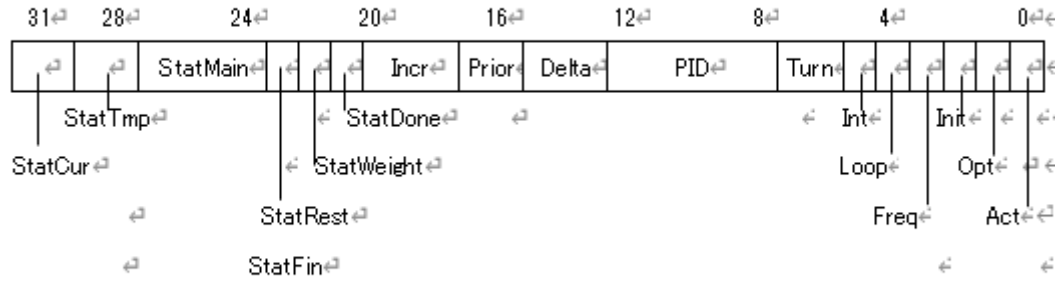
Address	Register Name	Description
0000_8000	Reset	Reset Control
0000_8004	Clock	Clock Control
0000_8008	Info	Status (Overall)
0000_800c	Int	Interrupt Enable (Overall)
0000_8014	Timer	Priority Control Timer
0000_8018	Grp	Activation (Overall)
0000_8200 + 4n	ClearVec[n]	Clear Vector(n=0～2^{CNR}/32-1)
0000_8300 + 4n	PauseVec[n]	Stop Vector(n=0～2^{CNR}/32-1)
0000_8400 + 4n	ActVec[n]	Activation Vector(n=0～2^{CNR}/32-1)
0000_8500 + 4n	StatVec[n]	Status Vector(n=0～2^{CNR}/32-1)
0000_8600 + 4n	DoneVec[n]	Stop Vector(n=0～2^{CNR}/32-1)
0000_8700 + 4n	IntVec[n]	Interrupt Vector(n=0～2^{CNR}/32-1)
0000_8800 + 4n	IntDisVec[n]	Interrupt Disable Vector(n=0～2^{CNR}/32-1)
0000_8900 + 4n	IntInOnVec[n]	Interrupt Logical Channel Enable Vector (n=0 ～ 2^{CNR}/32-1)
0000_9000 + 4n	IntInChVec[n]	Interrupt Logical Channel Vector(n=0～2^{CNR}/4-1)
0000_c000 + 256n	PC[n]	Pipeline control(n=0～2^{PNR}-1)
0000_c004 + 256n	PD[n]	Pipeline Configuration(n=0～2^{PNR}-1)
0000_c008 + 256n	PI[n]	Pipeline infomation(n=0～2^{PNR}-1)

4.5. Details (Command)

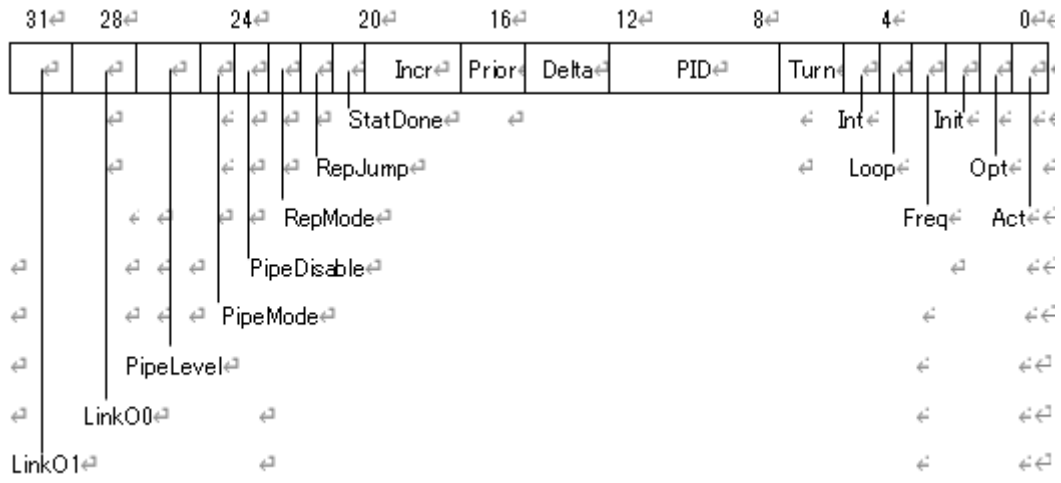
4.5.1.1. Cnt[n] Register

[Address: 0x0001_0000 + 64n]

