

## Fails "app.c"

```

/*****
MPLAB Harmony Application Source File

Company:
  Microchip Technology Inc.

File Name:
  app.c

Summary:
  This file contains the source code for the MPLAB Harmony application.

Description:
  This file contains the source code for the MPLAB Harmony application. It
  implements the logic of the application's state machine and it may call
  API routines of other MPLAB Harmony modules in the system, such as drivers,
  system services, and middleware. However, it does not call any of the
  system interfaces (such as the "Initialize" and "Tasks" functions) of any of
  the modules in the system or make any assumptions about when those functions
  are called. That is the responsibility of the configuration-specific system
  files.
*****/

// DOM-IGNORE-BEGIN
/*****
Copyright (c) 2013-2014 released Microchip Technology Inc. All rights reserved.

Microchip licenses to you the right to use, modify, copy and distribute
Software only when embedded on a Microchip microcontroller or digital signal
controller that is integrated into your product or third party product
(pursuant to the sublicense terms in the accompanying license agreement).

You should refer to the license agreement accompanying this Software for
additional information regarding your rights and obligations.

SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND,
EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF
MERCHANTABILITY, TITLE, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE.
IN NO EVENT SHALL MICROCHIP OR ITS LICENSORS BE LIABLE OR OBLIGATED UNDER
CONTRACT, NEGLIGENCE, STRICT LIABILITY, CONTRIBUTION, BREACH OF WARRANTY, OR
OTHER LEGAL EQUITABLE THEORY ANY DIRECT OR INDIRECT DAMAGES OR EXPENSES
INCLUDING BUT NOT LIMITED TO ANY INCIDENTAL, SPECIAL, INDIRECT, PUNITIVE OR
CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF PROCUREMENT OF
SUBSTITUTE GOODS, TECHNOLOGY, SERVICES, OR ANY CLAIMS BY THIRD PARTIES
(INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF), OR OTHER SIMILAR COSTS.
*****/
// DOM-IGNORE-END

// *****/
// *****/
// Section: Included Files
// *****/
// *****/

#include "app.h"
#include "M2M_interface.h"
#include "RONIN_CNC.h"
#include "UDPComm/PIC32_Harmony/udpcomm.h"

// *****/

```

```

// *****
// Section: Global Data Definitions
// *****
// *****

// *****
/* Application Data

Summary:
    Holds application data

Description:
    This structure holds the application's data.

Remarks:
    This structure should be initialized by the APP_Initialize function.

    Application strings and buffers are be defined outside this structure.
*/

APP_DATA appData;

// *****
// *****
// Section: Application Callback Functions
// *****
// *****

/* TODO: Add any necessary callback functions.
*/

// *****
// *****
// Section: Application Local Functions
// *****
// *****

uint8_t buff[20];
/* TODO: Add any necessary local functions.
*/
static void LedTask( void )
{
}

void Test_Callback ( uintptr_t context, uint32_t currTick )
{
    //SYS_CONSOLE_PRINT("Motor actual velocity: %d \r\n", CO_OD_RAM.motorActualVelocity1);
#ifdef RONIN_CM_V1_4
    static int state = 0;
    switch(state)
    {
        case 0: LED_RToggle(); break;
        case 1: LED_GToggle(); break;
        case 2: LED_BToggle(); break;
    }
    state++;
    // SYS_CONSOLE_PRINT("State: %d\r\n", state);
    if (state == 3) state = 0;
#endif
}

void Timer50ms_Callback ( uintptr_t context, uint32_t currTick )

```

```

{
#if USE_CAN == 1
    bool CO_OD_RAM_tst = false;
    int16_t speed = 0; //(int16_t)CO_OD_RAM.readAnalogueInput16Bit[1];
    static int pwmStep = 40;

    speed = (speed << 1); //conversion from 15bit to 16bit signed
    speed /= 2;

//    if(CO_OD_RAM_tst)
//    {
//        SYS_CONSOLE_PRINT("%d\t%d\t%d\n\r",
//            CO_OD_RAM.readAnalogueInput16Bit[0],
//            speed,
//            CO_OD_RAM.readAnalogueInput16Bit[2]);
//    }
//    CO_OD_RAM.writeAnalogueOutput16Bit[0] += pwmStep;
//    if (CO_OD_RAM.writeAnalogueOutput16Bit[0] >= 8000)
//    {
//        pwmStep = -200;
//        CO_OD_RAM.writeAnalogueOutput16Bit[0] = 8000;
//    }
//    else if (CO_OD_RAM.writeAnalogueOutput16Bit[0] <= 0)
//    {
//        CO_OD_RAM.writeOutput8Bit[0] ^= 1;
//        pwmStep = 100;
//        CO_OD_RAM.writeAnalogueOutput16Bit[0] = 0;
//    }
//    else
//    {
//        CO_OD_RAM.writeOutput8Bit[0] &= 0xFE;
//    }
#endif
}

// *****
// *****
// Section: Application Initialization and State Machine Functions
// *****
// *****

/*****
Function:
    void APP_Initialize ( void )

Remarks:
    See prototype in app.h.
*/

void APP_Initialize ( void )
{
    /* Place the App state machine in its initial state. */
    appData.state = APP_STATE_INIT;

    /* TODO: Initialize your application's state machine and other
    * parameters.
    */
}

```

```

/*****
Function:
    void APP_Tasks ( void )

Remarks:
    See prototype in app.h.
*/

void APP_Tasks ( void )
{

    /* Check the application's current state. */
    switch ( appData.state )
    {
        /* Application's initial state. */
        case APP_STATE_INIT:
        {
            if (SYS_TMR_Status(sysObj.sysTmr) == SYS_STATUS_READY)
            {
                appData.Tmr50msHandle = SYS_TMR_CallbackPeriodic(50, 0,
&Timer50ms_Callback);
                appData.Tmr1sHandle = SYS_TMR_CallbackPeriodic(100, 0, &Test_Callback);
            }
            else
                break;

            RONIN_COMMAND_PROC_Initialize();
            //    SYS_PRINT("Application startup ok!\r\n");
            //    CO_OD_RAM.writeAnalogueOutput16Bit[0] = 0;
            RONINCNC_Initialize();
            ESP_COMM_Initialize(0);
            appData.state = APP_STATE_WAIT_IP;

            break;
        }
        case APP_STATE_WAIT_IP:
            if (UDPCOMM_GetState() == UDPCOMM_READY)
            {
                appData.state = APP_STATE_WAITING_CAN_READY;
            }
            break;
        case APP_STATE_WAITING_CAN_READY:
            if (CANOPENAPP_CANopenReady())
            {
                SYS_CONSOLE_PRINT("Application ready...\r\n");
                appData.state = APP_STATE_SERVICE_TASKS;
            }
            break;
        case APP_STATE_SERVICE_TASKS:
        {
            RONINCNC_Tasks();
            ESP_COMM_Tasks();
            RONIN_COMMAND_PROC_Tasks();

            LedTask();
            //SYS_CMD_READY_TO_READ();

            // SYS_MESSAGE(appData.ConsoleBuf);
            break;
        }

        /* TODO: implement your application state machine.*/
    }
}

```

```
/* The default state should never be executed. */
default:
{
    /* TODO: Handle error in application's state machine. */
    break;
}
}
```

```
/******
End of File
*/
```

## Fails "app.h"

```

/*****
MPLAB Harmony Application Header File

```

```

Company:
  Microchip Technology Inc.

```

```

File Name:
  app.h

```

```

Summary:
  This header file provides prototypes and definitions for the application.

```

```

Description:
  This header file provides function prototypes and data type definitions for
  the application. Some of these are required by the system (such as the
  "APP_Initialize" and "APP_Tasks" prototypes) and some of them are only used
  internally by the application (such as the "APP_STATES" definition). Both
  are defined here for convenience.

```

```

*****/

```

```

//DOM-IGNORE-BEGIN

```

```

/*****
Copyright (c) 2013-2014 released Microchip Technology Inc. All rights reserved.

```

Microchip licenses to you the right to use, modify, copy and distribute Software only when embedded on a Microchip microcontroller or digital signal controller that is integrated into your product or third party product (pursuant to the sublicense terms in the accompanying license agreement).

You should refer to the license agreement accompanying this Software for additional information regarding your rights and obligations.

SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY, TITLE, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL MICROCHIP OR ITS LICENSORS BE LIABLE OR OBLIGATED UNDER CONTRACT, NEGLIGENCE, STRICT LIABILITY, CONTRIBUTION, BREACH OF WARRANTY, OR OTHER LEGAL EQUITABLE THEORY ANY DIRECT OR INDIRECT DAMAGES OR EXPENSES INCLUDING BUT NOT LIMITED TO ANY INCIDENTAL, SPECIAL, INDIRECT, PUNITIVE OR CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF PROCUREMENT OF SUBSTITUTE GOODS, TECHNOLOGY, SERVICES, OR ANY CLAIMS BY THIRD PARTIES (INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF), OR OTHER SIMILAR COSTS.

```

*****/
//DOM-IGNORE-END

```

```

#ifndef _APP_H
#define _APP_H

```

```

// *****/
// *****/
// Section: Included Files
// *****/
// *****/

```

```

#include <stdint.h>
#include <stdbool.h>
#include <stddef.h>
#include <stdlib.h>
#include "system_config.h"
#include "system_definitions.h"
#include "RONINCommandProc.h"

```

```

// DOM-IGNORE-BEGIN
#ifdef __cplusplus // Provide C++ Compatibility

extern "C" {

#endif
// DOM-IGNORE-END

// *****
// *****
// Section: Type Definitions
// *****
// *****

#define USE_CAN 1

    // *****
/* Application states

    Summary:
        Application states enumeration

    Description:
        This enumeration defines the valid application states. These states
        determine the behavior of the application at various times.
*/

typedef enum
{
    /* Application's state machine's initial state. */
    APP_STATE_INIT=0,
    APP_STATE_WAITING_CAN_READY,
    APP_STATE_WAIT_IP,
    APP_STATE_SERVICE_TASKS,

    /* TODO: Define states used by the application state machine. */
} APP_STATES;

// *****
/* Application Data

    Summary:
        Holds application data

    Description:
        This structure holds the application's data.

    Remarks:
        Application strings and buffers are be defined outside this structure.
*/

typedef struct
{
    /* The application's current state */
    APP_STATES state;

    /* TODO: Define any additional data used by the application. */

    SYS_TMR_HANDLE Tmr1sHandle;
    SYS_TMR_HANDLE Tmr50msHandle;
    uint8_t ConsoleBuf[100];

```

```
} APP_DATA;
```

```
// *****  
// *****  
// Section: Application Callback Routines  
// *****  
// *****  
/* These routines are called by drivers when certain events occur.  
*/
```

```
// *****  
// *****  
// Section: Application Initialization and State Machine Functions  
// *****  
// *****
```

```
/* *****
```

Function:

```
void APP_Initialize ( void )
```

Summary:

MPLAB Harmony application initialization routine.

Description:

This function initializes the Harmony application. It places the application in its initial state and prepares it to run so that its APP\_Tasks function can be called.

Precondition:

All other system initialization routines should be called before calling this routine (in "SYS\_Initialize").

Parameters:

None.

Returns:

None.

Example:

```
<code>  
APP_Initialize();  
</code>
```

Remarks:

This routine must be called from the SYS\_Initialize function.

```
*/
```

```
void APP_Initialize ( void );
```

```
/* *****
```

Function:

```
void APP_Tasks ( void )
```

Summary:

MPLAB Harmony Demo application tasks function

Description:

This routine is the Harmony Demo application's tasks function. It defines the application's state machine and core logic.

Precondition:



The system and application initialization ("SYS\_Initialize") should be called before calling this.

Parameters:

None.

Returns:

None.

Example:

```
<code>
APP_Tasks();
</code>
```

Remarks:

This routine must be called from SYS\_Tasks() routine.

\*/

```
void APP_Tasks( void );
```

```
#endif /* _APP_H */
```

```
//DOM-IGNORE-BEGIN
```

```
#ifdef __cplusplus
```

```
}
```

```
#endif
```

```
//DOM-IGNORE-END
```

```
/******
```

```
End of File
```

```
*/
```

## Fails "CANopenApp.c"

```

/*****
MPLAB Harmony Application Source File

Company:
    Microchip Technology Inc.

File Name:
    canopenapp.c

Summary:
    This file contains the source code for the MPLAB Harmony application.

Description:
    This file contains the source code for the MPLAB Harmony application. It
    implements the logic of the application's state machine and it may call
    API routines of other MPLAB Harmony modules in the system, such as drivers,
    system services, and middleware. However, it does not call any of the
    system interfaces (such as the "Initialize" and "Tasks" functions) of any of
    the modules in the system or make any assumptions about when those functions
    are called. That is the responsibility of the configuration-specific system
    files.
*****/

// DOM-IGNORE-BEGIN
/*****
Copyright (c) 2013-2014 released Microchip Technology Inc. All rights reserved.

Microchip licenses to you the right to use, modify, copy and distribute
Software only when embedded on a Microchip microcontroller or digital signal
controller that is integrated into your product or third party product
(pursuant to the sublicense terms in the accompanying license agreement).

You should refer to the license agreement accompanying this Software for
additional information regarding your rights and obligations.

SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND,
EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF
MERCHANTABILITY, TITLE, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE.
IN NO EVENT SHALL MICROCHIP OR ITS LICENSORS BE LIABLE OR OBLIGATED UNDER
CONTRACT, NEGLIGENCE, STRICT LIABILITY, CONTRIBUTION, BREACH OF WARRANTY, OR
OTHER LEGAL EQUITABLE THEORY ANY DIRECT OR INDIRECT DAMAGES OR EXPENSES
INCLUDING BUT NOT LIMITED TO ANY INCIDENTAL, SPECIAL, INDIRECT, PUNITIVE OR
CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF PROCUREMENT OF
SUBSTITUTE GOODS, TECHNOLOGY, SERVICES, OR ANY CLAIMS BY THIRD PARTIES
(INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF), OR OTHER SIMILAR COSTS.
*****/
// DOM-IGNORE-END//

*****
// *****
// Section: Included Files
// *****
// *****

#include "canopenapp.h"
#include "UDPComm/PIC32_Harmony/udpcomm.h"

// *****
// *****

```

```

// Section: Global Data Definitions
// *****
// *****
#define CO_TMR_ISR_FLAG      IFS0bits.T2IF      /* Interrupt Flag bit */

typedef enum
{
    CANOPENAPP_TRANSFER_STATE_Init = 0,
    CANOPENAPP_TRANSFER_STATE_Idle,
    CANOPENAPP_TRANSFER_STATE_DownloadingObjects,
    CANOPENAPP_TRANSFER_STATE_UploadingObjects,
    CANOPENAPP_TRANSFER_STATE_Error
} CANOPENAPP_TRANSFER_STATE;

typedef struct
{
    CANOPENAPP_TRANSFER_STATE State;
    uint8_t TransferNodeID;
    CO_OBJECT *ObjectsToTransfer;
    CO_OBJECT *TransferBaseObject;
    int16_t TransferCount;

    CO_OBJECT *ObjectsToUpload;
    CO_OBJECT *UploadBase;
    int16_t UploadCount;
    CO_OBJECT *ObjectsToDownload;
    CO_OBJECT *DownloadBase;
    int16_t DownloadCount;
    CANOPENAPP_SDO_ClientOpeResult_t TransferResult;
    struct
    {
        uint32_t WaitComplete:1;
    } Flags;
} CANOPENAPP_TRANSFER_TAKS_DATA;

// *****
/* Application Data

Summary:
    Holds application data

Description:
    This structure holds the application's data.

Remarks:
    This structure should be initialized by the APP_Initialize function.

    Application strings and buffers are be defined outside this structure.
*/

const CO_CANbitRateData_t CO_CANbitRateData[8] = {CO_CANbitRateDataInitializers};
CANOPENAPP_DATA canopenappData;
CANOPENAPP_TRANSFER_TAKS_DATA _td;

void Tmrlms_Callback ( uintptr_t context, uint32_t currTick );

// *****
// *****

```

```

// Section: Application Callback Functions
// *****
// *****

/* TODO: Add any necessary callback functions.
*/

// *****
// *****
// Section: Application Local Functions
// *****
// *****

/* TODO: Add any necessary local functions.
*/
#ifdef RONIN_CM_V1_4
void __ISR(_CAN1_VECTOR, ip15AUTO) IntCAN1(void)
{
    #if USE_CAN == 1
        Nop();
        Nop();
        Nop();

        CO_CANinterrupt(CO->CANmodule[0]);
        IFS4bits.CAN1IF = 0;
    #endif
}
#endif
#ifdef RONIN_PIC32MZ_starter
void __ISR(_CAN1_VECTOR, ip15AUTO) IntCAN1(void)
{
    Nop();
    Nop();
    Nop();

    CO_CANinterrupt(CO->CANmodule[0]);
    IFS4bits.CAN1IF = 0;
}
#endif
#ifdef RONIN_CM_B
void __ISR(_CAN1_VECTOR, ip15AUTO) IntCAN1(void)
{
    Nop();
    Nop();
    Nop();

    CO_CANinterrupt(CO->CANmodule[0]);
    IFS4bits.CAN1IF = 0;
}
#endif

#ifdef RONIN_Explorer16
void __ISR(_CAN_1_VECTOR, ip11AUTO) IntCAN1(void)
{
    Nop();
    Nop();
    Nop();

    CO_CANinterrupt(CO->CANmodule[0]);
    IFS1bits.CAN1IF = 0;
}
#endif

```

```

void TmrIms_Callback ( uintptr_t context, uint32_t currTick )
{
#if USE_CAN == 1
    CO_TMR_ISR_FLAG = 0;
    bool_t syncWas;
    canopenappData.CO_timerIms++;

    if(CO->CANmodule[0]->CANnormal)
    {
        /* Process Sync */
        //syncWas = CO_process_SYNC(CO, 1000, NULL);
        syncWas = false;

        /* Read inputs */
        CO_process_RPDO(CO, syncWas);

        /* Further I/O or nonblocking application code may go here. */
#if CO_NO_TRACE > 0
        OD_time.epochTimeOffsetMs++;
        for(i=0; i<OD_traceEnable && i<CO_NO_TRACE; i++) {
            CO_trace_process(CO->trace[i], OD_time.epochTimeOffsetMs);
        }
#endif
        /* Write outputs */
        CO_process_TPDO(CO, syncWas, 1000, NULL);

        /* verify timer overflow */
        if(CO_TMR_ISR_FLAG == 1){
            CO_errorReport(CO->em, CO_EM_ISR_TIMER_OVERFLOW, CO EMC SOFTWARE_INTERNAL, 0);
            CO_TMR_ISR_FLAG = 0;
        }
    }
    /* calculate cycle time for performance measurement */
    // uint16_t t = CO_TMR_TMR / (CO_PBCLK / 100);
    // OD_performance[ODA_performance_timerCycleTime] = t;
    // if(t > OD_performance[ODA_performance_timerCycleMaxTime])
    //     OD_performance[ODA_performance_timerCycleMaxTime] = t;
#endif
}

// *****
// *****
// Section: Application Initialization and State Machine Functions
// *****
// *****

/*****
Function:
    void CANOPENAPP_Initialize ( void )

Remarks:
    See prototype in canopenapp.h.
*/

void CANOPENAPP_Initialize ( void )
{
#if USE_CAN == 1

    /* Place the App state machine in its initial state. */
    canopenappData.state = CANOPENAPP_STATE_INIT;

    /* TODO: Initialize your application's state machine and other
    * parameters.

