BMRuntime LIBRARY

User Guide

June 2018

Directory

Chapter 1. Introduction

BMRuntime is a library specifically designed for building high-performance neural networks applications, and these applications run on chip BM1682. BMRuntime provides a serial of simple and flexible interfaces, which make the deployment of neural networks' applications easy and efficient.

As an overview, the library provides many functions which can be classified as following:

- BM device operations
- BM context operations
- BM kernel operations
- BM memory operations
- BM networks operations

BM device operations include enumerating BM1682 device, opening device, querying it, configuring it and closing it.

BM context operations include context creation, context binding and context destruction.

BM kernel operations include kernel creating and kernel destruction.

BM memory operations include device memory allocation, host memory allocation, device memory freeing, host memory freeing and memory copying between host memory and device memory.

BM networks operations include registering neural networks, inferencing networks as input and cleaning up networks.

Chapter 2. General Description

2.1 Programming Model

The interfaces provided by BMRuntime are very flexible, and can satisfy different requirements of different scenes. In general, the application must initialize the handle to the BM context by calling the **bm_init()** function. After the application initializes BM context handle, it allocates device memory and host memory for computing data. Then it can call **bmruntime_bmkernel_create()** function to create a BM kernel. After BM kernel is created, applications can use it to submit the commands.

If an application has got a compiled neural network, it does not need creating a BM kernel. It can call **bmnet_register()** to register a BM network, then call **bmnet_inference()** to do inference. After application finishes the work, it must call **bmnet_cleanup()** to destroy the networks. For kernel created applications, it must call **bmruntime_bmkernel_destroy()** to destroy the kernel. Finally, the application must call **bm_exit()** to exit gracefully and free all occupied resources.

2.2 Requirements

To run applications with BMRuntime, BM1682 is required.

Chapter 3. BMRuntime Data Type References

3.1 bmerr_t

bmerr_t is a enum type which enumerates all API returned status. The values are listed as follows.

Value	Description
BM_SUCCESS	API call returns successfully
BM_ERR_AGAIN	API call fails due to device not ready
BM_ERR_FAILURE	API call returns general failure
BM_ERR_TIMEOUT	API call fails due to time out
BM_ERR_UNINITIALIZED	API call fails due to device un-initialized
BM_ERR_INVALID_ARGUMENT	API call fails due to invalid argument
BM_ERR_NOMEM	API call fails due to insufficient memory
BM_ERR_DATA	API call fails due to data error
BM_ERR_BUSY	API call fails due to device being busy
BM_ERR_NOT_SUPPORTED	API call fails due to unsupported parameters

3.2 bmctx_t

bmctx_t is a pointer type which points to a BM context.

3.3 bmdev_t

bmdev_t is a pointer type which points to a BM device.

3.4 bm_devinfo_t

bm_devinfo_t is a structure type which describes device info.
It includes the following members:

Value	Description
chip	A enum value describes chip version.
npu_num	A uint32_t type value describes NPU number.
eu_num	A uint32_t type value describes EU number.
lmem_size	A uint32_t type value describes local memory size.
lmem_banks	A uint32_t type value describes local memory bank number.
gmem_size	A uint32_t type value describes global memory size.

3.5 bmmem_t

bmmem_t is a pointer type which points to BM memory structure.
bmmem_device_t, bmmem_host_t and bmmem_system_t are all alias of bmmem_t.

3.6 bmnet_info_t

bmnet_info_t is a structure type which describes BM network's input info.
It includes the following members:

Value	Description
batch_size	A uint32_t type value describes batch size.
weight	A uint8_t pointer type value which points to the weight memory.
weight_size	A size_t type value describes weight memory size.
cmdbuf	A uint8_t pointer type value which points to the command buffer.
cmdbuf_size	A size_t type value describes command buffer size.
neuron_size	A size_t type value describes neuron memory size.
input_size	A size_t type value describes input memory size.
output_offset	A uint64_t type value describes output neuron offset to neuron base.
output_size	A size_t type value describes output neuron size in bytes.