When 'Read'



When 'Write'



Name	Type	Default	Description
LinkO0,1[1:0]	W	x	Controls link behavior toward other logical channels. It does not affect the channel's own link control. In Mag mode, the Mag register is referenced, and the link pointer is multiplied to enable proportional link control. In Mask mode, the Mask register is referenced, and the link pointer is masked. Mask mode is not enabled if either LinkO0 or LinkO1 is set to 0 in Mag mode.

LinkO	Description
0	Mag mode

			1 Mask mode 2 Always block 3 Always free
PipeLevel[1:0]	W	x	Set the dimension of the Index mask for the Pipeline. Level: 0 Mask for $Index_X$; 1 Mask for $Index_{XY}$; 2 Mask for $Index_Z$; 3 Mask for $Index_W$
PipeMode	W	x	Set to '1' to enable the Restore function for Pipelines.
PipeDisable	W	x	Set to '0' to allow Pipeline execution. When set to '1', only the Index is updated and the Pipeline is bypassed.
RepMode	W	x	Set to '1' to clear the Index during replacement. Note that if Loop in the Cntl Register is '0' and Init is '1', and replacement is performed at the $Index_X$ level, there is a risk of entering an infinite loop without meeting the termination condition.
RepJump	W	x	Set to '1' to control the starting address of the command during replacement. If the termination condition is met, the address will be the current address plus 0x20. If the termination condition is not met, or if the setting is '0', the address will be taken from the Rep Register.
StatCur	R	0	Indicates the termination value of the updated Index (value of dimension 4). Writing '1' will clear it. It is also cleared during automatic loading if RepMode is set to '1'.
StatTmp	R	0	Indicates the pre-update termination value of the Index (value of dimension 4). Writing '1' will clear it. It is also cleared during automatic loading if RepMode is set to

'1'.

StatMain R 0 Indicates the status of individual logical channels.

statMain	Name	Description
0	IDLE	Stop
1	PROG	In Progress
2	LOCK	Waiting for Scheduling
3	WAIT	Waiting for Processing
4	FLUSH	Terminating
5	REST	Restoring
6	-	Reserved
7	-	Reserved
8	LDRQn	Requesting New Command
9	LDRQo	Requesting continued Command
10	-	Reserved
11	LDRDo	Waiting for Command
12	LDDO	Loading for Command
13	-	Reserved
14	-	Reserved
15	-	Reserved

StatRest R 0 Indicates that the system is in a Restore state. It is automatically cleared upon activation.

StatDone† R/WC 0 Indicates a termination condition. Writing '1' will clear it. It can also serve as an interrupt trigger.

Incr[2:0] R/W x Suppresses Index clearing and carry-up. No Clear means the corresponding Index will not be cleared even if it reaches the termination condition. No Carry-Up

means the corresponding Index will not be incremented even when a lower Index reaches its termination condition.

Incr	No clear	No Carry-Up
0	-	-
1	$Index_W$	$Index_C$
2	$Index_Z$ $Index_W$	$Index_W$ $Index_C$
3	$Index_Y$ $Index_Z$ $Index_W$	$Index_Z$ $Index_W$ $Index_C$
4	-	$Index_C$
5	$Index_W$	$Index_W$ $Index_C$
6	$Index_Z$ $Index_W$	$Index_Z$ $Index_W$ $Index_C$
7	$Index_Y$ $Index_Z$ $Index_W$	$Index_Y$ $Index_Z$ $Index_W$ $Index_C$

Prior[1:0] R/W x Sets the priority level. '0' is the lowest, and '3' is the highest. Priority level '3' gives preference to the same CID (current logical channel) in subsequent processing with the same PID. In this case, the corresponding physical channel's PD[n].Lock must be set to '1'; otherwise, the same CID will not be prioritized.

