ffb ffi ffbfiS ffi

S ffi ffi

ffi ffi fj fbffi

•

•

•

•

•

•

ffi

-

ffiffi fb ffiffiffbffi ffi
ffi ffi ffi ffi

•

•

•

•

•

ffi ffi fb ffiffiffbffi ffi ffb
ffi ffi ffi ffbffiS ffi ffb ffb

•

•

•

ffiffi fb ffiffffbffi ffi ffi ffb
`_size` ffj ffi ffi ffi

`_size`

-
-
-
-

every

-

ffi ffi ffi fflffi ffi ffb
fj fb ffi ffi

`ROCBLAS_DEVICE_MEMORY_SIZE`

-
-

`rocblas_status rocblas_set_device_memory_size(rocblas_handle, size_t size);`

-
-
-
-

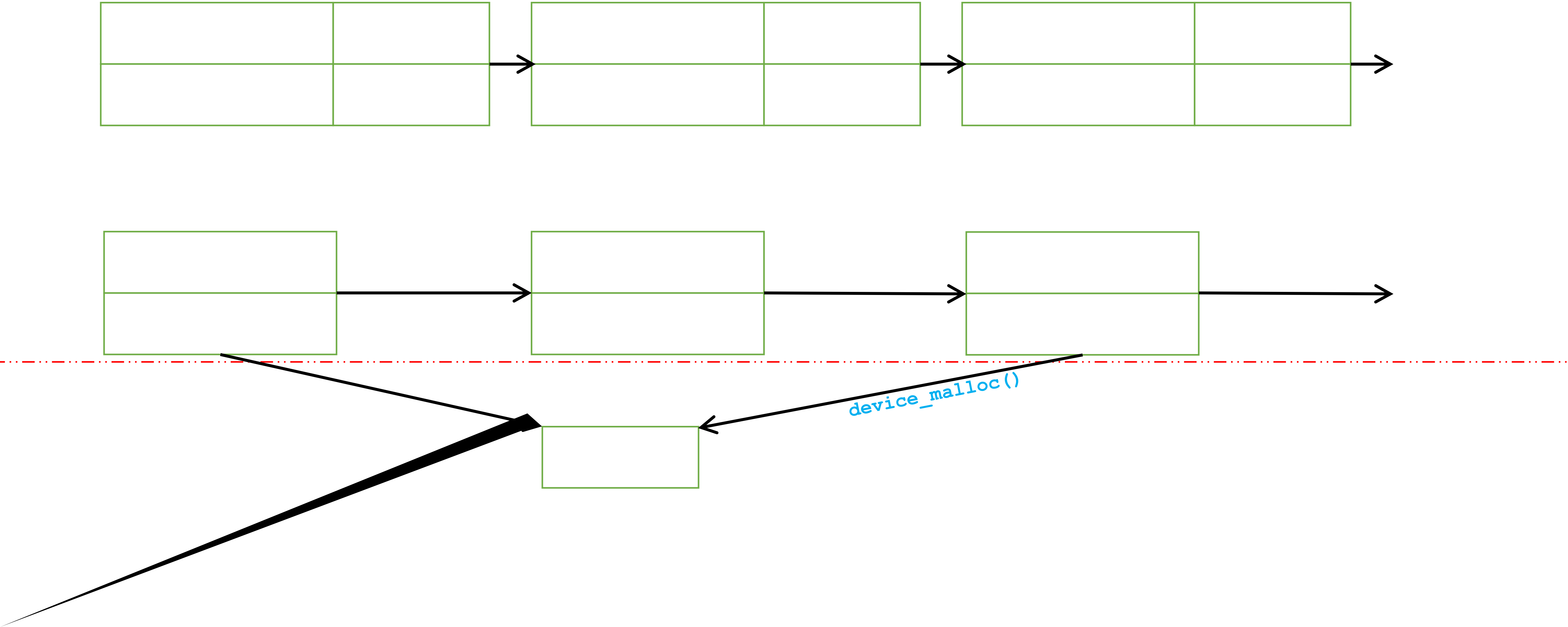
`rocblas_status rocblas_get_device_memory_size(rocblas_handle, size_t *size);`

- `*size`

`bool rocblas_is_managing_device_memory(rocblas_handle handle);`

-

fj



`rocblas_status rocblas_start_device_memory_size_query(rocblas_handle);`

-
-
- `rocblas_status_size_query_mismatch`
  `rocblas_status_success`
-
  `rocblas_handle`
- `*_ex` `void*` `size_t*`
- `_ex`

