uavcan

BootLoaderAppCommon (APP Written)
+CAN_F1R1:Signature = {0xb0a0,0x4150}
+CAN_F1R2:bitRate = from App
+CAN_F2R1:thisNodeID = From App
+CAN_F3R1-F3R2-:crc = Over CAN_F1R1 to and including CAN_F2R2

BootLoaderAppCommon (BL Written)
+Signature = {0xb0a0,0x424C}
+bitRate = From BL Auto Baud
+thisNodeID = From BL Alloction
+crc = Over Signature to and Including thisNodeID

Defined in Application Bottloader Common header, instantiated in memory. The Data will be passed between the App and BL in the CAN Filter registers starting at CAN_F1R1
Both Object share the same Registers. The Signature changes to reflect the last writer.

Req559.DynamicNodeIDAllocation.uavcan
-int8 ALLOCATOR_INITIALIZATION_DELAY_SEC = 30
-int8 ALLOCATEE_MIN_BROADCAST_INTERVAL_SEC = 2
-int7 RECOMMENDED_DYNAMIC_NODE_ID_RANGE_MIN = 64
-int7 RECOMMENDED_DYNAMIC_NODE_ID_RANGE_MAX = 125
-truncated uint57 short_unique_id
-truncated uint7 node_id

Rsp559.DynamicNodeIDAllocation.uavcan
-int8 ALLOCATOR_INITIALIZATION_DELAY_SEC = 30
-int8 ALLOCATEE_MIN_BROADCAST_INTERVAL_SEC = 2
-int7 RECOMMENDED_DYNAMIC_NODE_ID_RANGE_MIN = 64
-int7 RECOMMENDED_DYNAMIC_NODE_ID_RANGE_MAX = 125
-truncated uint57 short_unique_id
-truncated uint7 node_id

1023.LogMessage.uavcan
+LogLevel level
+uint8[<32] source
+uint8[<128] text

SoftwareVersion.uavcan
+uint8 major
+uint8 minor
+uint8 OPTIONAL_FIELD_MASK_VCS_COMMIT = 1
+uint8 OPTIONAL_FIELD_MASK_IMAGE_CRC = 2
+uint8 optional_field_mask
+uint32 vcs_commit
+uint64 image_crc

HardwareVersion.uavcan
+uint8 major = -DOTP_HW_MAJOR
+uint8 minor = -DOTP_HW_MINOR
+uint8[16] unique_id
+uint8[<256] certificate_of_authenticity

Defined in Application with section type and Located By .ld script

AppImageDescriptor
-fwImageSize
-fwImageCRC

Req551.GetNodeInfo.uavcan

Resp551.GetNodeInfo.uavcan
+NodeStatus status
+SoftwareVersion software_version
+HardwareVersion hardware_version
+uint8[<=80] name

0.GlobalTimeSync.uavcan
-uint16 MAX_PUBLICATION_PERIOD_MS = 1100 # Milliseconds
uint16 MIN_PUBLICATION_PERIOD_MS = 350 # Milliseconds
uint16 PUBLISHER_TIMEOUT_MS = 10000   # Milliseconds
uint64 previous_transmission_timestamp_usec # Microseconds

550.NodeStatus.uavcan
+int16 MAX_PUBLICATION_PERIOD_MS = 1000
+int16 MIN_PUBLICATION_PERIOD_MS = 2
+int16 OFFLINE_TIMEOUT_MS = 3000
+int28 uptime_sec
+int4 STATUS_OK = 0
+int4 STATUS_INITIALIZING = 1
+int4 STATUS_WARNING = 2
+int4 STATUS_CRITICAL = 3
+int4 STATUS_OFFLINE = 15
+int4 status_code
+int16 vendor_specific_status_code

Req580.BeginFirmwareUpdate.uavcan
+uint8 source_node_id
+Path image_file_remote_path

Resp580.BeginFirmwareUpdate.uavcan
+uint8 ERROR_OK = 0
+uint8 ERROR_INVALID_MODE = 1
+uint8 ERROR_IN_PROGRESS = 2
+uint8 ERROR_UNKNOWN = 255
+uint8 error = [BootLoadInProcess] ? ERROR_IN_PROGRESS : ERROR_OK

```
+uint8 ERROR_UNKNOWN = 255
+uint8 error = [BootLoadInProcess] ? ERROR_IN_PROGRESS  : ERROR_OK
+uint8[<128] optional_error_message
```