3.7 bmnet_output_info_t

bmnet_output_info_t is a structure type which describes BM network's output info.
It includes the following members:

Value	Description
output_size	Total output size of inference.
output_num	Output number.
name_array	Array of pointers to the output names
shape_array	Array of pointers to the output shapes
threshold_array	Array of pointers to the output threshold（meaningless on BM1682）

3.8 bmnet_t

bmnet_t is a pointer type which points to BM networks structure.

3.9 bmkernel_handle_t

bmkernel_handle_t is a pointer type which points to BM kernel.

Chapter 4. BMRuntime API References

In this chapter, all the API's in the BMRuntime library are described in detail.

4.1 bm_init

```
bmerr_t    bm_init(
    int            index,
    bmctx_t     *ctx)
```

bm_init() initializes BM device and creates a handle to BM context.

Parameter	Type	Description
index	Input	The index of BM device.
ctx	Output	The pointer of BM context handle.

4.2 bm_exit

```
void    bm_exit(
    bmctx_t    ctx)
```

bm_exit() must be called before application exits. It will release all internal resources.

Parameter	Type	Description
ctx	Input	The BM context handle which is created by bm_init().

4.3 bm_enum_devices

```
void    bm_enum_devices(
    int                *count,
    bm_devinfo_t     devinfo[])
```

bm_enum_devices() enumerates all BM devices in the system.

Parameter	Type	Description
count	Output	The count of BM device.
devinfo	Output	The array of device info.

4.4 bm_device_open

```
bmerr_t    bm_device_open(
    int             index,
    bmdev_t      *dev)
```

bm_device_open() opens a BM device.

Parameter	Type	Description
index	Input	The index of BM device.
dev	Output	The pointer of BM device handle.

4.5 bm_device_close

```
void        bm_device_close(
            bmdev_t       dev)
```

bm_device_close() closes an opened BM device.

Parameter	Type	Description
dev	Input	The BM device handle.

4.6 bm_device_query

```
bmerr_t     bm_device_query(
            bmdev_t       dev,
            int           id,
            void          *buf)
```

bm_device_query() always returns BM_ERR_NOT_SUPPORTED now.

4.7 bm_device_config

```
bmerr_t     bm_device_config(
            bmdev_t       dev,
            int           id,
            void          *buf)
```

bm_device_config() always returns BM_ERR_NOT_SUPPORTED now.

4.8 bm_device_get_info

```
bm_devinfo_t    bm_device_get_info(
                bmdev_t     dev)
```

bm_device_get_info() return a BM device information.

Parameter	Type	Description
dev	Input	The BM device handle.

4.9 bm_context_create

```
bmerr_t       bm_context_create(
                bmctx_t         *ctx)
```

bm_context_create() creates a BM context.

Parameter	Type	Description
ctx	Output	The pointer of BM context handle.

4.10 bm_context_destroy

```
void      bm_context_destroy(
            bmctx_t         ctx)
```

bm_context_destroy() destroys a BM context.

Parameter	Type	Description
ctx	Input	The BM context handle.

4.11 bm_bind_device

```
bmerr_t     bm_bind_device(
              bmctx_t         ctx,
              bmdev_t         dev)
```

bm_bind_device() binds a BM context with a BM device.

Parameter	Type	Description
ctx	Input	The BM context handle.
dev	Input	The BM device handle.

4.12 bm_unbind_device

```
void        bm_unbind_device(
              bmctx_t         ctx)
```

bm_unbind_device() unbinds a BM context with the BM device.

Parameter	Type	Description
ctx	Input	The BM context handle.

4.13 bm_get_device

```
bmdev_t      bm_get_device(
             bmctx_t      ctx)
```

bm_get_device() returns the BM device handle which is bound with the BM context.

Parameter	Type	Description
ctx	Input	The BM context handle.

4.14 bmruntime_bmkernel_create

```
bmerr_t      bmruntime_bmkernel_create(
             bmctx_t                 ctx,
             bmkernel_handle_t       *p_bk_ctx)
```

bmruntime_kernel_create() creates a BM kernel.

Parameter	Type	Description
ctx	Input	The BM context handle.
p_bk_ctx	Output	The pointer of BM kernel handle.

bmruntime_bmkernel_create() creates a BM kernel with the BM context.

