

```
# Calculate TWS, TWD and TWA (teoretical wind) by COG/SOG and AWS/AWA
# (c) 2019, Yacht Devices Ltd. Version 1.1, 29/06/2019
# Byte code size is 660 bytes (4% of user program space)

FW_CAN1_TO_CAN2=OFF
FW_CAN2_TO_CAN1=OFF

init()
{
    # T = 0 # No COG/SOG data, but all variables are 0 at start
    M = 0 - 1.0 # Float -1.0
}

# COG & SOG, Rapid Update
match(CAN1,0x1F80200,0xFFFF00)
{
    R = get(DATA+1,UINT8) & 0x3 # True Reference = 0
    A = get(DATA+2,UINT16) # Course Over Ground, 0.0001 rad/bit
    B = get(DATA+4,UINT16) # Speed Over Ground, 0.01 m/s
    if (R == 0) {
        if (A < 0xFFFFD) {
            if (B < 0xFFFFD) {

                T = timer() # Timestamp of COG/SOG data
                C = cast(A,FLOAT) / 10000 # COG, to radians
                S = cast(B,FLOAT) / 100 # SOG, m/s

            }
        }
    }
}

# Wind Data
match(CAN1,0x1FD0200,0xFFFF00)
{
    R = get(DATA+5,UINT8) & 0x7 # Apparent Wind = 2
    W = get(DATA+1,UINT16) # Wind speed, 0.01 m/s
    A = get(DATA+3,UINT16) # Wind direction, 0.0001 rad/bit

    # Have valid apparent wind data?
    if (R == 2) {
        if (A < 0xFFFFD) {
            if (W < 0xFFFFD) {

                # Have actual COG/SOG data (2000 ms)
                if (T > 0) {
                    if (timediff(T) < 2000) {

                        # TWD calculation when no speed or TWS is about zero
                        A = cast(A,FLOAT) / 10000 # Wind angle
                        Y = C + A # TWD=COG+AWA

                        if (S == 0) {

                            X = W # TWS is equal to AWS when no speed
                        }
                        else {
                            W = cast(W,FLOAT) / 100 # Wind speed, float

                            # Calculate TWS
                            U = sqrt(W*W + S*S - 2*W*S*cos(A))
                            X = cast(U*100,UINT16) # 0.01 m/s per bit

                            # Calculate TWD
                            if (X != 0) {

                                F = (S*S + U*U - W*W) / (2 * U * S)
                                # Possible possible rounding issue
                                if (F < M) {
                                    F = M
                                }
                                F = M_PI - acos( F ) # M_PI is 3.14... constant
                                if (A > M_PI) {
                                    Y = C - F
                                }
                                else {
                                    Y = C + F
                                }
                            }
                        }
                    }
                }

                # Check is TWD in 360 degrees
                if (Y > M_2PI) {
                    Y = Y - M_2PI # M_2PI is 2*PI constant, M_PI2 is PI/2
                }
                if (Y < 0) {
                    Y = Y + M_2PI
                }

                # Calculate TWA and check the range of value
                Z = Y - C
                if (Z < 0) {
                    Z = Z + M_2PI
                }

                # Transform to NMEA 2000 resolution, 0.0001 rad/bit
                Y = cast(Y*10000,UINT16)
                Z = cast(Z*10000,UINT16)

                # Theoretical Wind (ground referenced, referenced to True North;
                # calculated using COG/SOG) - TWD/TWS
                set(DATA+5, UINT8, 0xF8) # Data type
                set(DATA+1, UINT16, X) # TWS
                set(DATA+3, UINT16, Y) # TWD
                # Send message to the same interface, using the address
                # of the wind sensor
                send(CAN1)

                # Theoretical (Calculated to Centerline of the vessel, refernced
                # to ground; calculated using COG/SOG) - TWA/TWS
                set(DATA+5, UINT8, 0xFB) # Data type
                set(DATA+3, UINT16, Z) # TWA
                send(CAN1)
            }
        }
    }
}

# End of program
```