`rocblas_status rocblas_stop_device_memory_size_query(rocblas_handle, size_t* size);`

-
  `*size`
- `rocblas_status_size_query_mismatch`
  `rocblas_status_invalid_pointer` `size` `nullptr` `rocblas_status_success`

ffi ffi ffbiS ffi ffi ffi ffi

`bool _rocblas_handle::is_device_memory_size_query() const;`

-

`rocblas_status _rocblas_handle::set_optimal_device_memory_size(size...);`

-
-
- `rocblas_status_size_unchanged`
  `rocblas_status_size_increased`
  `rocblas_status_internal_error`
- `rocblas_status_success`

```
if(handle->is_device_memory_size_query())
{
   size_t size = m * n * sizeof(T);  // Compute optimal size
   return handle->set_optimal_device_memory_size(size);
}
```

size_t rocblas_sizeof_datatype(rocblas_datatype type)

- *runtime*
-
-
-

RETURN_ZERO_DEVICE_MEMORY_SIZE_IF_QUERIED(handle)

- return rocblas_status_size_unchanged;
-

```
rocblas_status rocblas_kernel(rocblas_handle handle, ...)
{
    RETURN_ZERO_DEVICE_MEMORY_SIZE_IF_QUERIED(handle);
    // ...
}
```

ffb ffi ffi ffi fflbiS ffi ffb

```
auto mem = handle->device_malloc(size...);
```

•

•

•

•

•

• `false`

• `void*`

• `std::tie(ptr1, ptr2, …)`

•

•

-
- device_malloc() std::tie(ptr1, ptr2, ptr3, …)
-
-
- constexpr
  device_malloc(1024, 256, size, 512)
  size+1792
-
-

```
void *buf1, *buf2, *buf3;
size_t bufsize1, bufsize2, bufsize3;
auto mem = handle->device_malloc(bufsize1, bufsize2, bufsize2);
if(!;
```

rocblas_status_memory_error

-

mem

mem

rocblas_status_memory_error

rocblas_status_perf_degraded

-

```
rocblas_status ret = rocblas_status_success;




        ret = rocblas_status_perf_degraded;

        ret = rocblas_status_memory_error;


return ret;
```

push_pointer_mode(rocblas_pointer_mode)

-
-
- rocblas_pointer_mode
-

```
auto saved_pointer_mode = handle->push_pointer_mode(rocblas_pointer_mode_host)


if(saved_pointer_mode == rocblas_pointer_mode_host)
```

ffi

`rocblas_handle handle`

)

~~rocblas_trsm_option option,~~
~~size_t* x_temp_size,~~
~~void* x_temp_workspace)~~

-

`option = rocblas_trsm_low_memory`

ffi

```
{
    // By default return success
    rocblas_status rb_memory_status = rocblas_status_success;

    // Compute the optimal size in bytes for maximum speed
    size_t x_temp_size = rocblas_sizeof_datatype(compute_type) * m * n;

    // If this call is a device memory size query,
    // return the size in bytes recommended for maximum speed
    if(handle->is_device_memory_size_query())
        return handle->set_optimal_device_memory_size(x_temp_size);

    // Attempt to allocate the optimal size
    auto x_temp_workspace = handle->device_malloc(x_temp_size);
    if(!x_temp_workspace)
    {
        // If optimal size is not available, try the smaller size
        x_temp_size = rocblas_sizeof_datatype(compute_type) * m;
        x_temp_workspace = handle->device_malloc(x_temp_size);

        // If the smaller size cannot be allocated, return error
        if(!x_temp_workspace)
            return rocblas_status_memory_error;

        // Set return status to indicate degraded performance
        rb_memory_status = rocblas_status_perf_degraded;
    }
```

ffi

```
// Pass the large or small x_temp_size and x_temp_workspace
rb_status = rocblas_trsm_ex_template<TRSM_BLOCK>(
                                    handle,
                                    side,
                                    uplo,
                                    trans_a,
                                    diag,
                                    m,
                                    n,
                                    static_cast<const float*>(alpha),
                                    static_cast<const float*>(a),
                                    lda,
                                    static_cast<float*>(b),
                                    ldb,
                                    static_cast<const float*>(invA),
                                    ld_invA,
                                    &x_temp_size,
                                    static_cast<float*>(x_temp_workspace));

 return rb_status != rocblas_status_success ? rb_status : rb_memory_status;
}
```