4.15 bmruntime_bmkernel_destroy

```
void     bmruntime_bmkernel_destroy(
         bmctx_t      ctx)
```

bmruntime_bmkernel_destroy() destroys the BM kernel with the BM context.

Parameter	Type	Description
ctx	Input	The BM context handle.

4.16 bmmem_device_alloc_raw

```
bmmem_device_t    bmmem_device_alloc_raw(
                  bmctx_t      ctx,
                  size_t       size)
```

bmmem_device_alloc_raw() allocates device memory as the input size.

Parameter	Type	Description
ctx	Input	The BM context handle.
size	Input	The size of device memory.

4.17 bmmem_device_prealloc_raw

```
bmmem_device_t      bmmem_device_prealloc_raw(
                    bmctx_t          ctx,
                    size_t           size,
                    uint64_t         addr)
```

bmmem_device_prealloc_raw() allows application to allocate device memory that has been allocated by bmmem_device_alloc_raw(). Pre-allocate memory size and address should not exceed the size and address scope of memory allocated by bmmem_device_alloc_raw().

Parameter	Type	Description
ctx	Input	The BM context handle.
size	Input	The size of pre-allocate device memory.
addr	Input	The address of pre-allocate device memory.

4.18 bmmem_device_alloc

```
bmmem_device_t    bmmem_device_alloc(
                  bmctx_t          ctx,
                  bmshape_t       *shape)
```

bmmem_device_alloc() allocates device memory as the input shape.

Parameter	Type	Description
ctx	Input	The BM context handle.
shape	Input	The shape of device memory.

4.19 bmmem_device_prealloc

```
bmmem_device_t      bmmem_device_prealloc(
                    bmctx_t          ctx,
                    bmshape_t        *shape,
                    uint64_t         addr)
```

bmmem_device_prealloc() is similar with bmmem_device_prealloc_raw(), it allows application to allocate device memory that has been allocated.

Parameter	Type	Description
ctx	Input	The BM context handle.
shape	Input	The shape of pre-allocate device memory.
addr	Input	The address of pre-allocate device memory.

4.20 bmmem_device_free

```
void        bmmem_device_free(
            bmctx_t                 ctx,
            bmmem_device_t      mem)
```

bmmem_device_free() frees the device memory that are allocated by the above allocating functions.

Parameter	Type	Description
ctx	Input	The BM context handle.
mem	Input	The device memory handle.

4.21 bmmem_host_alloc

```
bmmem_host_t    bmmem_host_alloc(
            bmctx_t                 ctx,
            bmshape_t               *shape)
```

bmmem_host_alloc() always returns BM_ERR_NOT_SUPPORTED now.

4.22 bmmem_host_free

```
void        bmmem_host_free(
            bmctx_t                 ctx,
            bmmem_host_t        mem)
```

bmmem_host_free() always returns BM_ERR_NOT_SUPPORTED now.

4.23 bmmem_device_size

```
size_t      bmmem_device_size(
            bmctx_t                 ctx,
            bmmem_device_t      mem)
```

bmmem_device_size() returns the device memory size.

Parameter	Type	Description
ctx	Input	The BM context handle.
mem	Input	The device memory handle.

4.24 bmmem_device_addr

```
uint64_t        bmmem_device_addr(
                bmctx_t                 ctx,
                bmmem_device_t          mem)
```

bmmem_device_addr() returns the device memory address.

Parameter	Type	Description
ctx	Input	The BM context handle.
mem	Input	The device memory handle.

4.25 bmmem_host_v_addr

```
void*       bmmem_host_v_addr(
            bmctx_t             ctx,
            bmmem_host_t        mem)
```

bmmem_host_v_addr() always returns BM_ERR_NOT_SUPPORTED now.

4.26 bmmem_host_p_addr

```
uint64_t    bmmem_host_p_addr(
            bmctx_t             ctx,
            bmmem_host_t        mem)
```

bmmem_host_p_addr() always returns BM_ERR_NOT_SUPPORTED now.