Delta[2:0] R/W x Sets the fragment size, which becomes 4^{Delta}. However, if Delta = 7, the value is replaced with $Width_X$ + 1 from the Width register.

PID[4:0] R/W x Sets the Pipeline number. Only the bits PID[PNR-1:0] are valid.

Turn[1:0] R/W x Sets the temporary termination condition for the Pipeline. Note that the level is in reverse order.

Turn	Description
0	Stop at the end of $Index_W$.
1	Stop at the end of $Index_Z$.
2	Stop at the end of $Index_Y$.
3	Stop at the end of $Index_X$

Int† R/W 0 Set to '1' to trigger an interrupt on the termination condition. However, if the IntEn bit in the Int Register is not set to '1', the cntlIrq signal will not be asserted.

Loop† R/W 0 Set to '1' to continue execution even after the termination condition is met.

Freq† R/W 0 Specifies the command replacement behavior when Init is set to '1'.

Init† R/W 0 If set to '1' in the IDLE state, the command (from the Cntl to the Width register) will be replaced immediately after activation, and the system will activate with new parameters. The behavior after activation will vary depending on the Freq setting, which controls how the replacement operation behaves.

Init	Freq	Description
0	0	No Replacement
0	1	
1	0	Replacement at the end of $Index_a$ processing (where a refers to the specified Turn).
1	1	Replacement at the end of $Index_X$ processing.

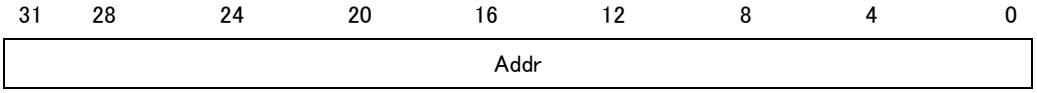
Opt† R/W 0 Set activation options.

Act†	R/W	0	Activation settings. When reading, it will be set to '1' if the system is in an execution state.

Opt	Act	Symbol	Description
0	0	Halt	stop
0	1	Start	activation
1	0	Clear	stop＆Initialization
1	1	Step	No Pipeline Processing

4.5.1.2. Rep [n] Register

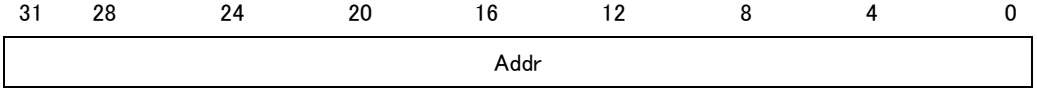
[Address: 0x0001_0004 + 64n]



Name	Type	Default	Description
Addr[31:0]	R/W	x	Set the starting address of the command (from Cntl to the Width Register). ※ For beppu, [0] = 1 is required, but it is prohibited for chichibu.

4.5.1.3. Pipe[n] Register

[Address: 0x0001_0008 + 64n]

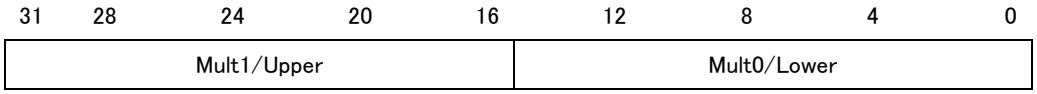


Name	Type	Default	Description
Addr[31:0]	R/W	x	Set the value to be sent to the Pipeline (via the piAddr signal).

※ For beppu, [0] = 1 is required, but it is prohibited for chichibu.

4.5.1.4. Mag/ Mask[n] Register

[Address: 0x0001_000c + 64n]



