CAN Message Format

This document describes the CAN message format used in Barrett products.

Data Link Specifications

1Mbaud CANbus
8 time quanta per bit
75% sampling point
Sync jump width = 1 time quanta (TQ)
11-bit MsgID (standard CAN)
Proprietary protocol, not compatible with DeviceNet or CANopen
Recommended reading: Controller Area Network by Konrad Etschberger

CANbus Timing

75µS to ask for position
75µS per puck to respond with the positions
Control-side processing time on PC
125µS to send a packed torque to the lower 4DOF
125µS to send a packed torque to the wrist

For the 4-DOF WAM, it is: 75+(4*75)+PC+125 = 500µS + PC
For the 7-DOF WAM, it is: 75+(7*75)+PC+(2*125) = 850µS + PC

These numbers are limited by the 1 Mbps CANbus. Each message has a 47-bit frame (47µS), plus payload data (3 bytes, 24µS typ).

CANbus transceivers are not rated above 1 Mbps due to slew-rate limitations.

Raw CAN frame

```
SOF   MSGID   RTR S/E R  DLC    D0           CRC      DLM ACK DLM   EOF   INT
 0 10000000100 0   0  0 0001 00001001 xxxxxxxxxxxxxxx 1   x   1  1111111 111

/*
   0 = Dominant
   1 = Recessive
                                                                                   NOTE
   SOF = Start of Frame                                                     Always zero
   MSGID = 11-bit Message Identifier                                      User-supplied
   RTR = Remote Transmit Request                                            Always zero
   S/E = Select standard(0)/extended(1) frame                               Always zero
   R   = Reserved bit                                                       Always zero
   DLC = Data Length Code                     Valid values: 0-8 (bytes of data payload)
   D0  = Data payload, first byte              Payload can be from zero to 8 bytes long
   CRC = 15-bit Cyclic Redundancy Check field             Calculated by hardware/driver
   DLM = Delimiter                                 1-bit recessive, handled by hardware
   ACK = Acknowledgement slot                    0 = ACK, 1 = NACK, handled by hardware
   EOF = 7-bit End of Frame                                  Handled by hardware/driver
   INT = 3-bit Intermission field                            Handled by hardware/driver
   ---
   47-bit minimum frame size (plus 0-8 8-bit bytes of data payload)
*/
```

```
Barrett MSGID
-------------
GRP  FROM   TO
 G  FFFFF TTTTT
 1  00000 00100

/*
   G = Group flag. If '1', then interpret 'To' as GroupID
   F = 5-bit 'From' address
   T = 5-bit 'To' address (or GroupID)
   ---
   The example above is interpreted as, “From node ID 0, to GroupID 4”
*/

/* CAN GroupID defaults:
        0 = All pucks (except safety puck)
        1 = 4DOF torques, packed (nodes 1-4)
        2 = Wrist torques, packed (nodes 5-7)
        3 = Position feedback (P), packed
        4 = Whole WAM (nodes 1-7)
        5 = Whole BHand (nodes 11-14)
        6 = Property feedback (non-position)
        7 = Secondary encoder feedback (JP), packed
        8 = Tactile Top10 data
        9 = Tactile Full data
       10 = F/T Sensor, force data
       11 = F/T Sensor, torque data
*/
```

Every CAN node has 4 mailboxes. The primary mailbox always receives messages that are addressed directly to that node's ID. For example, if my ID is 3 and I hear a message with this MSGID on the CAN bus: “0 00000 00011”, I will receive and process that message. This is a non-group message, from ID 0 (control PC), to ID 3 (me). The other 3 mailboxes can be configured to receive group broadcast messages. Their associated configuration properties are named GRPA, GRPB, and GRPC.

```
Example configuration for WAM puck ID 3:
MBX
 1 Primary, receive messages directed to my ID (=3)
 2 GRPA = 0, receive messages broadcast to GroupID 0 (all WAM pucks)
 3 GRPB = 1, receive messages broadcast to GroupID 1 (receive 4DOF packed data)
 4 GRPC = 4, receive messages broadcast to GroupID 4 (receive WAM broadcast properties)

Example configuration for WAM Safety Board (ID 10):
MBX
 1 Primary, receive messages directed to my ID (=10)
 2 GRPA = 1, receive messages broadcast to GroupID 1 (receive 4DOF packed torques)
 3 GRPB = 2, receive messages broadcast to GroupID 2 (receive Wrist packed torques)
 4 GRPC = 3, receive messages broadcast to GroupID 3 (receive packed positions)
```

