4.5 MRAPI non-blocking operations

The MRAPI provides both blocking and non-blocking versions of communication functions that may be delayed because the implementation requires synchronization between multiple nodes. The non-blocking version of functions is denoted by an _i() suffix. For example, the `mrapi_rmem_write()` function copies a data buffer from local memory to a remote shared memory buffer. Since the data copy operation might take many cycles, MRAPI also provides `mrapi_rmem_write_i()` function, which initiates the DMA operation and returns immediately. Like all non-blocking functions, `mrapi_rmem_write_i()` fills in a `mrapi_request_t` object before returning.

The `mrapi_request_t` object provides a unique identifier for each in-flight non-blocking operation. These 'request handles' can be passed to the `mrapi_test(), mrapi_wait()`, or `mrapi_wait_any()` methods in order to find out when the non-blocking operation has completed. When one of these API calls determines that a non-blocking request has finished, it returns indicating completion and fills in an `mrapi_status_t` object to indicate why the request completed. The status object contains an error code indicating whether the operation finished successfully or was terminated because of an error. The `mrapi_request_t` is an opaque data type and the user should not attempt to examine it.

Non-blocking operations may consume system resources until the programmer confirms completion by calling `mrapi_test(), mrapi_wait(),` or `mrapi_wait_any()`. Thus, the programmer should be sure to confirm completion of every non-blocking operation via these APIs. Alternatively, an in-flight operation can be cancelled by calling `mrapi_cancel()`. This function forces the operations specified by the `mrapi_request_t` object to stop immediately, releasing any system resources allocated in order to perform the operation.