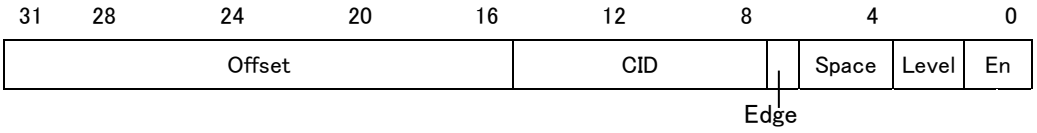
Name	Type	Default	Description
Mult1[15:8]	R/W	x	Set the multiplication factor minus 1 to be applied to the pointer of the destination in Link1.
Mult1[7:0]	R/W	x	Set the multiplication factor minus 1 to be applied to the pointer of the source in Link1.
Mult0[15:8]	R/W	x	Set the multiplication factor minus 1 to be applied to the pointer of the destination in Link0.
Mult0[7:0]	R/W	x	Set the multiplication factor minus 1 to be applied to the pointer of the source in Link0.
Lower[15:0]	R/W	x	Set the lower limit of the mask. Index values (specified by the Level in the Pipe Register) that are below this value, excluding the value itself, will be masked. This is equivalent to setting the En bit of the Pipe Register to '0'. If the Upper value is '0', this means something different, as described below.

Lower	Description
Bit0	Output $Width_X$ - $Index_X$ when set to '1'.
Bit1	Output $Width_Y$ - $Index_Y$ when set

	to '1'.
Bit2	Output $Width_Z$ - $Index_Z$ when set to '1'.
Bit3	Output $Width_W$ - $Index_W$ when set to '1'.
Bit4	Perform the following 1-dimensional access when set to '1': Link control requires careful attention to the Level setting (select a level where Width = 0 to avoid incorrect pointer comparisons). $Width_X + Width_Y * 65536 + Width_Z * 65536^2 + Width_W * 65536^3$.
Bit11-8	Set the invalid bits on the MSB side of the target Index during Source link control. (For example, if set to 3, the lower 13 bits will be valid.)
Bit15-12	Set the invalid bits on the MSB side of the target Index during Destination link control. (For example, if set to 3, the lower 13 bits will be valid.)

4.5.1.5. Link0,1[n] Register

[Address: 0x0001_0010 + 64n, 0x0001_0014 + 64n]



Name	Type	Default	Description

Offset[15:0] R/W x Set the offset or scale during link control.

Offset[15:14]	Description
0	Use Offset[13:0] as the offset.
1	Add Tmp pointer + 1 to the condition, along with the destination Cur.
2	Scale the source Tmp pointer by $2^{Offset[11:8]}$.
3	Scale the destination Cur pointer by $2^{Offset[11:8]}$.

CID[7:0] R/W x When En = '1' or En = '3', set the logical channel number to be referenced. When En = '2', set the logical channel number to be interrupted.
Only CID[CNR-1:0] is valid.

Edge R/W x Set to '1' if the offset is not ignored even when there is a periodic difference between the referenced Index and the channel's own Index.

Space[2:0] R/W x Set the buffer size to 2^{Space}. However, if Space = '0', the Width + 1 specified by the Level in the Width register will be used (as explained in the Level description, for $Index_C$ reference, it will be 1).
If the Width of the referenced Level is 0, only settings of 0 or 1 are guaranteed to function properly.

Level[1:0] R/W x Set the referenced Index. However, if the Width corresponding to this Level is 0, the reference will be made to $Index_C$.

Level	Description
0	Reference to $Index_X$
1	Reference to $Index_y$
2	Reference to $Index_z$

3	Reference to $Index_w$

Name	Type	Default	Description
En[1:0]	R/W	x	Set the link control.

En	Description
0	NOP
1	Reference to Source
2	Int occurrence
3	Reference to Destination

4.5.1.6. Witch0,1[n] Register

[Address: 0x0001_0018 + 64n]

31 28	24 20	16 12	8 4	0
Width1		Width0		

[Address: 0x0001_001c + 64n]

31 28	24 20	16 12	8 4	0
Width3		Width2		

Name	Type	Default	Description
Width0,1,2,3[15:0]	R/W	x	Set the transfer size minus 1 for dimensions 0, 1, 2, and 3.

4.5.1.7. Tmp0,1[n] Register

[Address: 0x0001_0020 + 64n]

31 28	24 20	16 12	8 4	0
Tmp1		Tmp0		

[Address: 0x0001_0024 + 64n]

31 28	24 20	16 12	8 4	0

Tmp3	Tmp2

Name	Type	Default	Description
Tmp0,1,2,3[15:0]	R	x	Indicates the Index for dimensions 0, 1, 2, and 3 (pre-update). It is not automatically cleared at activation, but will be cleared at the termination condition. This is for monitoring purposes.