Barrett Data Payload

```
--------------------
CAN specifies a maximum of 8 bytes/frame payload - our typical payload consists of 4-6 bytes:
[RPPPPPPP] [00000000] [LLLLLLLL] [mmmmmmmm] [MMMMMMMM] [HHHHHHHH]
R: Request, 0 = Get property, 1 = Set property
P: Property (128 possible values, 0..127, 0000000..1111111)
   For a list of Properties, see:
   http://web.barrett.com/support/Puck_Documentation/PuckProperties.pdf
   http://web.barrett.com/support/ForceTorque_Documentation/ForceTorqueProperties.doc
0: Second byte (almost) always set to zero (see exceptions below)
L: Low byte of data value
m: mid-low byte of data value

If sending a 32-bit integer value, the following are used:
M: Mid-high byte of data value
H: High byte of data value
```

Exceptions

```
----------
                                  PACKED POSITION FEEDBACK

22-bit packed position, only sent from pucks in response to a Get Position (P/JP) command:
    MSGID    DLC    D0         D1         D2
10011000011 0011 [10MMMMMM] [mmmmmmmm] [LLLLLLLL]
This example is from puck ID 6, to GroupID 3.

Dual 22-bit packed positions, only sent from pucks having both a motor and a secondary
encoder, in response to a Get Position (P):
    MSGID    DLC     D0         D1         D2         D3         D4         D5
10011000011 0011 [10MMMMMM] [mmmmmmmm] [LLLLLLLL] [10MMMMMM] [mmmmmmmm] [LLLLLLLL]
This example is from puck ID 6, to GroupID 3. First 3 bytes are P, last three are JP.

4 * 14-bit packed values:
    D0         D1         D2         D3         D4         D5         D6         D7
[0x80 | prop] [AAAAAAaa] [aaaaaaBB] [BBBBbbbb] [bbbbCCCC] [Cccccccc] [ccDDDDDD] [dddddddd]
Note: Each puck’s PIDX property governs which of the 4 packed values is used. PIDX = [1..4]
Note: Must be sent from host (from ID = 0)

                                FORCE/TORQUE SENSOR FEEDBACK

Reading property 'FT' from the F/T sensor will generate 2 CAN frames:
    MSGID    DLC     D0         D1         D2         D3         D4         D5
10100001010 0110 [aaaaaaaa] [AAAAAAAA] [bbbbbbbb] [BBBBBBBB] [cccccccc] [CCCCCCCC]
10100001011 0110 [dddddddd] [DDDDdddd] [eeeeeeee] [EEEEeeee] [ffffffff] [FFFFffff]
   AAAAAAAAaaaaaaaa = 16-bit force in X, divide by 256 to get N
   BBBBBBBBbbbbbbbb = 16-bit force in Y, divide by 256 to get N
   CCCCCCCCcccccccc = 16-bit force in Z, divide by 256 to get N
   DDDDdddddddddddd = 16-bit torque about X, divide by 4096 to get Nm
   EEEEeeeeeeeeeeee = 16-bit torque about Y, divide by 4096 to get Nm
   FFFFffffffffffff = 16-bit torque about Z, divide by 4096 to get Nm
If any of the F/T sensor’s strain gages have been saturated since the last Tare command, a
7th byte will be appended to the torque frame in the format: D6 = [RBGGGGGG]
R: Re-tare suggested (always ‘1’ when this byte is present)
B: Bad data present (if ‘1’, then force and torque data should be discarded)
GGGGGG: Latched saturated gage flags.
Example: D6 = [11001001] means “The force and torque data should be discarded because either
gage 1 or gage 4 is presently saturated. Gages 1 and 4 have experienced saturation since the
last Tare command. A re-tare is suggested.”
This extra byte will be dropped from the torque frame after the next Tare command.

Reading property 'A' from the F/T sensor will generate the following CAN frame:
    MSGID    DLC     D0         D1         D2         D3         D4         D5
10100001100 0110 [aaaaaaaa] [AAAAAAAA] [bbbbbbbb] [BBBBBBBB] [cccccccc] [CCCCCCCC]
   AAAAAAAAaaaaaaaa = 16-bit acceleration in X, divide by 1024 to get m/s^2
   BBBBBBBBbbbbbbbb = 16-bit acceleration in Y, divide by 1024 to get m/s^2
   CCCCCCCCcccccccc = 16-bit acceleration in Z, divide by 1024 to get m/s^2
```