```

```

*/

//C1MODEOff();          //Transciever mode - high speed

canopenappData.CO_timerlms = 0;
canopenappData.timerlmsDiff = 0;
canopenappData.timerlmsPrevious = 0;

canopenappData.reset = CO_RESET_NOT;
canopenappData.Flags.SDO_ClientUploadStarted = 0;
canopenappData.Flags.SDO_ClientDownloadStarted = 0;

#ifdef RONIN_CM_V1_4
    IFS4bits.CAN1IF = 0;
    IPC37bits.CAN1IP = 5;
    IEC4bits.CAN1IE = 1;
#endif

#ifdef RONIN_PIC32MZ_starter
    IFS4bits.CAN1IF = 0;
    IPC37bits.CAN1IP = 5;
    IEC4bits.CAN1IE = 1;
#endif

#ifdef RONIN_CM_B
    IFS4bits.CAN1IF = 0;
    IPC37bits.CAN1IP = 5;
    IEC4bits.CAN1IE = 1;
#endif

#ifdef RONIN_Explorer16
    IFS1bits.CAN1IF = 0;
    IPC11bits.CAN1IP = 1;
    IEC1bits.CAN1IE = 1;
#endif

//CAN tranceiver - high speed mode
CAN_MODEOff();

//CAN module address is given just 0 for CAN1 registers as driver
//address calculation macro uses only CAN1
/* Allocate memory */
canopenappData.err = CO_new(&canopenappData.HeapMemoryUsed);
if (canopenappData.err != CO_ERROR_NO) {
    while(1);
}

canopenappData.err = CO_CANinit((void *)_CAN1_BASE_ADDRESS, CANAAPP_BITRATE);

//    canopenappData.err = CO_init(
//        0/* CAN module address */,
//        9/* NodeID */,
//        CANAAPP_BITRATE /* bit rate */);

canopenappData.err = CO_CANOpenInit(9);    //TODO: How to get ID from configurator?

canopenappData.reset = CO_RESET_NOT;
canopenappData.timerlmsPrevious = canopenappData.CO_timerlms;
canopenappData.DownloadResult = CANOPENAPP_SDO_ClientIdle;
canopenappData.UploadResult = CANOPENAPP_SDO_ClientIdle;
#endif
}

```

```

CANOPENAPP_SDO_ClientOpeResult_t CANOPENAPP_TRANSFER_Initiate(uint8_t nodeID, CO_OBJECT
*objArr, uint16_t len, CANOPENAPP_SDO_TRANSFER_TYPE tType, bool waitComplete)
{
    CANOPENAPP_SDO_ClientOpeResult_t ret = CANOPENAPP_SDO_ClientBusy;
    if (!_td.TransferCount && _td.State == CANOPENAPP_TRANSFER_STATE_Idle)
    {
        _td.TransferNodeID = nodeID;
        _td.TransferBaseObject = objArr;
        _td.ObjectsToTransfer = objArr;
        _td.TransferCount = len;
        _td.Flags.WaitComplete = waitComplete ? 1 : 0;
        _td.TransferResult = CANOPENAPP_SDO_ClientBusy;
        switch(tType)
        {
            case CANOPENAPP_SDO_TRANSFER_UPLOAD:
                _td.State = CANOPENAPP_TRANSFER_STATE_UploadingObjects;
                break;
            case CANOPENAPP_SDO_TRANSFER_DOWNLOAD:
                _td.State = CANOPENAPP_TRANSFER_STATE_DownloadingObjects;
                break;
            default:
                break;
        }
        ret = CANOPENAPP_SDO_ClientOpeScheduled;
    }

    return ret;
}

bool CANOPENAPP_CANOpenReady(void)
{
    bool ret = false;
    if (_td.State > CANOPENAPP_TRANSFER_STATE_Init &&
        canopenappData.state == CANOPENAPP_STATE_SERVICE_TASKS)
        ret = true;
    return ret;
}

CANOPENAPP_SDO_ClientOpeResult_t CANOPENAPP_TRANSFER_Result(CO_OBJECT *transferBaseObj)
{
    CANOPENAPP_SDO_ClientOpeResult_t ret = CANOPENAPP_SDO_ClientBusy;
    /* Check if completion status request is about the same object array
    for which transfer was initiated, if not, show that busy */
    if (transferBaseObj == _td.TransferBaseObject || _td.State ==
CANOPENAPP_TRANSFER_STATE_Idle)
    {
        ret = _td.TransferResult;
        if (_td.TransferCount == 0)
        {
            _td.State = CANOPENAPP_TRANSFER_STATE_Idle;
        }
    }
    return ret;
}

/*
 * Handles data bulk transfer using SDO.
 * Called periodically in CANOPENAPP_Tasks
 */
void CANOPENAPP_DataTransferTasks(void)
{

```

```

CANOPENAPP_SDO_ClientOpeResult_t res;
switch(_td.State)
{
    case CANOPENAPP_TRANSFER_STATE_Init:
        _td.TransferCount = 0;
        _td.State = CANOPENAPP_TRANSFER_STATE_Idle;
        break;
    case CANOPENAPP_TRANSFER_STATE_Idle:
        _td.TransferResult = CANOPENAPP_SDO_ClientIdle;
        break;
    case CANOPENAPP_TRANSFER_STATE_DownloadingObjects:
        res = SDO_ClientDownload(
            _td.TransferNodeID,
            _td.ObjectsToTransfer->Idx,
            _td.ObjectsToTransfer->SubIdx,
            (uint8_t*)&_td.ObjectsToTransfer->Value,
            _td.ObjectsToTransfer->Size,
            NULL);

        switch(res)
        {
            case CANOPENAPP_SDO_ClientOpeResultOK:
                _td.TransferCount--;
                if (_td.TransferCount)
                    _td.ObjectsToTransfer++;
                else
                {
                    _td.TransferResult = CANOPENAPP_SDO_ClientOpeResultOK;
                    if (!_td.Flags.WaitComplete) _td.State =
CANOPENAPP_TRANSFER_STATE_Idle;
                }
                break;
            case CANOPENAPP_SDO_ClientCANerror:
            case CANOPENAPP_SDO_ClientArgError:
                _td.TransferCount = 0;
                _td.TransferResult = CANOPENAPP_SDO_ClientCANerror;
                if (!_td.Flags.WaitComplete) _td.State =
CANOPENAPP_TRANSFER_STATE_Idle;
                break;
            default:
                break;
        }
        break;
    case CANOPENAPP_TRANSFER_STATE_UploadingObjects:
        if (_td.TransferCount)
        {
            res = SDO_ClientUpload( _td.TransferNodeID,
                _td.ObjectsToTransfer->Idx,
                _td.ObjectsToTransfer->SubIdx,
                &_td.ObjectsToTransfer->Value,
                NULL);

            switch(res)
            {
                case CANOPENAPP_SDO_ClientOpeResultOK:
                    SYS_CONSOLE_PRINT("CAN%x(%x)\t%x\r\n",
                        _td.ObjectsToTransfer->Idx,
                        _td.ObjectsToTransfer->SubIdx,
                        _td.ObjectsToTransfer->Value);
                    _td.TransferCount--;
                    if (_td.TransferCount)
                        _td.ObjectsToTransfer++;
                    else
                    {
                        _td.TransferResult = CANOPENAPP_SDO_ClientOpeResultOK;

```



```

        if (!_td.Flags.WaitComplete) _td.State =
CANOPENAPP_TRANSFER_STATE_Idle;
    }
    break;
    case CANOPENAPP_SDO_ClientCANError:
    case CANOPENAPP_SDO_ClientArgError:
        _td.TransferCount = 0;
        _td.TransferResult = CANOPENAPP_SDO_ClientCANError;
        if (!_td.Flags.WaitComplete) _td.State =
CANOPENAPP_TRANSFER_STATE_Idle;
    break;
    default:
        break;
    }
}
break;
case CANOPENAPP_TRANSFER_STATE_Error:
    SYS_CONSOLE_PRINT("CAN open data transfer error\n\r");
    _td.State = CANOPENAPP_TRANSFER_STATE_Init;
    break;
default:
    break;
}
}

/*****
Function:
void CANOPENAPP_Tasks ( void )

Remarks:
See prototype in canopenapp.h.
*/

void CANOPENAPP_Tasks ( void )
{
    static uint8_t rxBuff[55], tmptmr = 0;
    static CO_SDOclient_return_t sdoClient_ret;
    CO_SDOclient_return_t ret = 100;
    size_t sizeIndicated = 0, sizeTransferred = 0;
    CO_SDO_abortCode_t sdoAbortCode;

#if USE_CAN == 1
    if (canopenappData.CO_timer1ms >= canopenappData.timer1msPrevious)
    {
        canopenappData.timer1msDiff =
            canopenappData.CO_timer1ms - canopenappData.timer1msPrevious;
    }
    else
    {
        canopenappData.timer1msDiff =
            canopenappData.timer1msPrevious - canopenappData.CO_timer1ms
            + 0xFFFFFFFF;
    }
    canopenappData.timer1msPrevious = canopenappData.CO_timer1ms;
    tmptmr += canopenappData.timer1msDiff;

    /* Check the application's current state. */
    switch ( canopenappData.state )
    {
        /* Application's initial state. */
        case CANOPENAPP_STATE_INIT:
        {
            bool appInitialized = false;