4.5.1.8. Cut0,1[n] Register

[Address: 0x0001_0028 + 64n]

31 28	24 20	16 12	8 4	0
Cur1		Cur0		

[Address: 0x0001_002c + 64n]

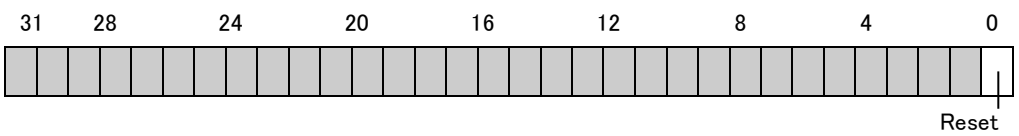
31 28	24 20	16 12	8 4	0
Cur3		Cur2		

Name	Type	Default	Description
Cur0,1,2,3[15:0]	R	x	Indicates the Index for dimensions 0, 1, 2, and 3 (pre-update). It is not automatically cleared at activation, but will be cleared at the termination condition. This is for monitoring purposes.

4.6. Details (General)

4.6.1.1. Resett Register

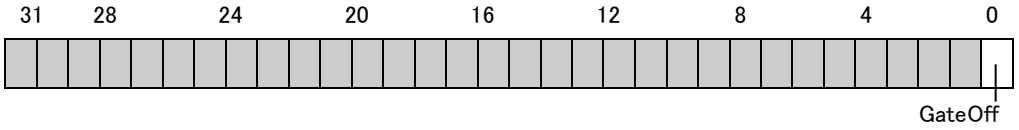
[Address: 0x0000_8000]



Name	Type	Default	Description
Reset	R/W	0	Synchronous reset. After setting to '1', the system enters an internal reset state and is automatically cleared. Unlike the **reset_n** signal, the contents of other registers are preserved.

4.6.1.2. Clock Register

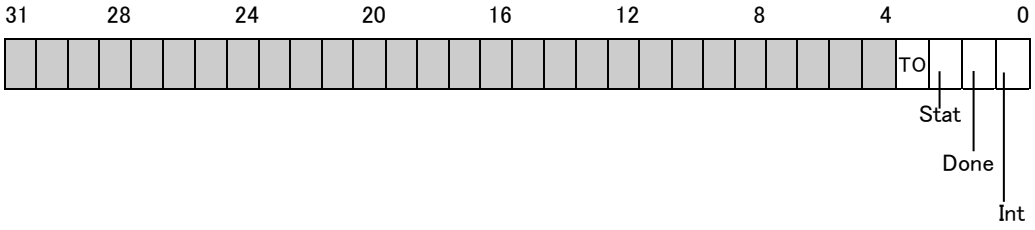
[Address: 0x0000_8004]



Name	Type	Default	Description
GateOff	R/W	0	Gated Clock Off mode. When set to '1', all bits of the gate signal are fixed to '1'

4.6.1.3. Info Register

[Address: 0x0000_8008]

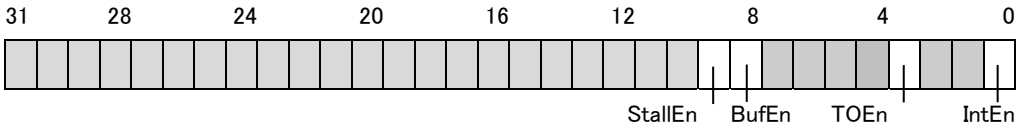


Name	Type	Default	Description
TO	R	0	Indicates that there is a timeout interrupt on any physical channel.
Stat	R	0	Indicates that there is an interrupt on any logical channel.
Done	R	0	Indicates that any logical channel has terminated.

Int	R	0	Indicates that any logical channel has terminated and interrupts are enabled.