4.27 bm_memcpy_s2d

```
bmerr_t     bm_memcpy_s2d(
            bmctx_t             ctx,
            bmmem_device_t      dst,
            uint8_t*            src)
```

bm_memcpy_s2d() copy system memory data to device memory. s means system, d means device.

Parameter	Type	Description
ctx	Input	The BM context handle.
dst	Input	The device memory handle.
src	Input	The system memory pointer.

4.28 bm_memcpy_d2s

```
bmerr_t        bm_memcpy_d2s(
                 bmctx_t              ctx,
                 uint8_t*             dst,
                 bmmem_device_t       src)
```

bm_memcpy_d2s copy device memory data to system memory.

Parameter	Type	Description
ctx	Input	The BM context handle.
dst	Input	The system memory pointer.
src	Input	The device memory handle.

4.29 bmnet_register

```
bmerr_t        bmnet_register(
                 bmctx_t              ctx,
                 bmnet_info_t        *info,
                 bmnet_t             *net)
```

bmnet_register() registers a compiled neuron network.

Parameter	Type	Description
ctx	Input	The BM context handle.
info	Input	The BM network info.
net	Output	The registered network handle.

4.30 bmnet_register_bmodel

```
bmerr_t        bmnet_register_bmodel (
                 bmctx_t              ctx,
                 char                *bmodel,
                 bmnet_t             *net)
```

bmnet_register_bmodel() registers a neuron network by bmodel file.

Parameter	Type	Description
ctx	Input	The BM context handle.
bmodel	Input	bmodel filename.
net	Output	The registered network handle.

4.31 bmnet_register_noalloc

```
bmerr_t         bmnet_register_noalloc(
                  bmctx_t              ctx,
                  bmnet_info_t         *info,
                  bmnet_t              *net)
```

bmnet_register_noalloc() registers a compiled neuron network without allocating weight and neuron device memory.

Parameter	Type	Description
ctx	Input	The BM context handle.
info	Input	The BM network info.
net	Output	The registered network handle.

4.32 bmnet_set_input_shape

```
bmerr_t         bmnet_set_input_shape(
                  bmnet_t        net,
                  shape_t        input_shape)
```

bmnet_set_input_shape () sets a input shape for a registered BM network. The bmodel support different input shapes, the API can set one of them.

Parameter	Type	Description
net	Input	The BM network handle.
input_shape	Input	The input shape.

4.33 bmnet_get_output_info

```
bmerr_t         bmnet_get_output_info(
                  bmnet_t                 net,
                  bmnet_output_info_t     *output_info)
```

bmnet_get_output_info () sets a input shape for a registered BM network.

Parameter	Type	Description
net	Input	The BM network handle.
output_info	Output	The output info.

4.34 bmnet_cleanup

```
void        bmnet_cleanup(
              bmnet_t     net)
```

bmnet_cleanup() cleans up a registered BM network.

Parameter	Type	Description
net	Input	The BM network handle.

4.35 bmnet_run

```
bmerr_t     bmnet_run(
              bmnet_t   net)
```

bmnet_run() runs a registered BM network.

Parameter	Type	Description
net	Input	The BM network handle.

4.36 bmnet_weight_devmem

```
bmmem_device_t    bmnet_weight_devmem(
                    bmnet_t           net)
```

bmnet_weight_devmem() retrieves the weight device memory handler from a registered BM network.

Parameter	Type	Description
net	Input	The BM network handle.

4.37 bmnet_neuron_devmem

```
bmmem_device_t    bmnet_neuron_devmem(
                    bmnet_t           net)
```

bmnet_neuron_devmem() retrieves neuron device memory handler from a registered BM network.

Parameter	Type	Description
net	Input	The BM network handle.

4.38 bmnet_input_devmem

```
bmmem_device_t      bmnet_input_devmem(
                      bmnet_t           net)
```

bmnet_input_devmem() retrieves input device memory handler from a registered BM network.

Parameter	Type	Description
net	Input	The BM network handle.

4.39 bmnet_output_devmem

```
bmmem_device_t      bmnet_output_devmem(
                      bmnet_t           net)
```

bmnet_output_devmem() retrieves output device memory handler from a registered BM network.

Parameter	Type	Description
net	Input	The BM network handle.