Req588.Read.uavcan
```
+uint32 offset
+Path path
```

Resp588.Read.uavcan
```
+Error error
+uint8[<=250] data
```

file.Error
```
+int16 OK = 0
+int16 UNKNOWN_ERROR = 32767
+int16 NOT_FOUND = 2
+int16 IO_ERROR = 5
+int16 ACCESS_DENIED = 13
+int16 IS_DIRECTORY = 21
+int16 INVALID_VALUE = 22
+int16 FILE_TOO_LARGE = 27
+int16 OUT_OF_SPACE = 28
+int16 value
```

file.Path
```
+uint8 SEPARATOR = '/' uint8[<=200] path
-uint8[<=200] path
```

Req585.GetInfo.uavcan
```
+Path path
```

Resp585.GetInfo.uavcan
```
+uint64 crc64
+uint32 size
-Error error
+EntryType entry_type
```

PX4Fmu

CAN Node Esc
0

Local UI

Start up script

BootLoaderAppCommon (APP Written)
+CAN_F1R1:Signature = (0xb0a0,0x4150)
+CAN_F1R2:bitRate = from App
+CAN_F2R1:thisNodeID = From App
+CAN_F2R2:reserved = 0
+CAN_F3R1-F3R2-:crc = Over CAN_F1R1 to and including CAN_F2R2

BootLoaderAppCommon (BL Written)
+Signature = (0xb0a0,0x424C)
+bitRate = From BL Auto Baud
+thisNodeID = From BL Alloction
+reserved
+crc = Over Signature to and Including thisNodeID

If the reboot is from Application in response to
Resp580.BeginFirmwareUpdate.uavcan
Then in Filter Registers starting at CAN_F1R1 will be be used to
commnicate the data for the BootLoaderAppCommon struct.

BootLoaderAppCommon would be written by the application with a
Signature of
Signature[0] = 0xb0a0, Signature[1] = 0x4150 (AP)
If the CRC is valid then we skip to RestartFromApp

3: Reset

1: Reset

2: Clock, IO Init

4: [-DOPT_ENABLE_WD] EnableWatchDog(/256 Mode 26.2144 Sec)

Kicking the WD not Shown

5: uavcan start

6: INDICATE_RESET

Pull together Bits and pieces to Init NodeInfo with
hardware info and if AppValid init Software info from
AppImageDescriptor

7: bool AppValid = (AppFlash[0] != 0xffffffff) && ComputeAppCRC(APP_LOADADDR, APP_INFO_OFFSET)

8: InitNodeInfo(AppValid)

AppImageDescriptor
-fwImageSize
-fwImageCRC

APP_LOADADDR is the Size of the Bootloader rounded to Flash
Erase Sector/Page Size ,
APP_INFO_OFFSET added to APP_LOADADDR at which the
AppImageDescriptor is locatiged in the Application Images

BootLoaderAppCommon (BL Written) Init
+Signature = (0xb0a0,0x424C)
+bitRate = [!AppBLRequest] ? 0 : bitRate from App
+thisNodeID = [!AppBLRequest] ? 0 : nodeId from App
+reserved = 0
+crc = [AppBLRequest] ? crc64(this,sizeof(*this)) : 0

9: bool AppBLRequest = Validate(BootLoaderAppCommon, AP)

init will not update
bitRate and thisNodeID if
AppBLRequest

10: Init(BootLoaderAppCommon, BL, AppBLRequest)

11: bool WaitForGetNodeInfo = -DOPT_WAIT_FOR_GET_NODE_INFO

Compile Time Option
Override with Jumper

12: [-DOPT_WAIT_FOR_GET_NODE_INFO_JUMPER_GPIO] : WaitForGetNodeInfo = IORead(-DOPT_WAIT_FOR_GET_NODE_INFO_JUMPER_GPIO)

13: [!WaitForGetNodeInfo && AppValid]: StartTboot([-DOPT_TBOOT)

14: [AppBLRequest]:RestartFromApp

15: [!AppBLRequest]:INDICATE_AUTOBAUDING

16: [!AppBLRequest]:Auto Baud

18: 550.NodeStatus.uavcan

17: [!AppBLRequest]:detect bitrate

Node will AutoBaud on any Message

550.NodeStatus.uavcan
+int16 MAX_PUBLICATION_PERIOD_MS = 1000
+int16 MIN_PUBLICATION_PERIOD_MS = 2
+int16 OFFLINE_TIMEOUT_MS = 3000
+int28 uptime_sec
+int4 STATUS_OK = 0
+int4 STATUS_INITIALIZING = 1
+int4 STATUS_WARNING = 2
+int4 STATUS_CRITICAL = 3
+int4 STATUS_OFFLINE = 15
+int4 status_code
+int16 vendor_specific_status_code

18.1: [!AppBLRequest]:SetBitRate(newBitRate)

18.2: [!AppBLRequest]:Update(BootLoaderAppCommon, BL)