TACTILE SENSOR FEEDBACK

```
Tactile Top10 (generated after “SET TACT = 1” or “GET TACT” when TACT == 1)
   [HighSSSS] [Mid SSSS] [Low SSSS] [AAAABBBB] [CCCCDDDD] [EEEEFFFF] [GGGGHHHH] [JJJJKKKK]
   SSSS = 24-bit sensor map, exactly 10 bits will be '1', the rest '0'
   AAAA = 4-bit value of the lowest sensor ID in the map (N/cm2)
   BBBB = 4-bit value of the next lowest sensor ID in the map (N/cm2)

Top10 Example:
24.....17  16......9   8......1   AAAABBBB   CCCCDDDD   EEEEFFFF   GGGGHHHH   JJJJKKKK
[10011000] [00111010] [10000011] [01100100] [01011110] [01110111] [10110110] [10010011]
Sensors 1, 2, 8, 10, 12, 13, 14, 20, 21, and 24 are reporting the highest pressures.
The pressures are, respectively: 6, 4, 5, 14, 7, 7, 11, 6, 9, 3 (N/cm2)

Tactile Full (generated after “SET TACT = 2” or “GET TACT” when TACT == 2)
5 messages are generated in the form:
   [NNNNAAAA] [aaaaaaaa] [BBBBbbbb] [bbbbCCCC] [cccccccc] [DDDDdddd] [ddddEEEE] [eeeeeeee]
   NNNN = 4-bit sensor group: 0 = sensors 1-5, 1 = sensors 6-10, etc.
   AAAAaaaaaaaa = 12-bit sensor data from first sensor in group, divide by 256 to get N/cm2
   BBBBbbbbbbbb = 12-bit sensor data from second sensor in group, divide by 256 to get N/cm2
```

```
Example messages for a WAM
--------------------------
   MSGID     DLC    D0
                 RPPPPPPP
00000000001 0001 00000101
   0x0001     1      5
From PC, to ID 1, Len = 1, Get STAT
Read as: Hello Puck 1, this is the PC, what is your STAT?

   MSGID     DLC    D0       D1       D2       D3
                 RPPPPPPP 00000000 LLLLLLLL mmmmmmmm
10000100110 0100 10000101 00000000 00000010 00000000
   0x0426     4  0x80 | 5     0        2        0
From ID 1, to GroupID 6, Len = 4, Set STAT = 2
Non-position property feedback from a Puck is sent to GroupID 6 (see page 2).
Read as:
Hello nodes listening to Group6 messages, this is Puck 1, my STAT is 2 [STATUS_READY].

   MSGID     DLC    D0       D1       D2       D3       D4       D5
                 RPPPPPPP 00000000 LLLLLLLL mmmmmmmm MMMMMMMM HHHHHHHH
00000000001 0110 10110000 00000000 10000111 11010110 00010010 00000000
   0x0001     6  0x80 | 48    0      0x87     0xD6     0x12     0x00
From PC, to ID 1, Len = 6, Set P = 1234567

   MSGID     DLC    D0       D1       D2       D3
                 RPPPPPPP 00000000 LLLLLLLL mmmmmmmm
00000000001 0100 10001000 00000000 00000010 00000000
   0x0001     4  0x80 | 8     0        2        0
From PC, to ID 1, Len = 4, Set MODE = 2

   MSGID     DLC    D0
                 RPPPPPPP
10000000000 0001 00110000
   0x0400     1     48
From PC, to GroupID 0, Len = 1, Get motor positions

   MSGID     DLC    D0       D1       D2
                 10MMMMMM mmmmmmmm LLLLLLLL
10000100011 0011 10010010 11010110 10000111
   0x423      3    0x12     0xD6     0x87
From ID 1, to GroupID 3, Len = 3, Packed position = 1234567

   MSGID     DLC    D0       D1       D2       D3       D4       D5       D6       D7
                 RPPPPPPP AAAAAAaa aaaaaaBB BBBBbbbb bbbbCCCC Cccccccc ccDDDDDD dddddddd
10000000001 1000 10101010 00000000 01000111 11111100 11100000 00010010 11111111 10011100
