Rabbit Ethernet/serial Downloader Protocol Description

Revised: 7-7-2004

1.0 COMMUNICATION STRUCTURE

The downloader is based on a data structure:
(below is defined in udpdnld.h)

```
#pragma pack(1)
typedef struct {
#ifdef SERIAL_LOADER
	ubyte flag
#endif
	ubyte cmd,status;
	ushort length;  // data size
	ulong address; // write address
	ubyte data[ETH_MTU-8];  // can be up to 512 bytes.
} UDP_DBG_HEADER;
#pragma pack()

#define UDP_DBG_HEADER_SIZE (sizeof(UDP_DBG_HEADER)-ETH_MTU+8)
```

The 'cmd' field has the following commands:

```
// command definitions
#define UDPDNLD_QUERY 1         // query for all boards (broadcast)
#define UDPDNLD_DOWNLOAD_RAM 2 // download RAM code
#define UDPDNLD_XMEM_SIZE 3            // xmem size needed for RAM code
#define UDPDNLD_DOWNLOAD_FLASH 4         // download FLASH code (data len=0 when
done)
#define UDPDNLD_RUN 5             // run download RAM code
#define UDPDNLD_REBOOT 6       // restart with new code
#define UDPDNLD_SET_IP 7         // set the board to diff IP
#define UDPDNLD_SET_SPECIFIC_IP 8   // set the board to diff IP
#define UDPDNLD_GET_USERBLOCK 9
#define UDPDNLD_NULL 127        // do nothing command to resolve ARP
```

The status field is used to return acknowledges.

```
// query status
#define UDPDNLD_STATUS_NULL 0 // not set
#define UDPDNLD_STATUS_NEED_CODE 1 // board needs code downloaded to RAM
#define UDPDNLD_STATUS_HAS_CODE 2 // board has code (not used)
#define UDPDNLD_STATUS_RAM_CODE 3        // ram code already running
#define UDPDNLD_STATUS_NOMEM 4 // no xmem available
#define UDPDNLD_STATUS_REBOOT 5 // rebooting
```

```
#define UDPDNLD_STATUS_ACK 6 // download acknowledge
#define UDPDNLD_STATUS_NACK 7 // download fail acknowledge
#define UDPDNLD_STATUS_SEQUENCE 8 // dld sequence error
#define UDPDNLD_STATUS_RAM_CODE_IN_XMEM 9 // RAM loader in flash
#define UDPDNLD_STATUS_FRAG 10      // fragmented packet received
```

All UDP packets are based on this header. Note, the structure must not have any alignment padding (#pragma pack(1) used to disable alignment packing.)

Serial Loader Notes:

The serial loader uses the same protocol with the "flag" field added. The flag field is set to 0x7e. This synchronizes packets as serial is a data stream.

The serial uses CRC's to check packets. The routine is as follows:

```
unsigned short (unsigned char * pcdataptr,unsigned short uslength,unsigned short uscrc)
{
        unsigned short usdata;
        if (uslength==0) return(0);
        while(uslength--)
        {
                usdata=(*(pcdataptr++))<<8;
                uscrc=((uscrc&0xff)<<8)+(uscrc>>8);
                uscrc=uscrc^usdata;
                usdata=uscrc& 0xff00;
                usdata<<=4;
                uscrc=uscrc^usdata;
                usdata = uscrc & 0xff00;
                usdata >>= 5;
                uscrc = uscrc ^ usdata;
                usdata >>= 7;
        }
        uscrc = ~(((uscrc & 0x00ff) <<8)+ (uscrc >> 8 ));
        return(uscrc);
}
```

The serial loader adds two bytes to the data area for the CRC.

Packets are read by first reading flag byte and waiting for a 0x7e. Then SER_COMM_HEADER_SIZE-1 bytes are read starting at the cmd byte. Then the number of bytes specified in the length field are read along with 2 additional bytes for the crc.

Packets are checked as follows:

```
unsigned short * crcpos;
```

```
crcpos=(unsigned short *)(pkt->data+pkt->length);
if (CRC16((unsigned char *)pkt,SER_COMM_HEADER_SIZE+pkt->length,0xffff)!=*crcpos)
        {
                ... error ....
        }
```

CRC's are generated as follows:

```
unsigned short * crcpos;
crcpos=(unsigned short *)(pkt->data+pkt->length);
*crcpos=CRC16((unsigned char*)pkt,SER_COMM_HEADER_SIZE+pkt->length,0xffff);
```

The structure is transmitted as a block starting with the beginning of the structure with a length of SER_COMM_HEADER_SIZE+pkt->length+2.

