DOGE Firmware Architecture Specification

# Messaging Protocols

### Data Link Layer Packets

#### Definitions

* TYPE – Packet type.
* ACK – Acknowledgement packet.
* SRC – Originating node id.
* DST – Destination node id.
* SH SRC – Single Hop source.
* SH DST – Single Hop destination.
* TTL – Time to live.
* PACKET ID – Packet ID used to track duplicate packet reception.
* RTA – Packet retransmission count.
* ADDRESS – Address for bulk data reads/writes.
* NUM PACKETS – Requested number of packets in a bulk data read/write request.
* SEQUENCE NUMBER – Sequence number of a packet in a bulk data transfer.

#### Raw Packet

##### Description

Raw packets are variable sized packets used by the application layer.

TYPE = RAW PACKET

ACK = TRUE/FALSE

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Size(bits) | 1 | 7 | 4 | 4 | 16 | 16 | 16 | 16 | 8 | 8 | 8 | Variable |
| Name | ACK | TYPE | PACKET  ID | RTA | SRC | DST | SH SRC | SH DST | TTL | CRC | SIZE | DATA |

#### Signaling Beacons

TYPE = SIGNALING BROADCAST BEACON

ACK = FALSE

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Size(bits) | 1 | 7 | 4 | 4 | 16 | 16 | 16 | 16 | 8 | 8 |
| Name | ACK | TYPE | PACKET  ID | RTA | SRC | DST | SH SRC | SH DST | TTL | CRC |

TYPE = SIGNALING UNICAST BEACON

ACK = FALSE

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Size(bits) | 1 | 7 | 4 | 4 | 16 | 16 | 16 | 16 | 8 | 8 |
| Name | ACK | TYPE | PACKET  ID | RTA | SRC | DST | SH SRC | SH DST | TTL | CRC |

TYPE = SIGNALING UNICAST BEACON

ACK = TRUE

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Size(bits) | 1 | 7 | 4 | 4 | 16 | 16 | 16 | 16 | 8 | 8 | 8 |
| Name | ACK | TYPE | PACKET  ID | RTA | SRC | DST | SH SRC | SH DST | TTL | CRC | ERROR CODE |

#### Bulk Data Read Request Packets

TYPE = BULK DATA READ REQUEST

ACK = FALSE

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Size(bits) | 1 | 7 | 4 | 4 | 16 | 16 | 16 | 16 | 8 | 8 | 16 | 8 |
| Name | ACK | TYPE | PACKET ID | RTA | SRC | DST | SH SRC | SH DST | TTL | CRC | ADDRESS | NUM  PACKETS |

TYPE = BULK DATA READ REQUEST

ACK = TRUE

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Size(bits) | 1 | 7 | 4 | 4 | 16 | 16 | 16 | 16 | 8 | 8 | 8 |
| Name | ACK | TYPE | PACKET  ID | RTA | SRC | DST | SH SRC | SH DST | TTL | CRC | ERROR CODE |

#### Bulk Data Read Data Packets

TYPE = BULK DATA READ DATA

ACK = FALSE

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Size(bits) | 1 | 7 | 4 | 4 | 16 | 16 | 16 | 16 | 8 | 8 | 10 | 160 |
| Name | ACK | TYPE | PACKET ID | RTA | SRC | DST | SH SRC | SH DST | TTL | CRC | SEQUENCE NUMBER | PAYLOAD |

TYPE = BULK DATA READ DATA

ACK = TRUE

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Size(bits) | 1 | 7 | 4 | 4 | 16 | 16 | 16 | 16 | 8 | 8 | 8 | 10 |
| Name | ACK | TYPE | PACKET ID | RTA | SRC | DST | SH SRC | SH DST | TTL | CRC | ERROR CODE | SEQUENCE NUMBER |

#### Bulk Data Write Request Packets

TYPE = BULK DATA WRITE REQUEST

ACK = FALSE

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Size(bits) | 1 | 7 | 4 | 4 | 16 | 16 | 16 | 16 | 8 | 8 | 16 | 8 |
| Name | ACK | TYPE | PACKET ID | RTA | SRC | DST | SH SRC | SH DST | TTL | CRC | ADDRESS | NUM PACKETS |

TYPE = BULK DATA WRITE REQUEST

ACK = TRUE

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Size(bits) | 1 | 7 | 4 | 4 | 16 | 16 | 16 | 16 | 8 | 8 | 8 |
| Name | ACK | TYPE | PACKET  ID | RTA | SRC | SH SRC | SH DST | DST | TTL | CRC | ERROR CODE |