From PC, to GroupID 1, Len = 8, Set 4DOF torques to [17, -50, 75, -100]
```

```
Example messages for a BarrettHand
----------------------------------
MSGID        DLC    D0            D1           D2            D3
                    RPPPPPPP      00000000     LLLLLLLL      mmmmmmmm
10000000101  0100   10011101      00000000     00001101      00000000
0x0405       4      29            0            13            0
From PC, to Group 5 (BHand, nodes 11-14), Len = 4, Set CMD (Prop #29) to 13 (CMD_HI)

  For a list of possible CMD values, see:
  http://web.barrett.com/support/Puck_Documentation/PuckCommandList.doc

MSGID        DLC    D0            D1           D2            D3
                    RPPPPPPP      00000000     LLLLLLLL      mmmmmmmm
00000001100  0100   10011101      00000000     00010010      00000000
             4      0x80 | 29     0            18            0
From PC, to Puck 12, Len = 4, Set CMD (Prop #29) to 18 (CMD_CLOSE), Close finger 2

MSGID        DLC    D0            D1           D2            D3
                    RPPPPPPP      00000000     LLLLLLLL      mmmmmmmm
00000001101  0100   10011101      00000000     00010100      00000000
             4      Ox80 | 29     0            20            0
From PC, to Puck 13, Len = 4, Set CMD (Prop #29) to 20 (CMD_OPEN), Open finger 3

MSGID        DLC    D0            D1           D2            D3
                    RPPPPPPP      00000000     LLLLLLLL      mmmmmmmm
00000001011  0100   10110100      00000000     00010000      00100111
             4      0x80 | 52     0                   10,000
From PC, to Puck 11, Len = 4, Set E (Prop #52) to 10,000
This sets the desired Endpoint position of finger 1 to 10,000 encoder cts.

MSGID        DLC    D0            D1           D2            D3
                    RPPPPPPP      00000000     LLLLLLLL      mmmmmmmm
00000001011  0100   10001000      00000000     00000101      00000000
             4      0x80 | 8      0            5             0
From PC, to Puck 11, Len = 4, Set MODE (Prop #8) to 5 (Trapezoidal Mode)
This begins a trapezoidal profile move from the present position to the specified Endpoint.

*** At this point, finger 1 will start to move. To determine when the move is complete,
*** you may poll for the MODE of Node 11 (finger 1). When the MODE is no longer 5, the move
*** is complete. The recommended polling frequency for MODE is 10Hz.
*** The following two messages represent typical communication when polling for MODE.

MSGID        DLC    D0
                    RPPPPPPP
00000001011  0001   00001000
             1      8
From PC, to Puck 11, Len = 1, Get MODE (Prop #8)

MSGID        DLC    D0            D1           D2            D3
                    RPPPPPPP      00000000     LLLLLLLL      mmmmmmmm
10101100110  0100   10001000      00000000     00000101      00000000
             4      0x80 | 8      0            5
From Puck 11, to GroupID 6, Len = 4, Return value of 5 for MODE (Prop #8)

MSGID        DLC    D0
                    RPPPPPPP
00000001100  0001   00011001
             1      25