4.40 bmnet_load_input

```
bmerr_t         bmnet_load_input(
                  bmnet_t       net,
                  uint8_t       *input)
```

bmnet_load_input() loads input data for a registered BM network.

Parameter	Type	Description
net	Input	The BM network handle.
input	Input	The input data pointer.

4.41 bmnet_import_weight_devmem

```
bmerr_t     bmnet_import_weight_devmem(
              bmnet_t           net,
              bmmem_device_t    weight_mem)
```

bmnet_import_weight_devmem() imports weight device memory for a registered BM network. application should allocate weight device memory firstly, then call it to import weight memory. This function and bmnet_import_neuron_devmem() function are usually used with bmnet_register_noalloc() function. Application can register BM network without allocating

weight and neuron device memory, and then use these two functions to import weight and neuron memory.

Parameter	Type	Description
net	Input	The BM network handle.
weight_mem	Input	The weight device memory handle.

4.42 bmnet_import_neuron_devmem

```
bmerr_t      bmnet_import_neuron_devmem(
                bmnet_t              net,
                bmmem_device_t       neuron_mem)
```

bmnet_import_neuron_devmem() imports neuron device memory for a registered BM network. application should allocate neuron device memory firstly, then call it to import neuron memory.

Parameter	Type	Description
net	Input	The BM network handle.
neuron_mem	Input	The neuron device memory handle.

4.43 bmnet_store_output

```
bmerr_t      bmnet_store_output (
                bmnet_t          net,
                uint8_t          *output)
```

bmnet_store_output() stores output data for a registered BM network. Application uses this function to copy output data from device memory to host memory.

Parameter	Type	Description
net	Input	The BM network handle.
output	Input	The output buffer pointer.

4.44 bmnet_store_neuron

```
bmerr_t      bmnet_store_neuron(
                bmnet_t        net,
                uint64_t       neuron_offset,
                int            neuron_size,
                uint8_t        *neuron)
```

bmnet_store_neuron() stores neuron data for a registered BM network. Application uses this function to copy neuron data from device memory to host memory.

Parameter	Type	Description
net	Input	The BM network handle.
neuron_offset	Input	The offset of neuron buffer.
neuron_size	Input	The neuron buffer size.
neuron	Input	The pointer to the neuron buffer.

4.45 bmnet_load_neuron

```
bmerr_t     bmnet_load_neuron(
              bmnet_t       net,
              uint64_t      neuron_offset,
              int           neuron_size,
              uint8_t       *neuron)
```

bmnet_load_neuron() loads neuron data for a registered BM network.

Parameter	Type	Description
net	Input	The BM network handle.
neuron_offset	Input	The offset of neuron buffer.
neuron_size	Input	The neuron buffer size.
neuron	Input	The pointer to the neuron buffer.

4.46 bmnet_inference

```
bmerr_t     bmnet_inference(
              bmnet_t       net,
              uint8_t       *input,
              uint8_t       *output)
```

bmnet_inference() runs inference with a registered BM network.

Parameter	Type	Description
net	Input	The BM network handle.
input	Input	The input buffer pointer.
output	Input	The output buffer pointer.

4.47 bmnet_inference_once

```
bmerr_t      bmnet_inference_once(
                bmctx_t          ctx,
                uint8_t          *input,
                uint8_t          *output,
                uint32_t         batch_size,
                uint8_t          *weight,
                size_t           weight_size,
                uint8_t          *cmdbuf,
                size_t           cmdbuf_size,
                size_t           neuron_size,
                size_t           input_size,
                uint64_t         output_offset,
                size_t           output_size)
```

bmnet_inference_once() runs inference all in one function, it includes the functions of bmnet_register(), bmnet_inference() and bmnet_cleanup() in one call.

Parameter	Type	Description
ctx	Input	The BM context handle.
input	Input	The input buffer pointer.
output	Input	The output buffer pointer.
batch_size	Input	The batch size.
weight	Input	The weight buffer pointer.
weight_size	Input	The weight buffer size.
cmdbuf	Input	The command buffer pointer.
cmdbuf_size	Input	The command buffer size.
neuron_size	Input	The neuron buffer size.
input size	Input	The input buffer size.
output_offset	Input	The output neuron offset to neuron base.
output_size	Input	The output neuron size in bytes.