```

```

        if (SYS_TMR_Status(sysObj.sysTmr) == SYS_STATUS_READY && (UDPCOMM_GetState()
== UDPCOMM_READY))
        {
            canopenappData.Tmr1msHandle = SYS_TMR_CallbackPeriodic(1, 0,
&Tmr1ms_Callback);
            appInitialized = true;
        }
        if (appInitialized)
        {
            /* start CAN */
            CO_CANsetNormalMode(CO->CANmodule[0]);

            canopenappData.netBootupTimer = 0;
            CO_NMT_sendCommand(CO->NMT, CO_NMT_RESET_COMMUNICATION, 0); //Adress all
nodes
            canopenappData.state = CANOPENAPP_STATE_RESETTING_NODES;
        }
        break;
    }
    case CANOPENAPP_STATE_RESETTING_NODES:

        canopenappData.netBootupTimer += canopenappData.timer1msDiff;
        if (canopenappData.netBootupTimer >= 5)
        {
            canopenappData.netBootupTimer = 0;
            CO_NMT_sendCommand(CO->NMT, CO_NMT_ENTER_OPERATIONAL, 0); //Adress all
nodes
            canopenappData.state = CANOPENAPP_STATE_STARTING_NODES;
        }
        break;
    case CANOPENAPP_STATE_STARTING_NODES:
        canopenappData.netBootupTimer += canopenappData.timer1msDiff;
        if (canopenappData.netBootupTimer >= 10)
        {
            canopenappData.state = CANOPENAPP_STATE_SETTING_UP_NODES;
        }
        break;
    case CANOPENAPP_STATE_SETTING_UP_NODES:

        CO->NMT->resetCommand = 0;          //TODO: Workaround, because was set and not
unset somewhere
                                           //as a result constantly resets CAN

        canopenappData.state = CANOPENAPP_STATE_SERVICE_TASKS;

        break;
    case CANOPENAPP_STATE_SERVICE_TASKS:
    {
        canopenappData.reset = CO_RESET_NOT;
        if (CO_OD_RAM.errorStatusBits[0] != 0) CO_OD_RAM.errorStatusBits[0] = 0;
//TODO: Workaround
        if (CO_OD_RAM.errorStatusBits[2] != 0) CO_OD_RAM.errorStatusBits[2] = 0;
        if (CO_OD_RAM.errorStatusBits[3] != 0) CO_OD_RAM.errorStatusBits[3] = 0;

        /* CANopen process */
        canopenappData.reset = CO_process(CO, canopenappData.timer1msDiff * 1000,
NULL);

        if (canopenappData.reset != CO_RESET_NOT)
            canopenappData.state = CANOPENAPP_STATE_ERROR;
    }

```

```

if (CO->NMT->operatingState != CO_NMT_OPERATIONAL)
    canopenappData.state = CANOPENAPP_STATE_ERROR;

CANOPENAPP_DataTransferTasks();

/*
 * SDO client upload process
 */
switch(canopenappData.UploadResult)
{
    case CANOPENAPP_SDO_ClientOpeScheduled:
        ret = CO_SDOclientUpload(
            CO->SDOclient[0],
            canopenappData.timer1msDiff * 1000,
            &sdoAbortCode,
            &sizeIndicated, &sizeTransferred, NULL);
        if (ret <= CO_SDOcli_ok_communicationEnd)
        {
            if (ret == CO_SDOcli_ok_communicationEnd)
            {
                CO_SDOclientUploadBufRead(
                    CO->SDOclient[0],
                    canopenappData.SDO_ClientUploadBuff,
                    sizeTransferred);
                canopenappData.UploadResult =
CANOPENAPP_SDO_ClientOpeResultOK;
            }
            else
                canopenappData.UploadResult = CANOPENAPP_SDO_ClientCANerror;
                CO_SDOclientClose(CO->SDOclient[0]);
        }
        break;
    default:
        break;
}

/*
 * SDO client download process
 */
switch(canopenappData.DownloadResult)
{
    case CANOPENAPP_SDO_ClientOpeScheduled:
        ret = CO_SDOclientDownload(
            CO->SDOclient[0],
            canopenappData.timer1msDiff * 1000,
            false,
            &sdoAbortCode,
            &sizeTransferred, NULL);
        if (ret <= CO_SDOcli_ok_communicationEnd)
        {
            if (ret == CO_SDOcli_ok_communicationEnd)
                canopenappData.DownloadResult =
CANOPENAPP_SDO_ClientOpeResultOK;
            else
                canopenappData.DownloadResult = CANOPENAPP_SDO_ClientCANerror;
                CO_SDOclientClose(CO->SDOclient[0]);
        }
        break;
    default:
        break;
}
break;

```

```

}
case CANOPENAPP_STATE_ERROR:
    canopenappData.netBootupTimer = 0;
    CO_NMT_sendCommand(CO->NMT, CO_NMT_RESET_COMMUNICATION, 0);
    canopenappData.state = CANOPENAPP_STATE_RESETTING_NODES;
    break;

/* TODO: implement your application state machine.*/

/* The default state should never be executed. */
default:
{
    /* TODO: Handle error in application's state machine. */
    break;
}
}
#endif
}

```

```

CANOPENAPP_SDO_ClientOpeResult_t CANOPENAPP_ReadPDOparameters(uint8_t nodeID, uint16_t
startIdx, uint16_t PDOparamCount, uint32_t *buff)

```

```

{
    static uint8_t subidx = 0;
    static uint32_t objectCount = 0;
    static uint16_t curridx = 0;
    static uint8_t phase = 0;
    static uint32_t bufIdx;
    CANOPENAPP_SDO_ClientOpeResult_t ret = CANOPENAPP_SDO_ClientBusy;
    if (curridx == 0)
    {
        curridx = startIdx;
        bufIdx = 0;
        subidx = 0;
        objectCount = 0;
        phase = 0;
    }
    ret = SDO_ClientUpload(1, curridx, subidx, buff + bufIdx, NULL);
    switch(ret)
    {
        case CANOPENAPP_SDO_ClientOpeResultOK:
            ret = CANOPENAPP_SDO_ClientOpeScheduled;
            if (subidx == 0)
                objectCount = buff[bufIdx] + 1; //Max subindex + 1
            bufIdx++;
            subidx++;
            phase = 0;
            if (subidx == objectCount)
            {
                objectCount = 0;
                subidx = 0;
                curridx++;
                if (curridx == PDOparamCount + startIdx)
                {
                    ret = CANOPENAPP_SDO_ClientOpeResultOK;
                    curridx = 0;
                }
            }
            break;
        case CANOPENAPP_SDO_ClientOpeScheduled:
            phase = 1;
            break;
        case CANOPENAPP_SDO_ClientBusy:

```

```

        break;
    case CANOPENAPP_SDO_ClientCANError:
    case CANOPENAPP_SDO_ClientArgError:
        ret = CANOPENAPP_SDO_ClientCANError;
    default: break;
}

return ret;
}

CANOPENAPP_SDO_ClientOpeResult_t SDO_ClientUpload( uint8_t nodeID,
                                                    uint16_t idx,
                                                    uint8_t subidx,
                                                    uint8_t *dataRx,
                                                    CANOPENAPP_SDO_CLIENT_CALLBACK
callback)
{
    CO_SDOclient_return_t ret = CO_SDOcli_ok_communicationEnd;
    CANOPENAPP_SDO_ClientOpeResult_t result = CANOPENAPP_SDO_ClientBusy;
    CO_SDOclient_t *SDOclient = CO->SDOclient[0];
    switch(canopenappData.UploadResult)
    {
        case CANOPENAPP_SDO_ClientIdle:
            canopenappData.SDO_ClientUploadBuff = dataRx;
            /* Setup client. */
            ret = CO_SDOclient_setup( SDOclient, 0, 0, nodeID);
            if(ret == CO_SDOcli_ok_communicationEnd)
            {
                /* Initiate upload. */
                ret = CO_SDOclientUploadInitiate(
                    SDOclient, idx, subidx,
                    CO_SDO_SERVER_TIMEOUT,
                    CO_SDO_BLOCK_TRANSFER_ENABLE);
                if (ret == CO_SDOcli_ok_communicationEnd)
                {
                    result = canopenappData.UploadResult =
CANOPENAPP_SDO_ClientOpeScheduled;
                    break;
                }
            }
            result = CANOPENAPP_SDO_ClientCANError;
            break;
        case CANOPENAPP_SDO_ClientOpeResultOK:
        case CANOPENAPP_SDO_ClientCANError:
        case CANOPENAPP_SDO_ClientArgError:
            result = canopenappData.UploadResult;
            canopenappData.UploadResult = CANOPENAPP_SDO_ClientIdle;
        default:
            break;
    }
    return result;
}

CANOPENAPP_SDO_ClientOpeResult_t SDO_ClientDownload( uint8_t nodeID,
                                                      uint16_t idx,
                                                      uint8_t subidx,
                                                      uint8_t *dataTx,
                                                      size_t txSize,
                                                      CANOPENAPP_SDO_CLIENT_CALLBACK
callback)
{
    CO_SDOclient_return_t ret = CO_SDOcli_ok_communicationEnd;
    CANOPENAPP_SDO_ClientOpeResult_t result = CANOPENAPP_SDO_ClientBusy;

