NAME

ibv_query_qp - get the attributes of a queue pair (QP)

SYNOPSIS

#include <infiniband/verbs.h>

int ibv_query_qp(struct ibv_qp **qp***, struct ibv_qp_attr** **attr***,**
int *attr_mask***,**
struct ibv_qp_init_attr **init_attr***);**

DESCRIPTION

ibv_query_qp() gets the attributes specified in *attr_mask* for the QP *qp* and returns them through the pointers *attr* and *init_attr*. The argument *attr* is an ibv_qp_attr struct, as defined in <infiniband/verbs.h>.

```
struct ibv_qp_attr {
        enum ibv_qp_state       qp_state;           /* Current QP state */
        enum ibv_qp_state       cur_qp_state;       /* Current QP state - irrelevant for ibv_query_qp */
        enum ibv_mtu            path_mtu;           /* Path MTU (valid only for RC/UC QPs) */
        enum ibv_mig_state      path_mig_state;     /* Path migration state (valid if HCA supports APM) */
        uint32_t                qkey;               /* Q_Key of the QP (valid only for UD QPs) */
        uint32_t                rq_psn;             /* PSN for receive queue (valid only for RC/UC QPs) */
        uint32_t                sq_psn;             /* PSN for send queue (valid only for RC/UC QPs) */
        uint32_t                dest_qp_num;        /* Destination QP number (valid only for RC/UC QPs) */
        int                     qp_access_flags;    /* Mask of enabled remote access operations (valid only for RC/UC QPs)
        struct ibv_qp_cap       cap;                /* QP capabilities */
        struct ibv_ah_attr      ah_attr;            /* Primary path address vector (valid only for RC/UC QPs) */
        struct ibv_ah_attr      alt_ah_attr;        /* Alternate path address vector (valid only for RC/UC QPs) */
        uint16_t                pkey_index;         /* Primary P_Key index */
        uint16_t                alt_pkey_index;     /* Alternate P_Key index */
        uint8_t                 en_sqd_async_notify; /* Enable SQD.drained async notification - irrelevant for ibv_query_
        uint8_t                 sq_draining;        /* Is the QP draining? (Valid only if qp_state is SQD) */
        uint8_t                 max_rd_atomic;      /* Number of outstanding RDMA reads & atomic operations on the de
        uint8_t                 max_dest_rd_atomic; /* Number of responder resources for handling incoming RDMA rea
        uint8_t                 min_rnr_timer;      /* Minimum RNR NAK timer (valid only for RC QPs) */
        uint8_t                 port_num;           /* Primary port number */
        uint8_t                 timeout;            /* Local ack timeout for primary path (valid only for RC QPs) */
        uint8_t                 retry_cnt;          /* Retry count (valid only for RC QPs) */
        uint8_t                 rnr_retry;          /* RNR retry (valid only for RC QPs) */
        uint8_t                 alt_port_num;       /* Alternate port number */
```

```
        uint8_t         alt_timeout;    /* Local ack timeout for alternate path (valid only for RC QPs) */
};
```

For details on struct ibv_qp_cap see the description of **ibv_create_qp()**. For details on struct ibv_ah_attr see the description of **ibv_create_ah()**.

RETURN VALUE

ibv_query_qp() returns 0 on success, or the value of errno on failure (which indicates the failure reason).

NOTES

The argument *attr_mask* is a hint that specifies the minimum list of attributes to retrieve. Some RDMA devices may return extra attributes not requested, for example if the value can be returned cheaply. This has the same form as in **ibv_modify_qp()**.

Attribute values are valid if they have been set using **ibv_modify_qp()**. The exact list of valid attributes depends on the QP state.

Multiple calls to **ibv_query_qp()** may yield some differences in the values returned for the following attributes: qp_state, path_mig_state, sq_draining, ah_attr (if APM is enabled).

SEE ALSO

ibv_create_qp(3), **ibv_destroy_qp**(3), **ibv_modify_qp**(3), **ibv_create_ah**(3)

AUTHORS

Dotan Barak <dotanba@gmail.com>