4.48 bmnet_data_transition

```
int             bmnet_data_transition(
                        bmctx_t       ctx,
                        u64           gaddr_a,
                        int           input_n,
                        int           input_c,
                        int           input_h,
                        int           input_w,
                        int           stride_n,
                        int           stride_c,
                        int           stride_h,
                        float         S,
                        float         B,
                        u64           gaddr_r)
```

bmnet_data_transition() copies data between global memory. It uses NPU to copy data. It provides two parameters of scaling factor and bias value.

Parameter	Type	Description
ctx	Input	The BM context handle.
gaddr_a	Input	The source data global memory address.
input_n	Input	The input data N.
input_c	Input	The input data C.
input_h	Input	The input data H.
input_w	Input	The input data W.
stride_n	Input	The n dimension stride.
stride_c	Input	The c dimension stride.
stride_h	Input	The h dimension stride.
S	Input	The scaling factor.
B	Input	The bias value.
gaddr_r	Input	The destination data global memory address.

4.49 bmnet_data_copy_u8

```
int             bmnet_data_copy_u8(
                    bmctx_t         ctx,
                    u64             gaddr_s,
                    int             input_n,
                    int             input_c,
                    int             input_h,
                    int             input_w,
                    int             stride_n,
                    int             stride_c,
                    int             stride_h,
                    u64             gaddr_d)
```

bmnet_data_copy_u8() copies data between global memory. It uses NPU to copy data. It copies data as unsigned 8 bits integer.

Parameter	Type	Description
ctx	Input	The BM context handle.
gaddr_s	Input	The source data global memory address.
input_n	Input	The input data N.
input_c	Input	The input data C.
input_h	Input	The input data H.
input_w	Input	The input data W.
stride_n	Input	The n dimension stride.
stride_c	Input	The c dimension stride.
stride_h	Input	The h dimension stride.
gaddr_d	Input	The destination data global memory address.