```

```

CO_SDOclient_t *SDOclient = CO->SDOclient[0];
switch(canopenappData.DownloadResult)
{
    case CANOPENAPP_SDO_ClientIdle:
        canopenappData.DownloadSize = txSize;
        canopenappData.SDO_ClientDownloadBuff = dataTx;

        /* Setup client. */
        ret = CO_SDOclient_setup( SDOclient, 0, 0, nodeID);
        if(ret == CO_SDOcli_ok_communicationEnd)
        {
            /* Initiate download. */
            ret = CO_SDOclientDownloadInitiate(
                SDOclient, idx, subidx,
                txSize,
                CO_SDO_SERVER_TIMEOUT,
                CO_SDO_BLOCK_TRANSFER_ENABLE);
            if (ret == CO_SDOcli_ok_communicationEnd)
            {
                /* Write data to SDO buffer */
                if (txSize == CO_SDOclientDownloadBufWrite( SDOclient, dataTx, txSize
            ))
                {
                    result = canopenappData.DownloadResult =
CANOPENAPP_SDO_ClientOpeScheduled;
                    break;
                }
            }
            result = CANOPENAPP_SDO_ClientCANerror;
            break;
        case CANOPENAPP_SDO_ClientOpeResultOK:
        case CANOPENAPP_SDO_ClientCANerror:
        case CANOPENAPP_SDO_ClientArgError:
            result = canopenappData.DownloadResult;
            canopenappData.DownloadResult = CANOPENAPP_SDO_ClientIdle;
        default:
            break;
        }
    return result;
}

/*****
End of File
*/

```

## Fails "CANopenApp.h"

```

/*****
MPLAB Harmony Application Header File

```

```

Company:
  Microchip Technology Inc.

```

```

File Name:
  canopenapp.h

```

```

Summary:
  This header file provides prototypes and definitions for the application.

```

```

Description:
  This header file provides function prototypes and data type definitions for
  the application. Some of these are required by the system (such as the
  "APP_Initialize" and "APP_Tasks" prototypes) and some of them are only used
  internally by the application (such as the "APP_STATES" definition). Both
  are defined here for convenience.

```

```

*****/

```

```

//DOM-IGNORE-BEGIN

```

```

/*****
Copyright (c) 2013-2014 released Microchip Technology Inc. All rights reserved.

```

Microchip licenses to you the right to use, modify, copy and distribute Software only when embedded on a Microchip microcontroller or digital signal controller that is integrated into your product or third party product (pursuant to the sublicense terms in the accompanying license agreement).

You should refer to the license agreement accompanying this Software for additional information regarding your rights and obligations.

SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY, TITLE, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL MICROCHIP OR ITS LICENSORS BE LIABLE OR OBLIGATED UNDER CONTRACT, NEGLIGENCE, STRICT LIABILITY, CONTRIBUTION, BREACH OF WARRANTY, OR OTHER LEGAL EQUITABLE THEORY ANY DIRECT OR INDIRECT DAMAGES OR EXPENSES INCLUDING BUT NOT LIMITED TO ANY INCIDENTAL, SPECIAL, INDIRECT, PUNITIVE OR CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF PROCUREMENT OF SUBSTITUTE GOODS, TECHNOLOGY, SERVICES, OR ANY CLAIMS BY THIRD PARTIES (INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF), OR OTHER SIMILAR COSTS.

```

*****/
//DOM-IGNORE-END

```

```

#ifndef _APP_H
#define _APP_H

```

```

// *****/
// *****/
// Section: Included Files
// *****/
// *****/

```

```

#include <stdint.h>
#include <stdbool.h>
#include <stddef.h>
#include <stdlib.h>
#include "system_config.h"
#include "system_definitions.h"
#include "RONINCommandProc.h"

```

```

// DOM-IGNORE-BEGIN
#ifdef __cplusplus // Provide C++ Compatibility

extern "C" {

#endif
// DOM-IGNORE-END

// *****
// *****
// Section: Type Definitions
// *****
// *****

#define USE_CAN 1

    // *****
/* Application states

Summary:
    Application states enumeration

Description:
    This enumeration defines the valid application states. These states
    determine the behavior of the application at various times.
*/

typedef enum
{
    /* Application's state machine's initial state. */
    APP_STATE_INIT=0,
    APP_STATE_WAITING_CAN_READY,
    APP_STATE_WAIT_IP,
    APP_STATE_SERVICE_TASKS,

    /* TODO: Define states used by the application state machine. */
} APP_STATES;

// *****
/* Application Data

Summary:
    Holds application data

Description:
    This structure holds the application's data.

Remarks:
    Application strings and buffers are be defined outside this structure.
*/

typedef struct
{
    /* The application's current state */
    APP_STATES state;

    /* TODO: Define any additional data used by the application. */

    SYS_TMR_HANDLE Tmr1sHandle;
    SYS_TMR_HANDLE Tmr50msHandle;
    uint8_t ConsoleBuf[100];

```



```
} APP_DATA;
```

```
// *****  
// *****  
// Section: Application Callback Routines  
// *****  
// *****  
/* These routines are called by drivers when certain events occur.  
*/
```

```
// *****  
// *****  
// Section: Application Initialization and State Machine Functions  
// *****  
// *****
```

```
/* *****
```

Function:

```
void APP_Initialize ( void )
```

Summary:

MPLAB Harmony application initialization routine.

Description:

This function initializes the Harmony application. It places the application in its initial state and prepares it to run so that its APP\_Tasks function can be called.

Precondition:

All other system initialization routines should be called before calling this routine (in "SYS\_Initialize").

Parameters:

None.

Returns:

None.

Example:

```
<code>  
APP_Initialize();  
</code>
```

Remarks:

This routine must be called from the SYS\_Initialize function.

```
*/
```

```
void APP_Initialize ( void );
```

```
/* *****
```

Function:

```
void APP_Tasks ( void )
```

Summary:

MPLAB Harmony Demo application tasks function

Description:

This routine is the Harmony Demo application's tasks function. It defines the application's state machine and core logic.

Precondition:

The system and application initialization ("SYS\_Initialize") should be called before calling this.

Parameters:

None.

Returns:

None.

Example:

```
<code>
APP_Tasks();
</code>
```

Remarks:

This routine must be called from SYS\_Tasks() routine.

\*/

```
void APP_Tasks( void );
```

```
#endif /* _APP_H */
```

```
//DOM-IGNORE-BEGIN
```

```
#ifdef __cplusplus
```

```
}
```

```
#endif
```

```
//DOM-IGNORE-END
```

```
/******
```

```
End of File
```

```
*/
```

## Fails "CO\_config.h"

```

/**
 * Configuration macros for CANopenNode.
 *
 * @file          CO_config.h
 * @ingroup       CO_driver
 * @author       Janez Paternoster
 * @copyright    2020 Janez Paternoster
 *
 * This file is part of CANopenNode, an opensource CANopen Stack.
 * Project home page is <https://github.com/CANopenNode/CANopenNode>.
 * For more information on CANopen see <http://www.can-cia.org/>.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

#ifndef CO_CONFIG_FLAGS_H
#define CO_CONFIG_FLAGS_H

#ifdef __cplusplus
extern "C" {
#endif

/**
 * @defgroup CO_STACK_CONFIG Stack configuration
 * @ingroup CO_driver
 *
 * Stack configuration macros specify, which parts of the stack will be enabled.
 *
 * Default values for stack configuration macros are set in CO_driver.h file.
 * They can be overridden by CO_driver_target.h file. If specified so, they can
 * further be overridden by CO_driver_custom.h file.
 *
 * Stack configuration macro is specified as bits, where each bit
 * enables/disables some part of the configurable CANopenNode object. Flags are
 * used for enabling or checking specific bit. Multiple flags can be ORed
 * together.
 *
 * Configuration macros may be in relation with @ref CO_NO_OBJ macros from
 * CANopen.h file. Former enables/disables stack functionalities, latter
 * enables/disables objects in object dictionary. Note that some objects in
 * object dictionary may need some configuration macros to be enabled.
 * @{
 */

/**
 * Enable custom callback after CAN receive
 *
 * Flag enables optional callback functions, which are part of some CANopenNode
 * objects. Callbacks can optionally be registered by application, which
 * configures threads in operating system. Callbacks are called after something

```

```

* has been preprocessed by higher priority thread and must be further
* processed by lower priority thread. For example when CAN message is received
* and preprocessed, callback should wake up mainline processing function.
* See also @ref CO_process() function.
*
* If callback functions are used, they must be initialized separately, after
* the object initialization.
*
* This flag is common to multiple configuration macros.
*/
#define CO_CONFIG_FLAG_CALLBACK_PRE 0x1000

/**
* Enable calculation of timerNext_us variable.
*
* Calculation of the timerNext_us variable is useful for smooth operation on
* operating system. See also @ref CO_process() function.
*
* This flag is common to multiple configuration macros.
*/
#define CO_CONFIG_FLAG_TIMERNEXT 0x2000

/**
* Configuration of NMT_Heartbeat object
*
* Possible flags, can be ORed:
* - #CO_CONFIG_FLAG_CALLBACK_PRE - Enable custom callback after preprocessing
*   received NMT CAN message.
*   Callback is configured by CO_NMT_initCallbackPre().
* - #CO_CONFIG_FLAG_TIMERNEXT - Enable calculation of timerNext_us variable
*   inside CO_NMT_process().
* - CO_CONFIG_NMT_CALLBACK_CHANGE - Enable custom callback after NMT
*   state changes. Callback is configured by
*   CO_NMT_initCallbackChanged().
* - CO_CONFIG_NMT_MASTER - Enable simple NMT master
*/
#ifdef CO_DOXYGEN
#define CO_CONFIG_NMT (CO_CONFIG_FLAG_CALLBACK_PRE | CO_CONFIG_FLAG_TIMERNEXT |
CO_CONFIG_NMT_CALLBACK_CHANGE | CO_CONFIG_NMT_MASTER)
#endif
#define CO_CONFIG_NMT_CALLBACK_CHANGE 0x01
#define CO_CONFIG_NMT_MASTER 0x02

/**
* Configuration of SDO server object
*
* Possible flags, can be ORed:
* - #CO_CONFIG_FLAG_CALLBACK_PRE - Enable custom callback after preprocessing
*   received SDO CAN message.
*   Callback is configured by CO_SDO_initCallbackPre().
* - #CO_CONFIG_FLAG_TIMERNEXT - Enable calculation of timerNext_us variable
*   inside CO_SDO_process().
* - CO_CONFIG_SDO_SEGMENTED - Enable SDO server segmented transfer.
* - CO_CONFIG_SDO_BLOCK - Enable SDO server block transfer. If set, then
*   CO_CONFIG_SDO_SEGMENTED must also be set.
*/
#ifdef CO_DOXYGEN
#define CO_CONFIG_SDO (CO_CONFIG_FLAG_CALLBACK_PRE | CO_CONFIG_FLAG_TIMERNEXT |
CO_CONFIG_SDO_SEGMENTED | CO_CONFIG_SDO_BLOCK)
#endif
/* TODO with new OD */