4.6.1.4. Int Register

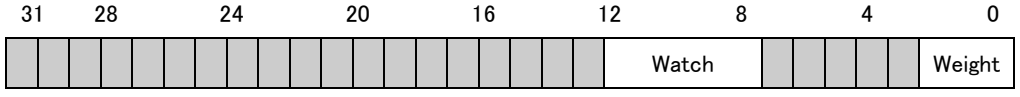
[Address: 0x0000_800c]



Name	Type	Default	Description
StallEn	R/W	0	When set to '1', the **cntlIrq** signal is asserted when the activation state is **Stall**.
BufEn	R/W	0	When set to '1', the **cntlIrq** signal is asserted during logical channel interrupts.
TOEn	R/W	0	When set to '1', Info.TO is asserted to the **cntlIrq** signal.
IntEn	R/W	0	When set to '1', Info.Int is asserted to the **cntlIrq** signal.

4.6.1.5. Timer Register

[Address: 0x0000_8014]

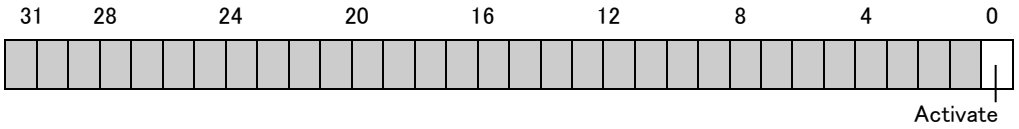


Name	Type	Default	Description
Watch[4:0]	R/W	0	Timeout measurement indicator. A timeout occurs when all bits from the bit indicated by Watch (starting from the MSB of the 32-bit execution cycle of the physical channel) to the LSB are set to '1'.
Weight[2:0]	R/W	PNR	Sets the timing for priority weight control. Every 2^{Weight} cycles, the priority is decremented by the weight value

assigned to each logical channel.

4.6.1.6. Grp Register

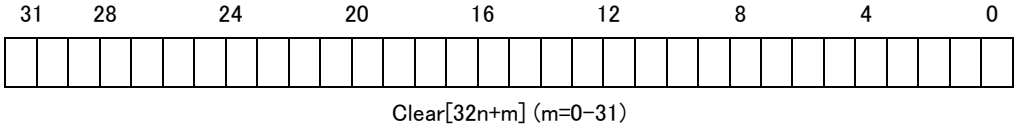
[Address: 0x0000_8018]



Name	Type	Default	Description
Activate	W	x	Activate all logical channels that are set to '1'.

4.6.1.7. ClearVec[n] Register

[Address: 0x0000_8200 + 4n (n=0-2^{CNR}/32-1)]



Name	Type	Default	Description
Clear	R/W	0	Internal information clear vector. Set '1' in the portion counted from the LSB based on the logical channel number to be cleared. This clears the information of logical channels affected by a slave causing inconsistency.

4.6.1.8. PauseVec(n)Register

[Address: 0x0000_8300 + 4n (n=0-2^{CNR}/32-1)]

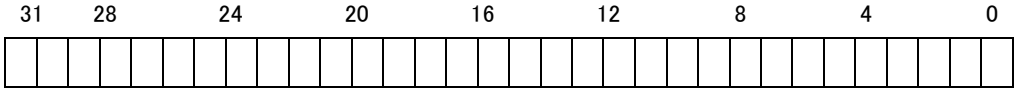
31	28	24	20	16	12	8	4	0

Pause[32n+m] (m=0-31)

Name	Type	Default	Description
Pause	R/W	0	Stop vector. Set '1' in the portion counted from the LSB based on the logical channel number to be stopped. This blocks automatic activation via the Init in the Cntl Register. Transition from the IDLE state is not blocked.

4.6.1.9. ActVec[n] Register

[Address: 0x0000_8400 + 4n (n=0-2^{CNR}/32-1)]

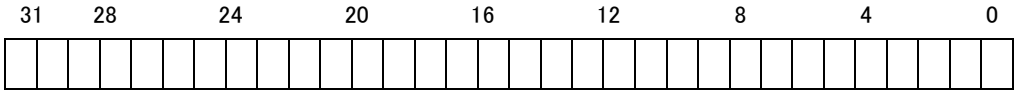


Act[32n+m] (m=0-31)

Name	Type	Default	Description
Act	R/W	0	Activation vector. Set '1' in the portion counted from the LSB based on the logical channel number to be activated. This is used with the Activate in the Grp Register.

