When buffered writes are enabled, RRAMC will collect as much data as possible in the internal write-buffer, before bulk-committing the buffer to RRAM.

When committing to RRAM, the commit operation updates only the data in RRAM memory that has modified values. Values that have not been altered remain unchanged in RRAM.

To use buffered writes, perform the following operations:

1. Enable writing using CONFIG.WEN, and configure CONFIG.WRITEBUFSIZE.
2. Write to RRAM memory in incrementing address order.

RRAMC commits the write-buffer when either of the following occurs:

- The write-buffer is full
- The address written is outside the buffer area
- There is a read operation from a 128-bit word line in the buffer that has already been written to

RRAMC stalls while the commit takes place, and additional wait-states can be observed for the bus access.

In addition to the automatic commit, a commit can also be triggered by the following:

- After a time-out waiting for a new write operation, configured in READYNEXTTIMEOUT
- When the COMMITWRITEBUF task is triggered

The manual triggers are useful in situations where it is crucial to ensure that the write buffer has been committed. Register BUFSTATUS.WRITEBUFEMPTY can be used to check if the write-buffer is empty, or if it contains uncommitted data.

**Note:** The internal write-buffer is volatile, and data loss may occur during a power failure or when entering System OFF mode with uncommitted data in the buffer. Having uncommitted data in the internal write-buffer will keep the RRAM active, and thus increase power consumption during sleep mode.

### 4.2.6.3 Erasing RRAM

RRAMC provides a mechanism to erase the whole RRAM in one operation by using the ERASE.ERASEALL register.

When ERASEALL is triggered, RRAMC will write `0xFFFFFFFF` to the entire RRAM memory, including user information configuration registers (UICR) and the secure information configuration region (SICR) . ERASEALL will not erase the factory information configuration registers (FICR).

This functionality can be blocked by ERASE protection. For details, see CTRL-AP — Control access port on page 822.

**Note:** Unlike the CTRL-AP ERASEALL operation that can be activated from a debugger, the RRAMC ERASEALL operation will not automatically grant access to the debug access port.

**Note:** The write-buffer must be committed before initiating an erase operation.

### 4.2.6.4 Region protection

The RRAM memory can be divided into several regions and these regions can be configured to restrict accesses.

Each region is configured using the REGION[n].ADDRESS and REGION[n].CONFIG registers.

When the region write-once feature is enabled, writes to a 32-bit words in the region are allowed only when the current data is `0xFFFFFFFF`, else the writes are ignored.