18.3: [!AppBLRequest]:INDICATE_AUTOBAUDED

BootLoaderAppCommon (BL Written) SetBitRate
+Signature = (0xb0a0,0x424C)
+bitRate = newBitRate from Auto Baud
+thisNodeID = 0
-reserved = 0
+crc = crc64(this,sizeof(*this)

18.4: [Timeout(Tboot)]: jump_to_app (node_id is 0 => App need to Allocate or use Static NodeID

18.5: [!AppBLRequest]:INDICATE_ALLOCATION

18.6: GetUUID()

18.7: Req559.DynamicNodeIDAllocation.uavcan(uuid,0)

Req559.DynamicNodeIDAllocation.uavcan
-int8 ALLOCATOR_INITIALIZATION_DELAY_SEC = 30
-int8 ALLOCATEE_MIN_BROADCAST_INTERVAL_SEC = 2
-int7 RECOMMENDED_DYNAMIC_NODE_ID_RANGE_MIN = 64
-int7 RECOMMENDED_DYNAMIC_NODE_ID_RANGE_MAX = 125
-truncated uint57 short_unique_id
-truncated uint7 node_id

This Is repeated indefinitely Rate Limited at 0.5
message per Second and will Exit To Application if
Timeout(Tboot) occurs
A Timeout(Tboot) will Never occur if !AppValid

19: Resp559.DynamicNodeIDAllocation.uavcan(uuid,NodeID=1xx)

19.1: SetNodeID(newNodeID), Update(BootLoaderAppCommon, BL)

19.2: INDICATE_ALLOCATED

Rsp559.DynamicNodeIDAllocation.uavcan
-int8 ALLOCATOR_INITIALIZATION_DELAY_SEC = 30
-int8 ALLOCATEE_MIN_BROADCAST_INTERVAL_SEC = 2
-int7 RECOMMENDED_DYNAMIC_NODE_ID_RANGE_MIN = 64
-int7 RECOMMENDED_DYNAMIC_NODE_ID_RANGE_MAX = 125
-truncated uint57 short_unique_id
-truncated uint7 node_id

BootLoaderAppCommon (BL Written) SetNodeID
+Signature = (0xb0a0,0x424C)
+bitRate = newBitRate from Auto Baud
+thisNodeID = newNodeID from Allocation
-reserved = 0
+crc = crc64(this,sizeof(*this)
-attribute

20: [Timeout(Tboot)]: jump_to_app (node_id is 0 => App need to Allocate or use Static

21: RestartFromApp

21.1: initSeconds(0)

21.2: 550.NodeStatus.uavcan(uptime=0, STATUS_INITIALIZING)

21.2.1: Req551.GetNodeInfo.uavcan()

Req551.GetNodeInfo.uavcan

21.2.1.1: [(AppBLRequest || WaitForGetNodeInfo) && AppValid]: StartTboot([-DOPT_TBOOT)

21.2.1.2: [AppValid]:SetAppVersion(SwVerion)

21.2.1.3: 551.GetNodeInfo.uavcan

21.2.1.3: 551.GetNodeInfo.uavcan

21.2.1.2: [AppValid]:SetAppVersion(SwVerion)

Resp551.GetNodeInfo.uavcan
+NodeStatus status
+SoftwareVersion software_version
+HardwareVersion hardware_version
+uint8[<=80] name

21.2.1.4: [Timeout(Tboot)]: jump_to_app

550.NodeStatus.uavcan2
+int28 uptime_sec
+int4 STATUS_INITIALIZING = 1
+int4 status_code
+int16 vendor_specific_status_code

We wait until we get a 551.GetNodeInfo.uavcan Request or Timeout(Tboot), In the Case of Invalid App TBoot is not running we would be stuck here indefinite but since we are sending Node Status messages, when a master shows up we will get a 551.GetNodeInfo.uavcan Request and continue

SoftwareVersion.uavcan
+uint8 major
+uint8 minor
+uint8 OPTIONAL_FIELD_MASK_VCS_COMMIT = 1
+uint8 OPTIONAL_FIELD_MASK_IMAGE_CRC = 2
+uint8 optional_field_mask
+uint32 vcs_commit
+uint64 image_crc

HardwareVersion.uavcan
+uint8 major = -DOTP_HW_MAJOR
+uint8 minor = -DOTP_HW_MINOR
+uint8[16] unique_id
+uint8[<256] certificate_of_authenticity

22: Req580.BeginFirmwareUpdate.uavcan

Req580.BeginFirmwareUpdate.uavcan
+uint8 source_node_id
+Path image_file_remote_path

22.1: BootLoadInProcess = true

22.2: Resp580.BeginFirmwareUpdate.uavcan

Resp580.BeginFirmwareUpdate.uavcan
+uint8 ERROR_OK = 0
+uint8 ERROR_INVALID_MODE = 1
+uint8 ERROR_UNKNOWN = 255
+uint8 error = [BootLoadInProcess] ? ERROR_IN_PROGRESS : ERROR_OK
+uint8[<128] optional_error_message

22.3: fwPath = image_file_remote_path

22.4: fwSourceNodeID = source_node_id

23: INDICATE_FW_UPDATE_START

24: 550.NodeStatus.uavcan(uptime=t, STATUS_INITIALIZING)

25: SetRetryAndTimeout(3,1000MS)

26: 585.GetInfo.uavcan(path)

This Is repeated 3 times Rate Limited at 1 message per Second

Any Failure should send a log message.