```

```

#define CO_CONFIG_SDO_SEGMENTED 0x01
#define CO_CONFIG_SDO_BLOCK 0x02

/**
 * Size of the internal data buffer for the SDO server.
 *
 * Size must be at least equal to size of largest variable in
 * @ref CO_SDO_objectDictionary. If data type is domain, data length is not
 * limited to SDO buffer size. If block transfer is implemented, value should be
 * set to 889.
 *
 * Value can be in range from 7 to 889 bytes.
 */
#ifdef CO_DOXYGEN
#define CO_CONFIG_SDO_BUFFER_SIZE 32
#endif

/**
 * Configuration of Emergency object
 *
 * Possible flags, can be ORed:
 * - #CO_CONFIG_FLAG_CALLBACK_PRE - Enable custom callback after preprocessing
 *   emergency condition by CO_errorReport() or CO_errorReset() call.
 *   Callback is configured by CO_EM_initCallbackPre().
 * - #CO_CONFIG_FLAG_TIMERNEXT - Enable calculation of timerNext_us variable
 *   inside CO_EM_process().
 * - CO_CONFIG_EM_CONSUMER - Enable emergency consumer.
 */
#ifdef CO_DOXYGEN
#define CO_CONFIG_EM (CO_CONFIG_FLAG_CALLBACK_PRE | CO_CONFIG_FLAG_TIMERNEXT |
CO_CONFIG_EM_CONSUMER)
#endif
#define CO_CONFIG_EM_CONSUMER 0x01

/**
 * Configuration of Heartbeat Consumer
 *
 * Possible flags, can be ORed:
 * - #CO_CONFIG_FLAG_CALLBACK_PRE - Enable custom callback after preprocessing
 *   received heartbeat CAN message.
 *   Callback is configured by CO_HBconsumer_initCallbackPre().
 * - #CO_CONFIG_FLAG_TIMERNEXT - Enable calculation of timerNext_us variable
 *   inside CO_HBconsumer_process().
 * - CO_CONFIG_HB_CONS_CALLBACK_CHANGE - Enable custom callback after NMT
 *   state of the monitored node changes. Callback is configured by
 *   CO_HBconsumer_initCallbackNmtChanged().
 * - CO_CONFIG_HB_CONS_CALLBACK_MULTI - Enable multiple custom callbacks, which
 *   can be configured for each monitored node. Callback are configured by
 *   CO_HBconsumer_initCallbackHeartbeatStarted(),
 *   CO_HBconsumer_initCallbackTimeout() and
 *   CO_HBconsumer_initCallbackRemoteReset() functions.
 * - CO_CONFIG_HB_CONS_QUERY_FUNCT - Enable functions for query HB state or
 *   NMT state of the specific monitored node.
 */
#ifdef CO_DOXYGEN
#define CO_CONFIG_HB_CONS (CO_CONFIG_FLAG_CALLBACK_PRE | CO_CONFIG_FLAG_TIMERNEXT |
CO_CONFIG_HB_CONS_CALLBACK_CHANGE | CO_CONFIG_HB_CONS_CALLBACK_MULTI |
CO_CONFIG_HB_CONS_QUERY_FUNCT)
#endif
#define CO_CONFIG_HB_CONS_CALLBACK_CHANGE 0x01
#define CO_CONFIG_HB_CONS_CALLBACK_MULTI 0x02

```

```

#define CO_CONFIG_HB_CONS_QUERY_FUNCT 0x04

/**
 * Configuration of GFC
 *
 * Possible flags, can be ORed:
 * - CO_CONFIG_GFC_CONSUMER - Enable the GFC consumer
 * - CO_CONFIG_GFC_PRODUCER - Enable the GFC producer
 */
#ifdef CO_DOXYGEN
#define CO_CONFIG_GFC (CO_CONFIG_GFC_CONSUMER | CO_CONFIG_GFC_PRODUCER)
#endif
#define CO_CONFIG_GFC_CONSUMER 0x00
#define CO_CONFIG_GFC_PRODUCER 0x00

/**
 * Configuration of SRDO
 *
 * Possible flags, can be ORed:
 * - #CO_CONFIG_FLAG_CALLBACK_PRE - Enable custom callback after preprocessing
 *   received RSRDO CAN message.
 *   Callback is configured by CO_SRDO_initCallbackPre().
 * - #CO_CONFIG_FLAG_TIMERNEXT - Enable calculation of timerNext_us variable
 *   inside CO_SRDO_process() (Tx SRDO only).
 * - CO_CONFIG_RSRDO_CALLS_EXTENSION - Enable calling configured extension
 *   callbacks when received RSRDO CAN message modifies OD entries.
 * - CO_CONFIG_TRSRDO_CALLS_EXTENSION - Enable calling configured extension
 *   callbacks before TSRDO CAN message is sent.
 */
#ifdef CO_DOXYGEN
#define CO_CONFIG_SRDO (CO_CONFIG_FLAG_CALLBACK_PRE | CO_CONFIG_FLAG_TIMERNEXT |
CO_CONFIG_SRDO_CHECK_TX | CO_CONFIG_RSRDO_CALLS_EXTENSION |
CO_CONFIG_TSRDO_CALLS_EXTENSION)
#endif
#define CO_CONFIG_SRDO_CHECK_TX 0x0
#define CO_CONFIG_RSRDO_CALLS_EXTENSION 0x02
#define CO_CONFIG_TSRDO_CALLS_EXTENSION 0x04

/**
 * SRDO Tx time delay
 *
 * minimum time between the first and second SRDO (Tx) message
 * in us
 */
#ifdef CO_DOXYGEN
#define CO_CONFIG_SRDO_MINIMUM_DELAY 0
#endif

/**
 * Configuration of PDO
 *
 * Possible flags, can be ORed:
 * - #CO_CONFIG_FLAG_CALLBACK_PRE - Enable custom callback after preprocessing
 *   received RPDO CAN message.
 *   Callback is configured by CO_RPDO_initCallbackPre().
 * - #CO_CONFIG_FLAG_TIMERNEXT - Enable calculation of timerNext_us variable
 *   inside CO_TPDO_process().
 * - CO_CONFIG_PDO_SYNC_ENABLE - Enable SYNC object inside PDO objects.
 * - CO_CONFIG_RPDO_CALLS_EXTENSION - Enable calling configured extension
 *   callbacks when received RPDO CAN message modifies OD entries.
 * - CO_CONFIG_TPDO_CALLS_EXTENSION - Enable calling configured extension
 *   callbacks before TPDO CAN message is sent.
 */
#ifdef CO_DOXYGEN