2.0 Download Sequence

The sequence is basically follows the following sections in order.

Note: lib and RAM loader always reply with packet with the same cmd field as was sent.

2.1 Search for Boards

(start with search command from utility)

1. PC sends broadcast packet with cmd=UDPDNLD_QUERY.
    length=address=0;
2. Lib code replies with:
    status=UDPDNLD_STATUS_NEED_CODE (RAM loader not in flash.)
    status=UDPDNLD_STATUS_RAM_CODE_IN_XMEM (RAM loader in flash)
    data=idstring passed to UDPDL_Init()
    length=length of string;
    address=MTU size (low 16-bits) and flash block size (upper 16 bits.)
3. PC utility builds list of boards

2.2 Download and run RAM loader

(Download command hit in utility.)

If status returned was UDPDNLD_STATUS_NEED_CODE:

1. Send packet with cmd=UDPDNLD_XMEM_SIZE, length=amount of RAM needed
2. Lib allocates 'length' bytes of xmem and replies with:
    status=UDPDNLD_STATUS_NOMEM if xalloc fails
    status=UDPDNLD_STATUS_ACK if success
3. PC utility sends RAM loader file:
    cmd=UDPDNLD_DOWNLOAD_RAM
    length=#of bytes in data field (data size is limited by MTU setting

data=data
4. Lib copies data into xmem buffer.
5. Each packet is replied with status=UDPDNLD_STATUS_ACK

If status returned was UDPDNLD_STATUS_RAM_CODE_IN_XMEM or RAM loader complete:
1. PC sends packet with cmd=UDPDNLD_RUN
2. Lib copies loader from xmem buffer to root mem, saves the IP and reboots

2.3 Wait for RAM loader to boot up

Ram loader running, PC waiting for it to boot up:
1. PC utility sends packet with cmd=UDPDNLD_QUERY to the IP of the board.
2. Ram loader replies with:
    status=UDPDNLD_STATUS_RAM_CODE
    data="Ram loader" string
    length=length of string;
    address=MTU size (low 16-bits) and flash block size (upper 16 bits.)
3. PC waits for response, if no reply retries.

2.4 Check for UserBlock

Note: this is only done for files over 256k.

Version 2.1a and later will ask the board for the start of the user block address. It sends a packet with cmd=UDPDNLD_GET_USERBLOCK

The board will reply with the start address of the user block in the address field and an UDPDNLD_STATUS_ACK status.

If there is no reply from the board, the utility will try to locate the user/idblock area by scanning the file from 0x3ffff backwards.

2.5 Download .bin File

Ram loader ready, do download:
1. PC reads .bin file (and decrypts it if needed.)
2. If step 2.4 failes, the bin image is searched backwards from 0x40000 for '00's. This is where the user and ID block is stored on 256k-based boards with 512k flash. This area is marked to be skipped. If this is not done, the ID block will be overwritten.
3. PC utility sends data in blocks limited by the FLASH block size and the MTU; usually 1024 byte blocks.
4. Starting from 0 and skipping ID area the .bin file is sent in blocks with
    cmd=UDPDNLD_DOWNLOAD_FLASH
    length=number of bytes to load
    address=address of flash (offset into .bin file.)
    data=.bin file data

5. RAM loader queues of blocks until one sector is full.
6. If sector is filled it is written to flash.
7. Ram loader replies with:
    status=UDPDNLD_STATUS_ACK - success
    status=UDPDNLD_STATUS_NACK - flash write error
8. Steps 4-7 repeated until entire file is downloaded.
9. When done, PC sends packet with cmd=UDPDNLD_REBOOT
10. Ram loader replies with status=UDPDNLD_STATUS_REBOOT and then does a watchdog reset.

3.0 Notes

1. Ram loader ignores all UDP packets with CRC errors.
2. Ram loader will not handle fragmented packets. It replies with
    status=UDPDNLD_STATUS_FRAG. The PC utility will see this
    and reduce the packet size until it does not get the error.
3. The PC utility can send a broadcast packet with the cmd=UDPDNLD_SET_IP.
    The address field will have the new IP. The LIB will call the
    lib function sethostid() with the new address and reopen the
    udp port.
4. The PC utility can send a broadcast packet with the cmd=UDPDNLD_SET_SPECIFIC_IP.
    The address field will have the new IP and the data will have the MAC address.
    The LIB will call the lib function sethostid() with the new address and reopen the
    UDP port. Only boards that match the MAC address will change their IP address.
5. The PC utility will send a packet with cmd=UDPDNLD_NULL. The board will ignore this.
    This is used to try to resolve the MAC address.