4.6.1.10. StartVec Register

[Address: 0x0000_8500 + 4n (n=0-2^{CNR}/32-1)]



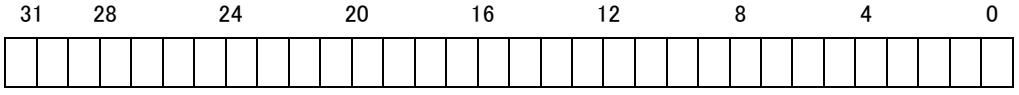
Stat[32n+m] (m=0-31)

Name	Type	Default	Description
Stat	R	0	Status vector. The state is determined by examining the portion counted from the LSB based on the logical

channel number. A value of '1' indicates that the channel is active.

4.6.1.11. DoneVec[n] Register

[Address: 0x0000_8600 + 4n (n=0-2^{CNR}/32-1)]

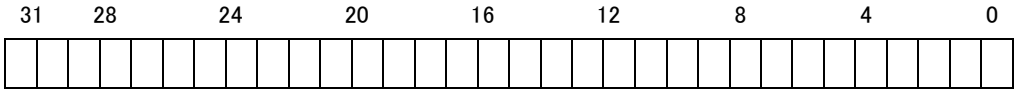


Done[32n+m] (m=0-31)

Name	Type	Default	Description
Done	R	0	Termination vector. The termination status is determined by examining the portion counted from the LSB based on the logical channel number. A value of '1' indicates that the channel has terminated.

4.6.1.12. IntVec[n] Register

[Address: 0x0000_8700 + 4n (n=0-2^{CNR}/32-1)]

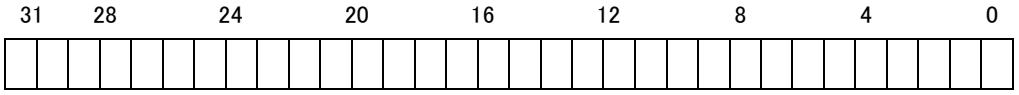


Int[32n+m] (m=0-31)

Name	Type	Default	Description
Int	R	0	Interrupt vector. The interrupt status is determined by examining the portion counted from the LSB based on the logical channel number. A value of '1' indicates that there is an interrupt.

4.6.1.13. IntDisVec[n] Register

[Address: 0x0000_8800 + 4n (n=0-2^{CNR}/32-1)]

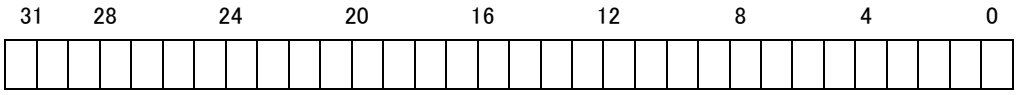


IntDis[32n+m] (m=0-31)

Name	Type	Default	Description
IntDis	R/W	0	Logical channel interrupt disable vector. Based on the logical channel number, set '1' in the portion counted from the LSB to disable the interrupt for that channel.

4.6.1.14. IntInOnVec[n] Register

[Address: 0x0000_8900 + 4n (n=0-2^{CNR}/32-1)]

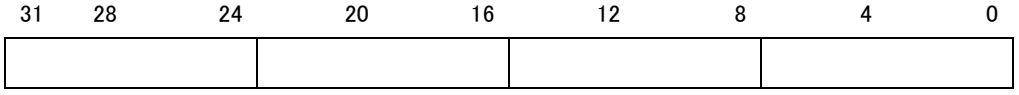


IntOnEn[32n+m] (m=0-31)

Name	Type	Default	Description
IntOnEn	R/W	0	Logical channel interrupt specification enable vector. Based on the logical channel number, set '1' in the portion counted from the LSB to allow only the specified logical channel as an interrupt source for IntInCh.

4.6.1.15. IntInChVec[n] Register

[Address: 0x0000_9000 + 4n (n=0-2^{CNR}/4-1)]

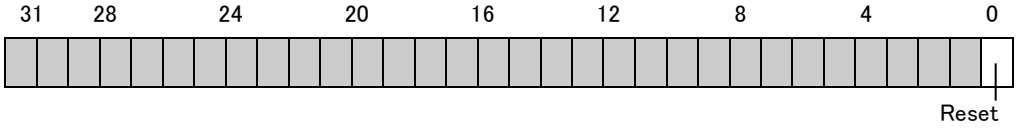


IntInCh[32n+:8m] (m=0-3)

Name	Type	Default	Description
IntInCh	R/W	0	Logical channel interrupt specified logical channel vector. Based on the logical channel number, set the portion counted from the LSB to specify the interrupt source logical channel.