```

```

#define CO_CONFIG_PDO (CO_CONFIG_FLAG_CALLBACK_PRE | CO_CONFIG_FLAG_TIMERNEXT |
CO_CONFIG_PDO_SYNC_ENABLE | CO_CONFIG_RPDO_CALLS_EXTENSION |
CO_CONFIG_TPDO_CALLS_EXTENSION)
#endif
#define CO_CONFIG_PDO_SYNC_ENABLE 0x00
#define CO_CONFIG_RPDO_CALLS_EXTENSION 0x02
#define CO_CONFIG_TPDO_CALLS_EXTENSION 0x04

/**
 * Configuration of SYNC
 *
 * Possible flags, can be ORed:
 * - #CO_CONFIG_FLAG_CALLBACK_PRE - Enable custom callback after preprocessing
 *   received SYNC CAN message.
 *   Callback is configured by CO_SYNC_initCallbackPre().
 * - #CO_CONFIG_FLAG_TIMERNEXT - Enable calculation of timerNext_us variable
 *   inside CO_SYNC_process().
 */
#ifdef CO_DOXYGEN
#define CO_CONFIG_SYNC (CO_CONFIG_FLAG_CALLBACK_PRE | CO_CONFIG_FLAG_TIMERNEXT)
#endif

/**
 * Configuration of SDO client object
 *
 * Possible flags, can be ORed:
 * - #CO_CONFIG_FLAG_CALLBACK_PRE - Enable custom callback after preprocessing
 *   received SDO CAN message.
 *   Callback is configured by CO_SDOclient_initCallbackPre().
 * - #CO_CONFIG_FLAG_TIMERNEXT - Enable calculation of timerNext_us variable
 *   inside CO_SDOclientDownloadInitiate(), CO_SDOclientDownload(),
 *   CO_SDOclientUploadInitiate(), CO_SDOclientUpload().
 * - CO_CONFIG_SDO_CLI_SEGMENTED - Enable SDO client segmented transfer.
 * - CO_CONFIG_SDO_CLI_BLOCK - Enable SDO client block transfer. If set, then
 *   CO_CONFIG_SDO_CLI_SEGMENTED must also be set.
 * - CO_CONFIG_SDO_CLI_LOCAL - Enable local transfer, if Node-ID of the SDO
 *   server is the same as node-ID of the SDO client. (SDO client is the same
 *   device as SDO server.) Transfer data directly without communication on CAN.
 */
#ifdef CO_DOXYGEN
#define CO_CONFIG_SDO_CLI (CO_CONFIG_FLAG_CALLBACK_PRE | CO_CONFIG_FLAG_TIMERNEXT |
CO_CONFIG_SDO_CLI_SEGMENTED | CO_CONFIG_SDO_CLI_BLOCK | CO_CONFIG_SDO_CLI_LOCAL)
#endif
#define CO_CONFIG_SDO_CLI_SEGMENTED 0x01
#define CO_CONFIG_SDO_CLI_BLOCK 0x02
#define CO_CONFIG_SDO_CLI_LOCAL 0x04
#define CO_CONFIG_SDO_CLI (CO_CONFIG_SDO_CLI_SEGMENTED | CO_CONFIG_SDO_CLI_BLOCK |
CO_CONFIG_SDO_CLI_LOCAL)

/**
 * Size of the internal data buffer for the SDO client.
 *
 * Circular buffer is used for SDO communication. it can be read or written
 * between successive SDO calls. So size of the buffer can be lower than size of
 * the actual size of data transferred. If only segmented transfer is used, then
 * buffer size can be as low as 7 bytes, if data are read/written each cycle. If
 * block transfer is used, buffer size should be set to at least 889 bytes, so
 * maximum blksize can be used. In that case data should be read/written proper
 * time between cycles.
 */
#ifdef CO_DOXYGEN
#define CO_CONFIG_SDO_CLI_BUFFER_SIZE 32

```

```
#endif
```

```
/**  
 * Configuration of TIME  
 *  
 * Possible flags, can be ORed:  
 * - #CO_CONFIG_FLAG_CALLBACK_PRE - Enable custom callback after preprocessing  
 *   received TIME CAN message.  
 *   Callback is configured by CO_TIME_initCallbackPre().  
 */  
#ifndef CO_DOXYGEN  
#define CO_CONFIG_TIME (CO_CONFIG_FLAG_CALLBACK_PRE)  
#endif
```

```
/**  
 * Configuration of LEDs object  
 *  
 * Possible flags, can be ORed:  
 * - #CO_CONFIG_FLAG_TIMERNEXT - Enable calculation of timerNext_us variable  
 *   inside CO_NMT_process().  
 * - CO_CONFIG_LEDS_ENABLE - Enable calculation of the CANopen LED indicators.  
 */  
#ifndef CO_DOXYGEN  
#define CO_CONFIG_LEDS (CO_CONFIG_FLAG_TIMERNEXT | CO_CONFIG_LEDS_ENABLE)  
#endif  
#define CO_CONFIG_LEDS_ENABLE 0x00
```

```
/**  
 * Configuration of LSS objects  
 *  
 * Possible flags, can be ORed:  
 * - #CO_CONFIG_FLAG_CALLBACK_PRE - Enable custom callback after preprocessing  
 *   received CAN message.  
 *   Callback is configured by CO_LSSmaster_initCallbackPre().  
 * - CO_CONFIG_LSS_SLAVE - Enable LSS slave  
 * - CO_CONFIG_LSS_SLAVE_FASTSCAN_DIRECT_RESPOND - Send LSS fastscan respond  
 *   directly from CO_LSSslave_receive() function.  
 * - CO_CONFIG_LSS_MASTER - Enable LSS master  
 */  
#ifndef CO_DOXYGEN  
#define CO_CONFIG_LSS (CO_CONFIG_FLAG_CALLBACK_PRE | CO_CONFIG_LSS_SLAVE |  
CO_CONFIG_LSS_SLAVE_FASTSCAN_DIRECT_RESPOND | CO_CONFIG_LSS_MASTER )  
#endif  
#define CO_CONFIG_LSS_SLAVE 0x00  
#define CO_CONFIG_LSS_SLAVE_FASTSCAN_DIRECT_RESPOND 0x02  
#define CO_CONFIG_LSS_MASTER 0x00
```

```
/**  
 * Configuration of gateway object usage.  
 *  
 * Gateway object is covered by standard CiA 309 - CANopen access from other  
 * networks. It enables usage of the NMT master, SDO client and LSS master as a  
 * gateway device.  
 *  
 * Possible flags, can be ORed:  
 * - CO_CONFIG_GTW_MULTI_NET - Enable multiple network interfaces in gateway  
 *   device. This functionality is currently not implemented.  
 * - CO_CONFIG_GTW_ASCII - Enable gateway device with ASCII mapping (CiA 309-3)  
 * - CO_CONFIG_GTW_ASCII_SDO - Enable SDO client  
 * - CO_CONFIG_GTW_ASCII_NMT - Enable NMT master
```



```

* - CO_CONFIG_GTW_ASCII_LSS - Enable LSS master
* - CO_CONFIG_GTW_ASCII_LOG - Enable non-standard message log read
* - CO_CONFIG_GTW_ASCII_ERROR_DESC - Print error description as additional
*   comments in gateway-ascii device for SDO and gateway errors.
* - CO_CONFIG_GTW_ASCII_PRINT_HELP - use non-standard command "help" to print
*   help usage.
* - CO_CONFIG_GTW_ASCII_PRINT_LEDS - Display "red" and "green" CANopen status
*   LED diodes on terminal.
*/
#ifdef CO_DOXYGEN
#define CO_CONFIG_GTW (CO_CONFIG_GTW_MULTI_NET | CO_CONFIG_GTW_ASCII |
CO_CONFIG_GTW_ASCII_SDO | CO_CONFIG_GTW_ASCII_NMT | CO_CONFIG_GTW_ASCII_LSS |
CO_CONFIG_GTW_ASCII_LOG | CO_CONFIG_GTW_ASCII_ERROR_DESC | CO_CONFIG_GTW_ASCII_PRINT_HELP
| CO_CONFIG_GTW_ASCII_PRINT_LEDS)
#endif
#define CO_CONFIG_GTW_MULTI_NET 0x01
#define CO_CONFIG_GTW_ASCII 0x02
#define CO_CONFIG_GTW_ASCII_SDO 0x04
#define CO_CONFIG_GTW_ASCII_NMT 0x08
#define CO_CONFIG_GTW_ASCII_LSS 0x10
#define CO_CONFIG_GTW_ASCII_LOG 0x20
#define CO_CONFIG_GTW_ASCII_ERROR_DESC 0x40
#define CO_CONFIG_GTW_ASCII_PRINT_HELP 0x80
#define CO_CONFIG_GTW_ASCII_PRINT_LEDS 0x100

/**
 * Number of loops of #CO_SDOclientDownload() in case of block download
 *
 * If SDO client has block download in progress and OS has buffer for CAN tx
 * messages, then #CO_SDOclientDownload() functionion can be called multiple
 * times within own loop (up to 127). This can speed-up SDO block transfer.
 */
#ifdef CO_DOXYGEN
#define CO_CONFIG_GTW_BLOCK_DL_LOOP 1
#endif

/**
 * Size of command buffer in ASCII gateway object.
 *
 * If large amount of data is transferred (block transfer), then this should be
 * increased to 1000 or more. Buffer may be refilled between the block transfer.
 */
#ifdef CO_DOXYGEN
#define CO_CONFIG_GTWA_COMM_BUF_SIZE 200
#endif

/**
 * Size of message log buffer in ASCII gateway object.
 */
#ifdef CO_DOXYGEN
#define CO_CONFIG_GTWA_LOG_BUF_SIZE 2000
#endif

/** @} */

#ifdef __cplusplus
}
#endif /* __cplusplus */

#endif /* CO_CONFIG_FLAGS_H */

```



## Fails "CO\_driver\_custom.h"

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

#include "system_config.h"

#define TMR_TASK_INTERVAL 1000
#define CO_FSYS SYS_CLK_BUS_PERIPHERAL_5 / 1000ul
#define CANAAPP_BITRATE 125//125//1000//125

/*
 * SDO server timeout in ms
 */
#define CO_SDO_SERVER_TIMEOUT 20

/*
 * Enable CO block transfer in CANopenApp SDO client upload/download
 */
#define CO_SDO_BLOCK_TRANSFER_ENABLE false
```

## Fails "CO\_driver\_target.h"

```

/*
 * Microchip PIC32MX specific definitions for CANOpenNode.
 *
 * @file      CO_driver_target.h
 * @author    Janez Paternoster
 * @copyright 2004 - 2020 Janez Paternoster
 *
 * This file is part of CANOpenNode, an opensource CANopen Stack.
 * Project home page is <https://github.com/CANopenNode/CANopenNode>.
 * For more information on CANopen see <http://www.can-cia.org/>.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

#ifndef CO_DRIVER_TARGET
#define CO_DRIVER_TARGET

/* This file contains device and application specific definitions.
 * It is included from CO_driver.h, which contains documentation
 * for definitions below. */

#include <p32xxxx.h>
#include <stddef.h>
#include <stdbool.h>
#include <stdint.h>