#### Bulk Data Write Data Packets

TYPE = BULK DATA WRITE DATA

ACK = FALSE

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Size(bits) | 1 | 7 | 4 | 4 | 16 | 16 | 16 | 16 | 8 | 8 | 10 | 160 |
| Name | ACK | TYPE | PACKET ID | RTA | SRC | DST | SH SRC | SH DST | TTL | CRC | SEQUENCE NUMBER | PAYLOAD |

TYPE = BULK DATA WRITE DATA

ACK = TRUE

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Size(bits) | 1 | 7 | 4 | 4 | 16 | 16 | 16 | 16 | 8 | 8 | 8 | 10 |
| Name | ACK | TYPE | PACKET ID | RTA | SRC | DST | SH SRC | SH DST | TTL | CRC | ERROR CODE | SEQUENCE NUMBER |

#### Bulk Data Read/Write Abort Packet

TYPE = BULK DATA RW ABORT

ACK = FALSE

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Size(bits) | 1 | 7 | 4 | 4 | 16 | 16 | 16 | 16 | 8 | 8 |
| Name | ACK | TYPE | PACKET  ID | RTA | SRC | DST | SH SRC | SH DST | TTL | CRC |

TYPE = BULK DATA RW ABORT

ACK = TRUE

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Size(bits) | 1 | 7 | 4 | 4 | 16 | 16 | 16 | 16 | 8 | 8 | 8 |
| Name | ACK | TYPE | PACKET  ID | RTA | SRC | DST | SH SRC | SH DST | TTL | CRC | ERROR CODE |

#### Link Layer Packet

##### Description

Link Layer Packets are used by the link layer for error reporting and establishing reliable channels between nodes.

TYPE = Link Layer Packet

ACK = TRUE/FALSE

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Size(bits) | 1 | 7 | 4 | 4 | 16 | 16 | 16 | 16 | 8 | 8 | 8 |
| Name | ACK | TYPE | PACKET  ID | RTA | SRC | DST | SH SRC | SH DST | TTL | CRC | ERROR CODE |

### Application Layer Packets

#### Command Packets

##### Read Register Command

CMD = READ REG

|  |  |  |
| --- | --- | --- |
| Size(bits) | 8 | 8 |
| Name | CMD | ADDRESS |

##### Read Register Responses

CMD = ACK

|  |  |  |  |
| --- | --- | --- | --- |
| Size(bits) | 8 | 8 | 8 |
| Name | CMD | ADDRESS | DATA |

CMD = NACK

ADDRESS = ERROR ADDRESS

DATA = ERROR (ERROR RANGE)

|  |  |  |  |
| --- | --- | --- | --- |
| Size(bits) | 8 | 8 | 8 |
| Name | CMD | ADDRESS | DATA |

##### Write Register Command

CMD = WRITE REG

|  |  |  |  |
| --- | --- | --- | --- |
| Size(bits) | 8 | 8 | 8 |
| Name | CMD | ADDRESS | DATA |

##### Write Register Responses

CMD = ACK

|  |  |  |  |
| --- | --- | --- | --- |
| Size(bits) | 8 | 8 | 8 |
| Name | CMD | ADDRESS | DATA |

CMD = NACK

ADDRESS = ERROR ADDRESS

DATA = ERROR (ERROR RANGE), ERROR (READ ONLY REG)

|  |  |  |  |
| --- | --- | --- | --- |
| Size(bits) | 8 | 8 | 8 |
| Name | CMD | ADDRESS | DATA |

##### No operation (NOP) Command

CMD = NOP

|  |  |
| --- | --- |
| Size(bits) | 8 |
| Name | CMD |

No Responses

##### Unknown Command

Packet contains an unimplemented command. Full application packet fields shown below.