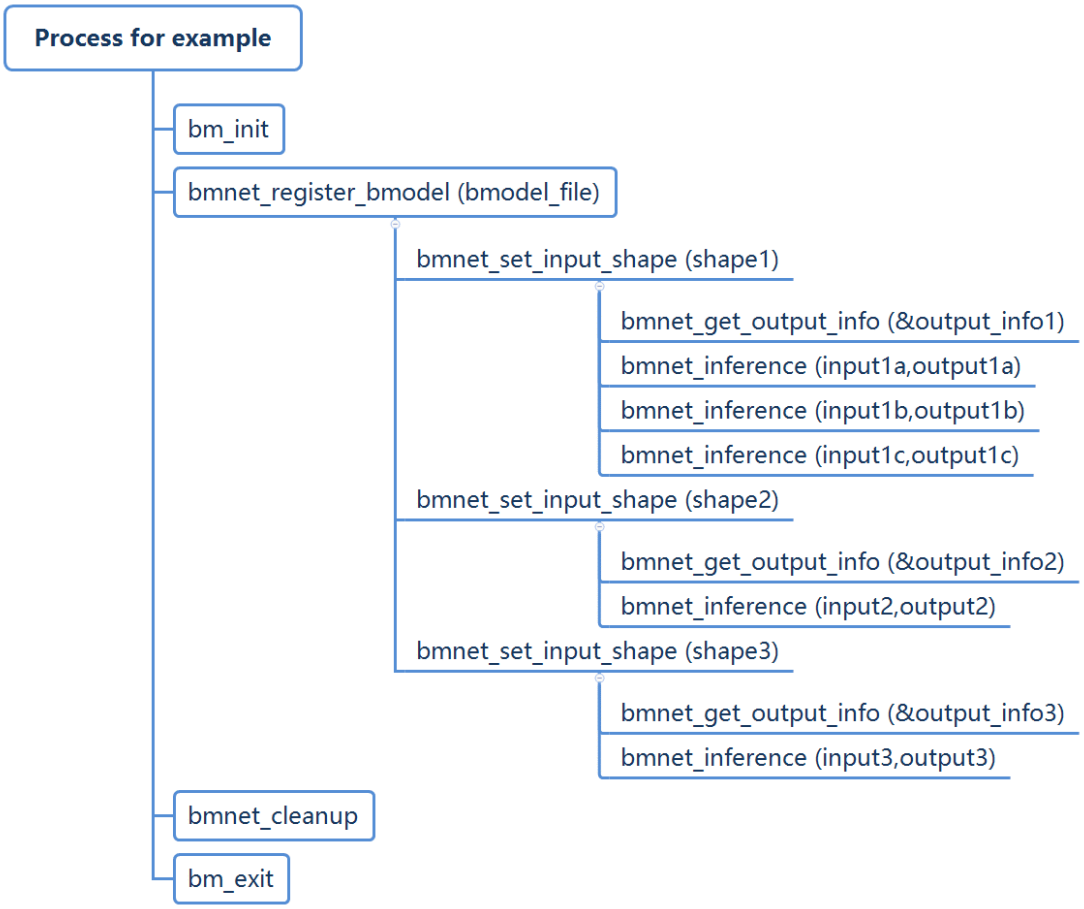
Chapter 5. Examples

In this chapter, some examples using flow-process diagram and Pseudocode, to show how to use the APIs of BMRuntime.

5.1 register and inference by bmodel

As one bmodel file support multiple shapes, bmnet_register_bmodel once can inference different shapes.

1) flow-process diagram



2) Pseudocode

```
bmerr_t ret;
bmctx_t ctx;
ret = bm_init (0, &ctx);
if (BM_SUCCESS != ret) { // exception code }

char * bmodel_file = "~/models/test.bmodel";
ret = bmnet_register_bmodel (ctx, bmodel_file, &net);
if (BM_SUCCESS != ret) { // exception code }
```