Just a single digit could be used to save space on the node. The Host could translate this to Full length text

Req585.GetInfo.uavcan
+Path path

size is <= -DOPT_APP_SIZE or Id file entry_type is a file errors == ERROR_OK

28: 585.GetInfo.uavcan(fw_path,fw_crc,fw_size...),

27: validateFileInfo(file_info, &errorcode)

Resp585.GetInfo.uavcan
+uint64 crc64
+uint32 size
-Error error
+EntryType entry_type

28.1: [Retries==0]:InvalidateBootLoaderAppCommon(),RestartWithDelay(20,000ms)

28.2: [!validateFileInfo(file_info)]:1023.LogMessage.uavcan (errorcode)

1023.LogMessage.uavcan
+LogLevel level
+uint8[<32] source
+uint8[<128] text

28.3: [!validateFileInfo(file_info)]:550.NodeStatus.uavcan(uptime=t, STATUS_CRITICAL)

28.4: save(file_info)

28.5: 550.NodeStatus.uavcan(uptime=t, STATUS_INITIALIZING)

NodeStatus Messages Must continue to be sent during the whole boot loading process including long operations: Erase & Programing

28.6: 1023.LogMessage.uavcan("Erase")

28.7: EraseFlash()   28.8: [Erase Failed]:INDICATE_FW_UPDATE_ERASE_FAIL

28.9: [Erase Fail]:1023.LogMessage.uavcan

1023.LogMessage.uavcan
+LogLevel level
+uint8[<32] source
+uint8[<128] text

28.10: [Erase Fail]:550.NodeStatus.uavcan(uptime=t,STATUS_CRITICAL)

This operation is repeated to fw_size/data times but must be Rate Limited to 5 Requests per Second

28.12: 588.Read.uavcan(0, path)

28.11: SetRetryAndTimeout(3,1000MS)

Per the Uavcan spec: Timeout is 1 Sec for each message with 3 retries

Req588.Read.uavcan
+uint32 offset
+Path path

28.12.1: 588.Read.uavcan(0,path,data)

29: Program(fw_size)

Resp588.Read.uavcan
+Error error
+uint8[<=250] data

30: SaveWord0

Save Away AppFlash[0] as word0 for use in crc and final write

31: [Program Fail]::1023.LogMessage.uavcan

1023.LogMessage.uavcan
+LogLevel level
+uint8[<32] source
+uint8[<128] text

32: [Program Fail]:550.NodeStatus.uavcan(uptime=t, STATUS_CRITICAL)

error == ERROR_OK
length data != 0 && Written < file_info.size
length data == 0 && Written == file_info.size

33: 588.Read.uavcan(N,path)

33.1: 583.Read.uavcan(0,path,data)

34: ValidateReadResp(resp)

36: [(retries != 0 && timeout) || !ValidReadResponse]:1023.LogMessage.uavcan

35: [(retries != 0 && timeout) || !ValidReadResponse]:INDICATE_FW_UPDATE_INVALID_RESPONSE

38: [(retries == 0 && timeout)]:1023.LogMessage.uavcan

37: [(retries == 0 && timeout)]:INDICATE_FW_UPDATE_TIMEOUT

39: [(retries == 0 && timeout)]:550.NodeStatus.uavcan(uptime=t, STATUS_CRITICAL)

40: [Retries==0]:InvalidateBootLoaderAppCommon(),   RestartWithDelay(20,000ms)

42: [crc Fail]:1023.LogMessage.uavcan

41: CalulateCRC(file_info)

1023.LogMessage.uavcan
+LogLevel level
+uint8[<32] source
+uint8[<128] text

+uint8[<32] source
+uint8[<128] text

43: [crc Fail]:INDICATE_FW_UPDATE_INVALID_CRC

44: [crc Fail]:550.NodeStatus.uavcan(uptime=t, STATUS_CRITICAL)

45: [crc Fail]:InvalidateBootLoaderAppCommon(),   RestartWithDelay(20,000ms)

46: [crc == fw_crc]:write word0

47: ValidateBootLoaderAppCommon()

48: KickTheDog()

49: 550.NodeStatus.uavcan(uptime=t, STATUS_INITIALIZING)

50: jump_to_app