4.6.1.16. PC[n] Register

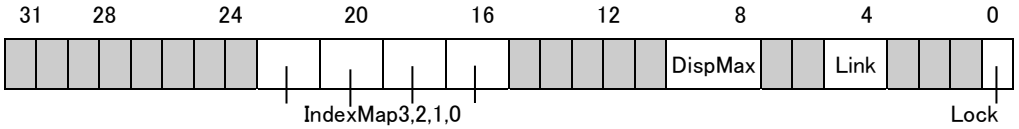
[Address: 0x0000_c000 + 256n]



Name	Type	Default	Description
Reset	WC	0	Clear the internal state related to Pipeline number n.

4.6.1.17. PD[n] Register

[Address: 0x0000_c004 + 256n]



Name	Type	Default	Description
IndexMap0[1:0]	R/W	0	Define the assignment of the **Index$_X$** signal to be output to the Pipeline. It also affects the oFin signal.

IndexMap0	Index$_X$ signal for the Pipeline
0	Internal Index$_X$ output
1	Internal Index$_Y$ output
2	Internal Index$_Z$ output

3	Internal $Index_W$ output

IndexMap1[1:0] R/W 0 Definition of the $Index_X$ signal to be output to the Pipeline.

IndexMap1	$Index_Y$ signal for the Pipeline
0	Internal $Index_Y$ output
1	Internal $Index_Z$ output
2	nternal $Index_W$ output
3	nternal $Index_X$ output

IndexMap2[1:0] R/W 0 Definition of the IndexZ signal to be output to the Pipeline.

IndexMap2	IndexZ signal for the Pipeline
0	nternal IndexZ output
1	nternal IndexW output
2	nternal IndexX output
3	nternal IndexY output

IndexMap3[1:0] R/W 0 Definition of the IndexW signal to be output to the Pipeline.

IndexMap3	IndexW signal for the Pipeline
0	nternal IndexW output
1	nternal IndexX output
2	nternal IndexY output
3	nternal IndexZ output

DispMax[2:0] R/W 0 Specify the maximum number of instructions to be held for the relevant Pipeline (issued instructions minus returned instructions). This setting is used when the output buffer size and negotiation control cannot be adjusted.

DispMax	Maximum number of instructions to be held.
0	No Limit.
1	1
2	2
3	3
4	4
5	5
6	6
7	7

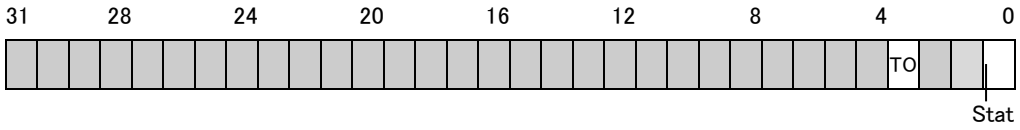
Link[1:0] R/W 0 Control the physical connection with the relevant Pipeline. The setting takes effect immediately upon configuration.

Link	Connection Type.
0	Connect the Pipeline.
1	Force terminate the Pipeline (Ground).
2	Terminate the Pipeline with a new entry (Short).
3	Temporarily hold the Pipeline (Open).

Lock R/W 0 Lock the arbitration of the logical channel that activates the relevant Pipeline until the processing of the logical channel is completed. Once the Pipeline is activated, the control of the same logical channel is occupied.

4.6.1.18. PI [n] Register

[Address: 0x0000_c008 + 256n]



Name	Type	Default	Description
TO	RWC	0	'1' indicates that the Pipeline is in a Time Out state. It

			shows that the 28–bit counter has completed one cycle. It is cleared with a '1' write.
Stat	R	0	'1' indicates that the Pipeline is in a Busy state (difference between the count of piVld & !piStall and poVld & !poStall). Since it is managed by the PSS, the observed state may not always match the actual Pipeline state exactly, depending on factors such as observation time.