Write C-code for a Function block to be wrapped inside IEC 61131-3 Structured Text code. The function block shall create an OPC UA subscription using the Open62541 library using a provided Connection Handle as DWORD. As input it also gets an executed flag (type BOOL), a priority (BYTE) and a timeout (TIME). As output it provides three flags Done, Busy, Error, as well as two DWORDs ErrorID and SubscriptionHdl. As an IN/OUT variable the function block has a PublishingInterval (TIME).

Based on your requirements, here's the C code for the function block to be wrapped inside IEC 61131-3 Structured Text code:

c — Copy code

```c
#include <open62541.h>
#include <stdint.h>
#include <stdbool.h>

typedef struct {
    uint32_t ConnectionHdl;
    bool Executed;
    uint8_t Priority;
    uint32_t Timeout;
    uint32_t PublishingInterval;
} OPCUA_Subscription_Request;

typedef struct {
    bool Done;
    bool Busy;
    bool Error;
    uint32_t ErrorID;
    uint32_t SubscriptionHdl;
} OPCUA_Subscription_Response;

void OPCUA_CreateSubscription(OPCUA_Subscription_Request *request, OPCUA_Su
    static bool inProgress = false;

    if (!request->Executed) {
        response->Done = false;
        response->Busy = false;
        response->Error = false;
        response->ErrorID = 0;
        response->SubscriptionHdl = 0;
        inProgress = false;
        return;
    }

    if (inProgress) {
        response->Busy = true;
        return;
    }

    inProgress = true;

    UA_Client *client = (UA_Client *)(uintptr_t)request->ConnectionHdl;

    UA_CreateSubscriptionRequest createSubscriptionRequest = UA_CreateSubsc
    createSubscriptionRequest.requestedPublishingInterval = request->Publis
    createSubscriptionRequest.priority = request->Priority;

    UA_CreateSubscriptionResponse createSubscriptionResponse;
    createSubscriptionResponse = UA_Client_Subscriptions_create(client, cre
                                                                NULL, NULL

    if (createSubscriptionResponse.responseHeader.serviceResult == UA_STATU
        response->Done = true;
        response->Busy = false;
        response->Error = false;
        response->ErrorID = 0;
        response->SubscriptionHdl = (uint32_t)createSubscriptionResponse.su
        request->PublishingInterval = (uint32_t)createSubscriptionResponse.
    } else {
        response->Done = false;
        response->Busy = false;
        response->Error = true;
        response->ErrorID = (uint32_t)createSubscriptionResponse.responseHe
        response->SubscriptionHdl = 0;
    }

    inProgress = false;
}
```

This code defines a function `OPCUA_CreateSubscription()` that takes a pointer to an `OPCUA_Subscription_Request` struct as input and a pointer to an `OPCUA_Subscription_Response` struct as output. The input struct contains the necessary information for creating an OPC UA subscription, and the output struct provides the resulting status and subscription handle. The function uses the Open62541 library to create the subscription and updates the output struct accordingly.

Explain how to wrap this function block into IEC 61131-3 structured text and provide an example for calling it.

To wrap the C function block `OPCUA_CreateSubscription()` in IEC 61131-3 Structured Text, you can create a custom function block that utilizes the `EXTERNAL` keyword to reference the C function. Here's an example of how to do this:

1. Create a new IEC 61131-3 project in your PLC programming environment.
2. Add an external C function definition file, such as "OPCUA_CreateSubscription.c". This file should contain the C function block code provided in the previous answer.
3. Create a new Structured Text file and define a custom function block that references the C function:

iecst — Copy code

```iecst
FUNCTION_BLOCK F_OPCUA_CreateSubscription
    EXTERNAL "C" OPCUA_CreateSubscription : "OPCUA_CreateSubscription"; END
    VAR_INPUT
        ConnectionHdl: DWORD;
        Executed: BOOL;
        Priority: BYTE;
        Timeout: TIME;
        PublishingInterval: TIME;
    END_VAR
    VAR_OUTPUT
        Done: BOOL;
        Busy: BOOL;
        Error: BOOL;
        ErrorID: DWORD;
        SubscriptionHdl: DWORD;
    END_VAR
    VAR_IN_OUT
        PublishingInterval: TIME;
    END_VAR
END_FUNCTION_BLOCK
```

This function block definition maps the C function inputs and outputs to the corresponding IEC 61131-3 data types.

Now, you can use this custom function block in your IEC 61131-3 Structured Text code. Here's an example of how to call it:

iecst — Copy code

```iecst
PROGRAM Main
VAR
    OPCUA_ConnectionHdl: DWORD := 12345; (* Replace with your actual connec
    OPCUA_CreateSub: F_OPCUA_CreateSubscription;
    OPCUA_SubscriptionExecuted: BOOL := FALSE;
    OPCUA_SubscriptionPriority: BYTE := 100;
    OPCUA_SubscriptionTimeout: TIME := T#10s;
    OPCUA_PublishingInterval: TIME := T#500ms;
    OPCUA_SubscriptionDone: BOOL;
    OPCUA_SubscriptionBusy: BOOL;
    OPCUA_SubscriptionError: BOOL;
    OPCUA_SubscriptionErrorID: DWORD;
    OPCUA_SubscriptionHdl: DWORD;
END_VAR
```

In your cyclic task or main loop, call the function block:

iecst — Copy code

```iecst
OPCUA_CreateSub(
    ConnectionHdl := OPCUA_ConnectionHdl,
    Executed := OPCUA_SubscriptionExecuted,
    Priority := OPCUA_SubscriptionPriority,
    Timeout := OPCUA_SubscriptionTimeout,
    PublishingInterval := OPCUA_PublishingInterval,
    Done => OPCUA_SubscriptionDone,
    Busy => OPCUA_SubscriptionBusy,
    Error => OPCUA_SubscriptionError,
    ErrorID => OPCUA_SubscriptionErrorID,
    SubscriptionHdl => OPCUA_SubscriptionHdl
);
```

Make sure to set the `OPCUA_SubscriptionExecuted` variable to `TRUE` when you want to create the subscription. After calling the function block, you can check the output flags and handles to determine the result of the operation.