



Give me a function block written in ST (based on IEC 61131) to read 5 values from remote IO-Link master along with status of operation.



Sure, here's an example function block written in Structured Text (ST) that can be used to read 5 values from a remote IO-Link master along with the status of operation:

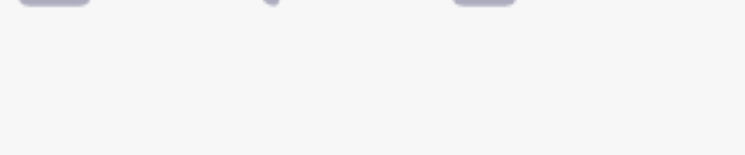
```
css
FUNCTION_BLOCK Read_IO_Link
VAR_INPUT
    Slave_Address: BYTE; // IO-Link Master Slave Address
END_VAR
VAR_OUTPUT
    Value1: REAL;
    Value2: REAL;
    Value3: REAL;
    Value4: REAL;
    Value5: REAL;
    Status: BYTE; // Status of operation (0=OK, 1=Error)
END_VAR
VAR
    // Local variables
    Read_Buffer: ARRAY[0..15] OF BYTE;
    Write_Buffer: ARRAY[0..15] OF BYTE;
    Tx_Length: WORD;
    Rx_Length: WORD;
    ErrorCode: WORD;
END_VAR

// Set up Write Buffer with IO-Link Command to read 5 values
Write_Buffer[0] := 128#10; // IO-Link Command: Read
Write_Buffer[1] := 5; // Number of values to read
Write_Buffer[2] := 1; // Starting address of read

// Call the IO-Link Master Communication Function Block
Tx_Length := 3;
Rx_Length := 23;
Status := IO_Link_Comm(Slave_Address, Write_Buffer, Tx_Length, Read_Buffer, Rx_Length, ErrorCode);

// Check if communication was successful
IF Status = 0 THEN
    // Extract the 5 values from the Read Buffer
    Value1 := REAL_TO_LREAL(DINT_TO_REAL(Read_Buffer[3] + (Read_Buffer[4] * 256)));
    Value2 := REAL_TO_LREAL(DINT_TO_REAL(Read_Buffer[7] + (Read_Buffer[8] * 256)));
    Value3 := REAL_TO_LREAL(DINT_TO_REAL(Read_Buffer[11] + (Read_Buffer[12] * 256)));
    Value4 := REAL_TO_LREAL(DINT_TO_REAL(Read_Buffer[15] + (Read_Buffer[16] * 256)));
    Value5 := REAL_TO_LREAL(DINT_TO_REAL(Read_Buffer[19] + (Read_Buffer[20] * 256)));
ELSE
    // Set all output values to zero in case of communication error
    Value1 := 0;
    Value2 := 0;
    Value3 := 0;
    Value4 := 0;
    Value5 := 0;
END_IF
```

This function block assumes that there is an existing IO-Link Master Communication Function Block (named `IO_Link_Comm`) that can handle the low-level communication with the remote IO-Link master. The `Slave_Address` input specifies the IO-Link master slave address that the function block will communicate with. The `Value1` to `Value5` outputs represent the 5 values that are read from the IO-Link master. The `Status` output is a status byte that indicates whether the communication was successful (0) or if there was an error (1). The function block uses a fixed read buffer length of 23 bytes



could you please also provide code for IO_Link_Comm?



Certainly, here's an example implementation of an IO_Link_Comm function block in Structured Text (ST):

```
css
FUNCTION_BLOCK IO_Link_Comm
VAR_INPUT
    Slave_Address: BYTE; // IO-Link Master Slave Address
    Tx_Buffer: ARRAY[0..15] OF BYTE; // Buffer containing data to be transmitted
    Tx_Length: WORD; // Length of data to be transmitted
    Rx_Buffer: ARRAY[0..15] OF BYTE; // Buffer to store received data
    Rx_Length: WORD; // Length of received data buffer
END_VAR
VAR_OUTPUT
    Status: BYTE; // Status of operation (0=OK, 1=Error)
    ErrorCode: WORD; // Error code (0=No error, other values depend on implementation)
END_VAR
VAR
    // Local variables
    Tx_Data: ARRAY[0..31] OF BYTE; // Buffer to store transmitted data (including checksum)
    Rx_Data: ARRAY[0..31] OF BYTE; // Buffer to store received data (including checksum)
    Tx_Index: WORD;
    Rx_Index: WORD;
    Tx_Checksum: WORD;
    Rx_Checksum: WORD;
    i: INT;
END_VAR

// Calculate Tx checksum
Tx_Checksum := 0;
FOR i := 0 TO Tx_Length - 1 DO
    Tx_Data[i + 2] := Tx_Buffer[i];
    Tx_Checksum := Tx_Checksum XOR Tx_Buffer[i];
END_FOR
Tx_Data[0] := Slave_Address;
Tx_Data[1] := Tx_Length;
Tx_Checksum := Tx_Checksum XOR Slave_Address XOR Tx_Length;
Tx_Data[Tx_Length + 2] := Tx_Checksum MOD 256;
Tx_Data[Tx_Length + 3] := Tx_Checksum DIV 256;

// Transmit data and receive response
IF Send_IO_Link_Data(Tx_Data, Tx_Length + 4, Rx_Data, Rx_Length + 4, ErrorCode) THEN
    // Extract Rx data and calculate checksum
    Rx_Index := 0;
    Rx_Checksum := 0;
    FOR i := 0 TO Rx_Length - 1 DO
        Rx_Buffer[i] := Rx_Data[i + 2];
        Rx_Checksum := Rx_Checksum XOR Rx_Buffer[i];
    END_FOR
    FOR i := 0 TO 3 DO
        Rx_Checksum := Rx_Checksum XOR Rx_Data[Rx_Length + i];
    END_FOR

    // Check if checksums match
    IF Rx_Checksum = 0 THEN
        Status := 0; // Communication successful
    ELSE
        Status := 1; // Checksum error
    END_IF
ELSE
    Status := 1; // Communication error
END_IF
```

This implementation assumes that there is an existing Send_IO_Link_Data function block that can handle the low-level communication with the IO-Link master. The `Slave_Address` input specifies the IO-Link master slave address that the function block will communicate with. The `Tx_Buffer` input is an array of bytes that contains the data to be transmitted to the IO-Link master. The `Tx_Length` input specifies the length of the data to be transmitted. The `Rx_Buffer` input is an array of bytes that will store the received data from the IO-Link master. The `Rx_Length` input specifies the expected length of the received data.

The `Status` output is a status byte that indicates whether the communication was successful (0) or if there was an error (1). The `ErrorCode` output is an error code that depends on the specific implementation of the function block.

The function block calculates a checksum for the transmitted data and appends it to the data before