CMD = UNKNOWN

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Size(bits) | 8 | 8 | 8 | 8 | 128 | 16 |
| Name | CMD | ADDRESS | DATA | BYTE NUMBER | PAYLOAD | RESERVED |

##### Unknown Command Responses

CMD = NACK

ADDRESS = 0

DATA = ERROR (ERROR COMMAND)

|  |  |  |  |
| --- | --- | --- | --- |
| Size(bits) | 8 | 8 | 8 |
| Name | CMD | ADDRESS | DATA |

##### Read Memory Command

CMD = READ MEM

|  |  |  |  |
| --- | --- | --- | --- |
| Size(bits) | 8 | 8 | 8 |
| Name | CMD | ADDRESS | DATA = SIZE IN BYTES |

##### Read Memory Responses

CMD = READ MEM ACK

|  |  |  |  |
| --- | --- | --- | --- |
| Size(bits) | 8 | 8 | Variable size from 8 to 96 |
| Name | CMD | ADDRESS | DATA |

CMD = NACK

ADDRESS = 0

DATA = ERROR (ERROR RANGE)

|  |  |  |  |
| --- | --- | --- | --- |
| Size(bits) | 8 | 8 | 8 |
| Name | CMD | ADDRESS | DATA |

##### Write Memory Command

CMD = WRITE MEM

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Size(bits) | 8 | 8 | 8 | Variable size from 8 to 96 |
| Name | CMD | ADDRESS | DATA = SIZE IN BYTES | DATA |

##### Write Memory Responses

CMD = WRITE MEM ACK

|  |  |  |  |
| --- | --- | --- | --- |
| Size(bits) | 8 | 8 | 8 |
| Name | CMD | ADDRESS | DATA |

CMD = NACK

ADDRESS = 0

DATA = ERROR (ERROR RANGE)

|  |  |  |  |
| --- | --- | --- | --- |
| Size(bits) | 8 | 8 | 8 |
| Name | CMD | ADDRESS | DATA |

# Networking Protocols

The DOGE networking protocols are a “Neighbor Protocol” for neighbor discovery, a MAC ALOHA protocol for data transmission on a shared wireless channel, and a “sliding window protocol” for reliable data transmission.

## Sliding Window Protocol

#### Notes

1. The sliding window protocol is used in the DOGE data link layer for reliable data transmission.
2. The window size used for the sliding windows is 1. This is due to the low amount of memory on the MSP430 that restricts the size of windows.

#### Requirements

1. The sliding window protocol shall provide a method for the network layer to send and receive reliable data transmissions.

## Neighbor Protocol

#### Notes

1. The neighbor protocol is used in the DOGE data link layer for neighbor discovery.
2. Signaling broadcast packets do not require ACKs. A node responds to a signaling broadcast packet with a unicast signaling packet.
3. *tsb* = signaling broadcast period
4. *tsr* = signaling receiving period
5. *tsleep* = node sleeping period

#### Requirements

1. When power is first applied to a sensor node, a node shall continuously broadcast signaling packets for '*tsb*' seconds (signaling broadcast period).
2. After a node's signaling broadcast period elapses, a node shall switch its radio into receiver mode for '*tsr*' seconds (signaling receiving period).
3. Upon receiving a signaling broadcast packet, a node shall:
   1. Check the signaling node's id for neighbor table inclusion.
   2. Switch into transmitter mode.
   3. Send a unicast signaling packet to the signaling broadcast node.
4. During a node's signaling receiving period, a node shall check received unicast signaling packets for neighbor table inclusion.
5. After a node's signaling receiving period elapses, a node shall count the unicast signaling packets received during the signaling receiving period to determine its next state.
   1. If the number of received unicast signaling packets is non-zero, the node shall switch into a normal receiving mode.
   2. If the number of received unicast signaling packets is zero, the node shall sleep for 'tp' seconds, and then reattempt the signaling broadcast sequence.

## MAC protocol (ALOHA)

#### Notes

1. The DOGE MAC Protocol based on ALOHA is used in the data link layer for channel access control.
2. *tr* = acknowledgement receiving period

