Write a self-contained function block in IEC 61131-3 Structured Text that implements a Kalman filter for estimating the position and velocity of an autonomous transport vehicle.

In IEC 61131-3 Structured Text, you can create a Function Block (FB) to implement a Kalman filter for estimating the position and velocity of an autonomous transport vehicle. Here's an example of how you can write such a function block:

iec_st Copy code

```
FUNCTION_BLOCK T_KalmanFilter
VAR_INPUT
    dt: REAL; // Time step (s)
    z: REAL; // Measurement (m)
    accel_noise: REAL; // Acceleration noise (m/s^2)
END_VAR

VAR_OUTPUT
    position: REAL; // Estimated position (m)
    velocity: REAL; // Estimated velocity (m/s)
END_VAR

VAR
    x: ARRAY [1..2] OF REAL; // State vector
    P: ARRAY [1..4] OF REAL; // State covariance matrix
    Q: ARRAY [1..4] OF REAL; // Process noise covariance matrix
    R: REAL; // Measurement noise covariance
    H: ARRAY [1..2] OF REAL; // Observation matrix
    F: ARRAY [1..4] OF REAL; // State transition matrix
    G: ARRAY [1..2] OF REAL; // Control input matrix
    K: ARRAY [1..2] OF REAL; // Kalman gain
    y: REAL; // Residual
    S: REAL; // Residual covariance
END_VAR

METHOD KalmanFilter: BOOL
    // Initialize matrices
    F[1] := 1; F[2] := dt; F[3] := 0; F[4] := 1;
    G[1] := 0.5 * dt * dt; G[2] := dt;
    H[1] := 1; H[2] := 0;

    // Predict step
    x := F * x + G * accel_noise;
    P := F * P * TRANSPOSE(F) + Q;

    // Update step
    y := z - H * x;
    S := H * P * TRANSPOSE(H) + R;
    K := P * TRANSPOSE(H) * (1 / S);

    // Update state and covariance estimates
    x := x + K * y;
    P := (IDENTITY2x2() - K * H) * P;

    // Output estimated position and velocity
    position := x[1];
    velocity := x[2];

    RETURN TRUE;
END_METHOD
END_FUNCTION_BLOCK
```

This implementation assumes that the autonomous transport vehicle moves in one dimension and its acceleration is constant between measurements. To use this Kalman filter, you will need to call the `KalmanFilter` method inside a control loop, passing the necessary input parameters, and reading the output estimated position and velocity. Note that you will need to initialize the state vector `x`, state covariance matrix `P`, process noise covariance matrix `Q`, and measurement noise covariance `R` before using the function block.

Provide the implementation of the functions TRANSPOSE and IDENTITY2x2.

Here are the implementations of the TRANSPOSE and IDENTITY2x2 functions in IEC 61131-3 Structured Text:

iec_st Copy code

```
FUNCTION TRANSPOSE : ARRAY [1..4] OF REAL
VAR_INPUT
    M: ARRAY [1..4] OF REAL; // 2x2 matrix to transpose
END_VAR

VAR_OUTPUT
    MT: ARRAY [1..4] OF REAL; // Transposed 2x2 matrix
END_VAR
BEGIN
    MT[1] := M[1];
    MT[2] := M[3];
    MT[3] := M[2];
    MT[4] := M[4];
END_FUNCTION
```

iec_st Copy code

```
FUNCTION IDENTITY2x2 : ARRAY [1..4] OF REAL
VAR_OUTPUT
    I: ARRAY [1..4] OF REAL; // 2x2 identity matrix
END_VAR
BEGIN
    I[1] := 1;
    I[2] := 0;
    I[3] := 0;
    I[4] := 1;
END_FUNCTION
```

The `TRANSPOSE` function takes a 2x2 matrix as an input and returns its transpose. The `IDENTITY2x2` function returns a 2x2 identity matrix. Note that these functions are specific to 2x2 matrices. You can include these functions in your project and use them in the T_KalmanFilter function block.