From PC, to Puck 12, Len = 1, Get SG (Prop #25)

MSGID        DLC    D0            D1           D2            D3
                    RPPPPPPP      00000000     LLLLLLLL      mmmmmmmm
10110000110  0100   10011001      00000000     11000101      00001001
             4      0x80 | 25     0                   2501
From Puck 12, to GroupID 6, Len = 4, Return value of 2501 for SG (Prop #25)
```

```
MSGID        DLC    D0
                    RPPPPPPP
00000001100  0001   00110000
             1      48
From PC, to Puck 12, Len = 1, Get P (Prop #48)

*** The message you receive in response will be a dual 22-bit packed position.
*** Nodes 11-13 (Fingers 1-3) will respond in this way because they have dual encoders.
*** The first 3 bytes are P (motor encoder). The last three are JP (inner link encoder).
*** The encoder-count to joint-angle conversion formulas can be found here:
*** http://support.barrett.com/wiki/Hand/280/KinematicsJointRangesConversionFactors

MSGID        DLC    D0        D1        D2        D3        D4        D5
                    10MMMMMM  mmmmmmmm  LLLLLLLL  10MMMMMM  mmmmmmmm  LLLLLLLL
10110000011  0110   10000001  11101000  01001000  10000000  00111010  10011000
             6
From Puck 12, to GroupID 3, Len = 6
Return value of 125000 for P (Prop #48) and 15000 for JP (Prop #96)

MSGID        DLC    D0
                    RPPPPPPP
00000001100  0001   01100000
             1      96
From PC, to Puck 12, Len = 1, Get JP (Prop #96)
*** When asking for JP (Prop #96), the response will be a single 22-bit packed position.

MSGID        DLC    D0            D1            D2
                    10MMMMMM      mmmmmmmm      LLLLLLLL
10110000111  0011   10000000      00111010      10011000
             3                           15000
From Puck 12, to GroupID 7, Len = 3
Return value of 15000 for JP (Prop #96)

MSGID        DLC    D0
                    RPPPPPPP
00000001110  0001   00001001
             1      9
From PC, to Puck 14, Len = 1, Get TEMP (Prop #9)

MSGID        DLC    D0            D1            D2            D3
                    RPPPPPPP      00000000      LLLLLLLL      mmmmmmmm
10111000110  0100   10001001      00000000      00100011      00000000
             4      0x80 | 9      0             35
From Puck 14, GroupID 6, Len = 4
Return value of 35 (degrees Celsius) for TEMP (Prop #9)

MSGID        DLC    D0            D1            D2            D3
                    RPPPPPPP      00000000      LLLLLLLL      mmmmmmmm
00000001011  0100   10110100      00000000      00110111      00000000
             4      0x80 | 52     0             55            0
From PC, to Puck 11, Len = 4, Set V (Prop #44) to 55
Set the desired velocity of finger 1 to 55 encoder counts / ms

MSGID        DLC    D0            D1            D2            D3
                    RPPPPPPP      00000000      LLLLLLLL      mmmmmmmm
00000001011  0100   10001000      00000000      00000100      00000000
             4      0x80 | 8      0             4             0
From PC, to Puck 11, Len = 4, Set MODE (Prop #8) to 4 (MODE_VELOCITY)
Set the MODE of finger 1 to “velocity mode”
```