#### Requirements

1. When a node is not in an acknowledgement receiving period and a packet is present in a node's tx packet queue, the node shall inspect the current tx packet to determine if it requires an acknowledgement receiving period.
   1. If the packet is a signaling broadcast packet, the node shall:
      1. Request that the radio driver transmit the packet.
      2. Remove the signaling broadcast packet from the packet queue.
   2. If the packet is not a signaling broadcast packet, the node shall:
      1. Assign the packet a unique id.
      2. Request that the radio driver transmit the packet.
      3. Switch the radio driver into rx mode for '*tr*' seconds (acknowledgement receiving period).
2. During an acknowledgement receiving period, a node shall continuously check the rx packet queue for an ACK packet.
   1. If an ACK is received, the node shall immediately cancel the current acknowledgement receiving period.
3. After a node's acknowledgment receiving period terminates (timeout or ACK received), a node shall check its rx packet queue to determine how to process the current tx packet.
   1. If a packet ACK is present, the node shall:
      1. Calculate a PRR (“Packet Reception Ratio”) from the ACK packet’s “packet transmission counter” and “packet received counter.”
      2. Update the neighbor table entry for this node with this ACK packet’s PRR value.
      3. Remove the current tx packet from the packet queue.
      4. Perform no further processing of the tx packet.
   2. If a packet ACK is not present, the node shall perform the following steps:
      1. If the current contention period is greater than or equal to the max contention period, the node shall:
         1. Calculate the RNP (“Required Number of Packet retransmissions”) by reading the tx packet’s RTA “packet retransmission count” value.
         2. Update the neighbor table entry for this node with this tx packet’s RNP value.
         3. Remove the current tx packet from the tx packet queue.
         4. Perform no further processing of the tx packet.
      2. If the current contention period is zero, the node shall set the current contention period to the minimum contention period (e.g. 32).
      3. If the current contention period is greater than zero and less than the max contention period, the node shall double the current contention period (exponential backoff).
      4. After assigning new contention periods, a node shall set the current backoff period to a random value between 0 and the current contention period, and then sleep for that amount of time before rechecking the tx packet queue.

## Radio Driver

#### Requirements

1. The radio driver shall provide a method to switch the RF transceiver and the radio driver between rx and tx mode.
2. When the radio driver is in rx mode, the radio driver shall check the RF transceiver for new packets.
3. Upon detecting a new RF transceiver packet, the radio driver shall check the rx packet queue to determine how it must process the new RF transceiver packet.
   1. If the packet id of the new RF transceiver packet exists within the rx packet queue, the radio driver shall increment the “packet received counter” within the corresponding packet in the rx packet queue.
   2. If the packet id of the new RF transceiver packet does not exist within the rx packet queue, the radio driver shall check the rx packet queue state to continue processing the current RF transceiver packet.
      1. If the rx packet queue is full, the radio driver shall drop the RF transceiver packet.
      2. If the rx packet queue is not full, the radio driver shall insert the RF transceiver packet into the rx packet queue.
4. The radio driver shall provide a method to transmit and retransmit a specific index within the tx packet queue.
5. The radio driver shall transmit packets only when the MAC layer has explicitly requested a switch into tx mode beforehand.
6. When the radio driver has been requested to transmit a specific index within the tx packet queue, the radio driver shall increment the “packet transmission counter” within the packet, and hand over the packet to the RF transceiver for transmission.

# Networking Rules

DOGE networking rules for routing and forwarding packets are contained within network tables stored in individual nodes.

## Network Table

A node’s network table is split between two tables of adjustable size: a neighbor table and a routing table. The neighbor table is used to forward packets to single hop neighbors, while the routing table is used to route packets to multi-hop neighbors. The entries for these two tables are described in the sections below.

## Neighbor Table Entries

A node’s neighbor table entry contains a single hop neighbor’s Node ID, Radio ID, and Network ID, as well as an estimate of the link quality to that single hop neighbor. The neighbor table fields are packed into a 32 bit structure as shown below.

|  |  |
| --- | --- |
| Field | Size(bits) |
| Single hop Node ID | 16 |
| Single hop LQE | 12 |
| Radio ID | 2 |
| Neighbor ID | 2 |

## Routing Table Entries

A node’s routing table contains a multi-hop neighbor’s Node ID, a neighbor table index pointing to the neighbor for forwarding single-hop packets, and a multi-hop Link Quality Estimation. The routing table fields are packed into a 32 bit structure as shown below.

|  |  |
| --- | --- |
| Field | Size(bits) |
| Multi hop Node ID | 16 |
| Single hop LQE | 12 |
| Radio ID | 2 |
| Neighbor ID | 2 |

## Reserved LQEs

There are 2 reserved LQE values for entries in the neighbor table.

|  |  |  |
| --- | --- | --- |
| Name | Value | Description |
| MASKED LQE | 0x00 | Blocks routing and forwarding rules from using the neighbor table entry. In other words, it “masks” the edge to the single hop neighbor. |
| PERFECT LQE | 0xFF | Forces routing and forwarding rules to use this entry for sending traffic to a single hop neighbor. The network LQE update logic will never set a perfect LQE value. This can only be done explicitly by a user. |
| MAX LQE | 0xFE | Maximum value set by the network LQE update logic. |
| MIN LQE | 0x01 | Minimum value set by the network LQE update logic. |