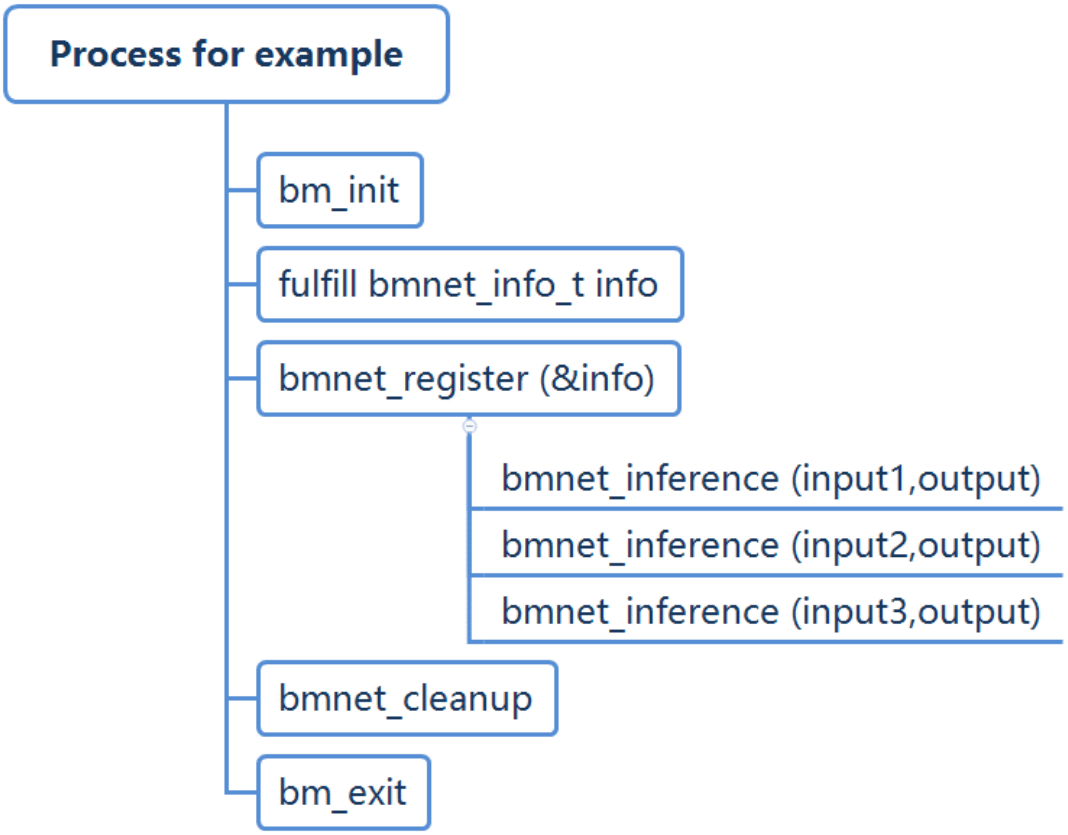
```
// do inference with shape[1,1,28,28]
shape_t input_shape = shape_t4 (1, 1, 28, 28);
// set shape, be careful [1,1,28,28] should be one shape in bmodel_file; if not, return error
ret = bmnet_set_input_shape (net, input_shape);
if (ret != BM_SUCCESS) { // exception code}

bmnet_output_info_t output_info;
bmnet_get_output_info(net, &output_info);
{ // deal with output_info, such as output_info.output_size, output_info.sub_outputs}
output = (uint8_t*) malloc(output_info.output_size);

{ //uint8_t *input1a, input buffer read from one input file }
bmnet_inference(net, input1a, output);
{ // deal with output data }

{ //uint8_t *input1b, input buffer read from one input file }
bmnet_inference(net, input1b, output);
{ // deal with output data }

{ //uint8_t *input1c, input buffer read from one input file }
bmnet_inference(net, input1c, output);
{ // deal with output data }
free(output)

// do inference with shape[4,1,28,28]
shape_t input_shape = shape_t4 (4, 1, 28, 28);
ret = bmnet_set_input_shape (net, input_shape);
if (ret != BM_SUCCESS) { // exception code}

bmnet_get_output_info(net, &output_info);
{ // deal with output_info, such as output_info.output_size, output_info.sub_outputs}
output = (uint8_t*) malloc(output_info.output_size);

{ // uint8_t *input2, input buffer read from one input file }
bmnet_inference(net, input2, output);
{ // deal with output data };
free (output);

// do inference with shape[8,1,28,28]
shape_t input_shape = shape_t4 (8, 1, 28, 28);
ret = bmnet_set_input_shape (net, input_shape);
if (ret != BM_SUCCESS) { // exception code}

bmnet_get_output_info(net, &output_info);
```

```
{ // show output_info, such as output_info.output_size, output_info.sub_outputs}
output = (uint8_t*) malloc(output_info.output_size);

{ // uint8_t *input3, input buffer read from one input file }
bmnet_inference(net, input3, output);
{ // deal with output data };
free (output);

// release resource
bmnet_cleanup(net);
bm_exit(ctx);
```

5.2 register and inference by bmnet_info_t

As bmnet_info_t only support one shape, bmnet_register once can inference only one shape.

1) flow-process diagram



2) Pseudocode

```
bmerr_t ret;
bmctx_t ctx;
ret = bm_init (0, &ctx);
if (BM_SUCCESS != ret) { // exception code }
```

```
// fulfill bmnet_info_t structure
bmnet_info_t info = {.batch_size = batch_size,
                     .weight = weight,
                     .weight_size = weight_size,
                     .cmdbuf = cmdbuf,
                     .cmdbuf_size = cmdbuf_size,
                     .neuron_size = neuron_size,
                     .input_size = input_size,
                     .output_offset = output_offset,
                     .output_size = output_size};
bmnet_t net;
bmnet_register (ctx, &info, &net);
if (BM_SUCCESS != ret) { // exception code }

// do inference with the same shape
{ // uint8_t *input1, input buffer read from one input file }
bmnet_inference(net, input1, output);
{ // deal with output data };

{ // uint8_t *input2, input buffer read from one input file }
bmnet_inference(net, input2, output);
{ // deal with output data };

{ // uint8_t *input3, input buffer read from one input file }
bmnet_inference(net, input3, output);
{ // deal with output data };

// release resource
bmnet_cleanup(net);
bm_exit(ctx);
```