# HEXABUS HACKERS' CHEAT SHEET

```
// Packet types
#define HXB_PTYPE_ERROR       0x00
#define HXB_PTYPE_INFO        0x01
#define HXB_PTYPE_QUERY       0x02
#define HXB_PTYPE_WRITE       0x04
#define HXB_PTYPE_EPINFO      0x09
#define HXB_PTYPE_EPQUERY     0x0A
// Data types
#define HXB_DTYPE_UNDEFINED   0x00
#define HXB_DTYPE_BOOL        0x01
#define HXB_DTYPE_UINT8       0x02
#define HXB_DTYPE_UINT32      0x03
#define HXB_DTYPE_DATETIME    0x04
#define HXB_DTYPE_FLOAT       0x05
#define HXB_DTYPE_128STRING   0x06
#define HXB_DTYPE_TIMESTAMP   0x07
// Error codes
//     0x00     reserved: No error
#define HXB_ERR_UNKNOWNEID    0x01
#define HXB_ERR_WRITEREADONLY 0x02
#define HXB_ERR_CRCFAILED     0x03
#define HXB_ERR_DATATYPE      0x04
// Operators for comparison in state machine
#define STM_EQ                0x00
#define STM_LEQ               0x01
#define STM_GEQ               0x02
#define STM_LT                0x03
#define STM_GT                0x04
#define STM_NEQ               0x05

struct datetime {
    uint8_t   hour;
    uint8_t   minute;
    uint8_t   second;
    uint8_t   day;
    uint8_t   month;
    uint16_t  year;
    uint8_t   weekday;  // sun=0, mon=1,...
} __attribute__ ((packed));

struct condition {
  uint8_t  sourceIP[16];
  uint32_t sourceEID;
  uint8_t  op;
  uint8_t  datatype;
  char     data[4];
} __attribute((packed));
```

```
struct transition {
  uint8_t  fromState;
  uint8_t  cond;
  uint32_t eid;
  uint8_t  goodState;
  uint8_t  badState;
  struct   hxb_value value; }

// Just the packet header.
struct hxb_packet_header {
  char      header[4];  // HX0B
  uint8_t   type;       // Packet type
  uint8_t   flags;      // Flags
  uint32_t  eid;        // EID / Error code
  uint8_t   datatype;
} __attribute__ ((packed));

// ERROR packet
struct hxb_packet_error {
  char      header[4];
  uint8_t   type;
  uint8_t   flags;
  uint8_t   errorcode;
  uint16_t  crc;
} __attribute__ ((packed));

// QUERY packet
struct hxb_packet_query {
  char      header[4];
  uint8_t   type;
  uint8_t   flags;
  uint32_t  eid;        // Endpoint ID
  uint16_t  crc;       // CRC16-Kermit
} __attribute__ ((packed));

// WRITE/INFO packet for BOOL and INT8
struct hxb_packet_int8 {
  char      header[4];
  uint8_t   type;
  uint8_t   flags;
  uint32_t  eid;
  uint8_t   datatype;
  uint8_t   value;
  uint16_t  crc;
} __attribute__ ((packed));
```

```
// WRITE/INFO packet for INT32
struct hxb_packet_int32 {
  char      header[4];
  uint8_t   type;
  uint8_t   flags;
  uint32_t  eid;
  uint8_t   datatype;
  uint32_t  value;
  uint16_t  crc;
} __attribute__ ((packed));

// DATE/TIME
struct hxb_packet_datetime {
    char      header[4];
    uint8_t   type;
    uint8_t   flags;
    uint32_t  eid;
    uint8_t   datatype;
    struct datetime  value;
    uint16_t  crc;
} __attribute__ ((packed));

// WRITE/INFO packet for FLOAT

struct hxb_packet_float {
  char      header[4];
  uint8_t   type;
  uint8_t   flags;
  uint32_t  eid;
  uint8_t   datatype;
  float     value;
  uint16_t  crc;
} __attribute__ ((packed));

// WRITE/INFO or EPINFO packet
struct hxb_packet_128string {
  char      header[4];
  uint8_t   type;
  uint8_t   flags;
  uint32_t  eid;
  uint8_t   datatype;     // this is set to
the datatype of the endpoint if it's an
EPINFO packet!
  char      value[128];
  uint16_t  crc;
} __attribute__ ((packed));
```