```
typedef enum {
  KR_VALGRIND_STRESSOR = 1,
  KR_QUVI,
  KR_TEST_TASK
} kr_mt_type;

typedef enum {
  KR_MT_READY = 1,
  KR_MT_RUNNING,
  KR_MT_FAILED,
  KR_MT_TIMEDOUT,
  KR_MT_SUCCEEDED,
  KR_MT_STOPPED
} kr_mt_state;

typedef enum {
  KR_MT_CREATE = 1,
  KR_MT_START,
  KR_MT_PATCH,
  KR_MT_PROGRESS,
  KR_MT_FAILURE,
  KR_MT_SUCCESS,
  KR_MT_STOP,
  KR_MT_TIMEOUT
} kr_mt_event_type;

static const kr_mt_state transitions[] = {
  [0] = 0,
    [KR_MT_CREATE] = KR_MT_READY,
    [KR_MT_START] = KR_MT_RUNNING,
    [KR_MT_PATCH] = 0,
    [KR_MT_PROGRESS] = 0,
    [KR_MT_FAILURE] = KR_MT_FAILED,
    [KR_MT_TIMEOUT] = KR_MT_TIMEDOUT,
    [KR_MT_STOP] = KR_MT_STOPPED,
    [KR_MT_SUCCESS] = KR_MT_SUCCEEDED
};

static const int startable_states[] = {
  [0] = 0,
  [KR_MT_READY] = 1,
  [KR_MT_RUNNING] = 0,
  [KR_MT_FAILED] = 1,
  [KR_MT_TIMEDOUT] = 1,
  [KR_MT_STOPPED] = 1,
  [KR_MT_SUCCEEDED] = 0,
};

static const int stopable_states[] = {
  [0] = 0,
  [KR_MT_READY] = 0,
  [KR_MT_RUNNING] = 1,
  [KR_MT_FAILED] = 0,
  [KR_MT_TIMEDOUT] = 0,
  [KR_MT_STOPPED] = 0,
  [KR_MT_SUCCEEDED] = 0,
};

#include "media_operations_types.h"

typedef struct kr_mt kr_mt;
typedef struct kr_mt_spec kr_mt_spec;
typedef struct kr_media_ops_server kr_media_ops_server;

typedef struct {
  kr_mt *mt;
  kr_patchset patchset;
  kr_mt_event_type event_type;
} kr_mt_event;
```

```
typedef void (kr_mt_event_cb)(kr_mt_event *);

struct kr_mt {
  kr_mt_event_type last_event;
  struct timespec last_event_time;
  void *handle;
  void *user;
  kr_mt_event_cb *event_cb;
  kr_mt_info info;
  int fd;
  kr_media_ops_server *kmops;
  kr_loop *loop;
};

typedef int (kr_mt_create)(kr_mt *, kr_mt_event_cb);
typedef int (kr_mt_ctl)(kr_mt *, kr_patchset *);
typedef int (kr_mt_start)(kr_mt *);
typedef int (kr_mt_stop)(kr_mt *);
typedef int (kr_mt_destroy)(kr_mt *);

struct kr_mt_spec {
  kr_mt_create *create;
  kr_mt_ctl *ctl;
  kr_mt_start *start;
  kr_mt_stop *stop;
  kr_mt_destroy *destroy;
};
```

```
static const kr_mt_type kr_mt_types[] = {
  [0] = 0,
  [KR_VALGRIND_STRESSOR] = KR_VALGRIND_STRESSOR,
  [KR_QUVI] = KR_QUVI,
  [KR_TEST_TASK] = KR_TEST_TASK
};

static const kr_mt_spec media_tasks[] = {
  [0] = {
    .create = NULL,
    .ctl = NULL,
    .start = NULL,
    .stop = NULL,
    .destroy = NULL
  },
  [KR_VALGRIND_STRESSOR] = {
    .create = kr_valgrind_stressor_create,
    .ctl = kr_valgrind_stressor_ctl,
    .start = kr_valgrind_stressor_start,
    .stop = kr_valgrind_stressor_stop,
    .destroy = kr_valgrind_stressor_destroy,
  },
  [KR_QUVI] = {
    .create = kr_quvi_create,
    .ctl = kr_quvi_ctl,
    .start = kr_quvi_start,
    .stop = kr_quvi_stop,
    .destroy = kr_quvi_destroy,
  },
  [KR_TEST_TASK] = {
    .create = kr_test_task_create,
    .ctl = kr_test_task_ctl,
    .start = kr_test_task_start,
    .stop = kr_test_task_stop,
    .destroy = kr_test_task_destroy,
  }
};

static const int nmt_types = (sizeof(media_tasks) / sizeof(media_tasks[0]));
```

```
typedef struct kr_mt_runtime_properties {
  int queue_pos;
  int priority;
  kr_mt_state state;
} kr_mt_runtime_properties;

typedef struct kr_valgrind_stressor_info {
} kr_valgrind_stressor_info;

typedef struct kr_quvi_info {
} kr_quvi_info;

typedef struct kr_test_task_info {
  int position;
} kr_test_task_info;

typedef struct kr_mt_info {
  kr_mt_type type;
  kr_mt_runtime_properties runtime_props;
  union {
    kr_valgrind_stressor_info valgrind_stressor;
    kr_quvi_info quvi;
    kr_test_task_info test_task;
  };
} kr_mt_info;
```

```
#include "test_task.h"

int kr_test_task_destroy(kr_mt *mt) {
  mt->info.runtime_props.state = 0;
  return 0;
}

int kr_test_task_create(kr_mt *mt, kr_mt_event_cb event_cb) {
  mt->event_cb = event_cb;
  mt->info.type = KR_TEST_TASK;
  mt->info.runtime_props.state = KR_MT_READY;
  kr_media_ops_raise_event(mt, KR_MT_CREATE);
  return 0;
}

int kr_test_task_ctl(kr_mt *mt, kr_patchset *patchset) {
  return 0;
}

int kr_test_task_start(kr_mt *mt) {
  mt->info.runtime_props.state = KR_MT_RUNNING;
  kr_media_ops_raise_event(mt, KR_MT_START);
  return 0;
}

int kr_test_task_stop(kr_mt *mt) {
  int ret;
  mt->info.runtime_props.state = KR_MT_STOPPED;
  kr_media_ops_raise_event(mt, KR_MT_STOP);
  return 0;
}
```