TWIM is started by triggering the DMA.TX.START or DMA.RX.START tasks, and stopped by triggering the STOP task. After a STOP task, TWIM generates a STOPPED event when it has stopped.

After TWIM starts, the DMA.TX.START and DMA.RX.START tasks must not be triggered again until TWIM has issued a LASTRX, LASTTX, or STOPPED event.

TWIM can be suspended using the SUSPEND task, which is useful when using TWIM in a low priority interrupt context. When TWIM enters the SUSPEND state, it will automatically issue a SUSPENDED event while performing a continuous clock stretching. This continues until a RESUME task is received. TWIM cannot be stopped while it is suspended. The STOP task must be issued after TWIM resumes operation.

Note: Any ongoing byte transfer is allowed to complete before suspend is enforced. A SUSPEND task has no effect unless TWIM is actively involved in a transfer.

If a NACK is clocked in from the target, TWIM generates an ERROR event.

8.23.2 Shared resources

The TWIM peripheral shares registers and other resources with peripherals that have the same ID as TWIM. Therefore, all peripherals that have the same ID as TWIM must be disabled before TWIM can be configured and used.

Disabling shared peripherals will not reset any of the registers that are shared with TWIM. Configure all relevant TWIM registers to ensure they operate correctly.

See the Instantiation table in Instantiation on page 216 for details on peripherals and their IDs.

8.23.3 EasyDMA

EasyDMA is implemented by TWIM in order to access RAM without the CPU.

TWIM implements the following EasyDMA channels.

Channel	Type	Register Cluster
TXD	READER	TXD
RXD	WRITER	RXD

Table 64: TWIM EasyDMA channels

The RXD.PTR, TXD.PTR, RXD.MAXCNT, and TXD.MAXCNT registers are double-buffered. They are ready for the next transmission immediately after receiving the EVENTS_DMA.RX.READY or EVENTS_DMA.TX.READY event.

The STOPPED event indicates that EasyDMA is finished accessing the buffer in RAM.

See EasyDMA on page 27 for more detailed information.

8.23.4 TWIM write sequence

A TWIM write sequence is started by triggering the DMA.TX.START task. After the DMA.TX.START task has been triggered, TWIM generates a start condition on the TWI bus. This is followed by clocking out the address and the READ/WRITE bit set to `0` (WRITE = `0`, READ = `1`).

The target device address the controller wants to write to must match the clocked address. The READ/WRITE bit is followed by an ACK/NACK bit (ACK = `0` or NACK = `1`) generated by the target.

After receiving the ACK bit, TWIM clocks out the data bytes found in the transmit buffer located in RAM at the address specified in the TXD.PTR register. Each byte clocked out from TWIM is followed by an ACK/NACK bit clocked in from the target.