#define CO_DRIVER_CUSTOM

#ifdef CO_DRIVER_CUSTOM
#include "CO_driver_custom.h"
#endif

#ifdef __cplusplus
extern "C" {
#endif

/* Stack configuration override from CO_driver.h.
 * For more information see file CO_config.h. */
#ifndef CO_CONFIG_NMT
#define CO_CONFIG_NMT CO_CONFIG_NMT_MASTER
#endif

#ifndef CO_CONFIG_SDO_BUFFER_SIZE
#define CO_CONFIG_SDO_BUFFER_SIZE 950
#endif

/* Basic definitions */
#define CO_LITTLE_ENDIAN
/* NULL is defined in stddef.h */
/* true and false are defined in stdbool.h */
/* int8_t to uint64_t are defined in stdint.h */

```

```

typedef unsigned char          bool_t;
typedef float                 float32_t;
typedef long double           float64_t;
typedef char                   char_t;
typedef unsigned char         oChar_t;
typedef unsigned char         domain_t;

/* CAN receive message structure as aligned in CAN module. */
typedef struct {
    unsigned    ident    :11; /* Standard Identifier */
    unsigned    FILHIT   :5;  /* Filter hit, see PIC32MX documentation */
    unsigned    CMSGTS   :16; /* CAN message timestamp, see PIC32MX documentation */
    unsigned    DLC      :4;  /* Data length code (bits 0...3) */
    unsigned    :5;
    unsigned    RTR      :1;  /* Remote Transmission Request bit */
    unsigned    :22;
    uint8_t     data[8]; /* 8 data bytes */
} CO_CANrxMsg_t;

/* Access to received CAN message */
#define CO_CANrxMsg_readIdent(msg) ((uint16_t)(((CO_CANrxMsg_t *) (msg))->ident))
#define CO_CANrxMsg_readDLC(msg)   ((uint8_t)(((CO_CANrxMsg_t *) (msg))->DLC))
#define CO_CANrxMsg_readData(msg)  ((uint8_t *)(((CO_CANrxMsg_t *) (msg))->data))

/* Received message object */
typedef struct {
    uint16_t ident;
    uint16_t mask;
    void *object;
    void (*CANrx_callback)(void *object, void *message);
} CO_CANrx_t;

/* Transmit message object */
typedef struct {
    uint32_t CMSGSID; /* Equal to register in transmit message buffer. Includes
standard Identifier */
    uint32_t CMSGEID; /* Equal to register in transmit message buffer. Includes data
length code and RTR */
    uint8_t data[8];
    volatile bool_t bufferFull;
    volatile bool_t syncFlag;
} CO_CANTx_t;

/* CAN module object */
typedef struct {
    void *CANptr;
    CO_CANrxMsg_t CANmsgBuff[33]; /* PIC32 specific: CAN message buffer for CAN module. 32
buffers for receive, 1 buffer for transmit */
    uint8_t CANmsgBuffSize; /* PIC32 specific: Size of the above buffer == 33. Take care
initial value! */
    CO_CANrx_t *rxArray;
    uint16_t rxSize;
    CO_CANTx_t *txArray;
    uint16_t txSize;
    uint16_t CANerrorStatus;
    volatile bool_t CANnormal;
    volatile bool_t useCANrxFilters;
    volatile bool_t bufferInhibitFlag;
    volatile bool_t firstCANTxMessage;
    volatile uint16_t CANTxCount;
    uint32_t errOld;
} CO_CANmodule_t;

```

```

/* (un)lock critical section in CO_CANsend() */
extern unsigned int CO_interruptStatus;
#define CO_LOCK_CAN_SEND()      CO_interruptStatus = __builtin_disable_interrupts()
#define CO_UNLOCK_CAN_SEND()   if(CO_interruptStatus & 0x00000001)
{__builtin_enable_interrupts();}

/* (un)lock critical section in CO_errorReport() or CO_errorReset() */
#define CO_LOCK_EMCY()         CO_interruptStatus = __builtin_disable_interrupts()
#define CO_UNLOCK_EMCY()      if(CO_interruptStatus & 0x00000001)
{__builtin_enable_interrupts();}

/* (un)lock critical section when accessing Object Dictionary */
#define CO_LOCK_OD()           CO_interruptStatus = __builtin_disable_interrupts()
#define CO_UNLOCK_OD()        if(CO_interruptStatus & 0x00000001)
{__builtin_enable_interrupts();}

/* Synchronization between CAN receive and message processing threads. */
#define CO_MemoryBarrier()
#define CO_FLAG_READ(rxNew) ((rxNew) != NULL)
#define CO_FLAG_SET(rxNew) {CO_MemoryBarrier(); rxNew = (void*)1L;}
#define CO_FLAG_CLEAR(rxNew) {CO_MemoryBarrier(); rxNew = NULL;}

/* Translate a kernel virtual address in KSEG0 or KSEG1 to a real
 * physical address and back. */
typedef unsigned long CO_paddr_t; /* a physical address */
typedef unsigned long CO_vaddr_t; /* a virtual address */
#define CO_KVA_TO_PA(v)        ((CO_paddr_t)(v) & 0x1fffffff)
#define CO_PA_TO_KVA0(pa)     ((void *) ((pa) | 0x80000000))
#define CO_PA_TO_KVA1(pa)     ((void *) ((pa) | 0xa0000000))

/* CAN bit rates
 *
 * CAN bit rates are initializers for array of eight CO_CANbitRateData_t
 * objects.
 *
 * Macros are not used by driver itself, they may be used by application with
 * combination with object CO_CANbitRateData_t.
 * Application must declare following global variable depending on CO_FSYS used:
 * const CO_CANbitRateData_t CO_CANbitRateData[8] = {CO_CANbitRateDataInitializers};
 *
 * There are initializers for eight objects, which corresponds to following
 * CAN bit rates (in kbps): 10, 20, 50, 125, 250, 500, 800, 1000.
 *
 * CO_FSYS is internal instruction cycle clock frequency in kHz units. See
 * PIC32MX documentation for more information on FSYS.
 *
 * Available values for FSYS:
 * kbps = | 10 | 20 | 50 | 125 | 250 | 500 | 800 | 1000
 * -----+-----+-----+-----+-----+-----+-----+-----+-----
 * 4 Mhz  | 0 | 0 | 0 | 0 | p | - | - | -
 * 8 Mhz  | 0 | 0 | 0 | 0 | 0 | p | - | -
 * 12 Mhz | 0 | 0 | 0 | 0 | p | p | - | -
 * 16 Mhz | 0 | 0 | 0 | 0 | 0 | 0 | p | p
 * 20 Mhz | 0 | 0 | 0 | 0 | 0 | 0 | - | p
 * 24 Mhz | 0 | 0 | 0 | 0 | 0 | p | 0 | p
 * 32 Mhz | p | 0 | 0 | 0 | 0 | 0 | p | 0
 * 36 Mhz | - | 0 | 0 | 0 | 0 | 0 | - | 0
 * 40 Mhz | - | 0 | 0 | 0 | 0 | 0 | p | 0
 * 48 Mhz | - | 0 | 0 | 0 | 0 | 0 | 0 | p
 * 56 Mhz | - | p | 0 | 0 | 0 | p | (p) | p
 * 64 Mhz | - | p | 0 | 0 | 0 | 0 | 0 | 0

```

```

*   72 Mhz | - | - | 0 | 0 | 0 | 0 | 0 | 0
*   80 Mhz | - | - | 0 | 0 | 0 | 0 | p | 0
*
* -----
*   (O=optimal; p=possible; -=not possible)
*/
#ifndef CO_FSYS
/* Macros, which divides K into (SJW + PROP + PhSeg1 + PhSeg2) */
#define TQ_x_7      1, 2, 3, 1
#define TQ_x_8      1, 2, 3, 2
#define TQ_x_9      1, 2, 4, 2
#define TQ_x_10     1, 3, 4, 2
#define TQ_x_12     1, 3, 6, 2
#define TQ_x_14     1, 4, 7, 2
#define TQ_x_15     1, 4, 8, 2 /* good timing */
#define TQ_x_16     1, 5, 8, 2 /* good timing */
#define TQ_x_17     1, 6, 8, 2 /* good timing */
#define TQ_x_18     1, 7, 8, 2 /* good timing */
#define TQ_x_19     1, 8, 8, 2 /* good timing */
#define TQ_x_20     1, 8, 8, 3 /* good timing */
#define TQ_x_21     1, 8, 8, 4
#define TQ_x_22     1, 8, 8, 5
#define TQ_x_23     1, 8, 8, 6
#define TQ_x_24     1, 8, 8, 7
#define TQ_x_25     1, 8, 8, 8

#if CO_FSYS == 4000
#define CO_CANbitRateDataInitializers \
{10, TQ_x_20, 10}, \
{5, TQ_x_20, 20}, \
{2, TQ_x_20, 50}, \
{1, TQ_x_16, 125}, \
{1, TQ_x_8, 250}, \
{1, TQ_x_8, 0}, \
{1, TQ_x_8, 0}, \
{1, TQ_x_8, 0}
#elif CO_FSYS == 8000
#define CO_CANbitRateDataInitializers \
{25, TQ_x_16, 10}, \
{10, TQ_x_20, 20}, \
{5, TQ_x_16, 50}, \
{2, TQ_x_16, 125}, \
{1, TQ_x_16, 250}, \
{1, TQ_x_8, 500}, \
{1, TQ_x_8, 0}, \
{1, TQ_x_8, 0}
#elif CO_FSYS == 12000
#define CO_CANbitRateDataInitializers \
{40, TQ_x_15, 10}, \
{20, TQ_x_15, 20}, \
{8, TQ_x_15, 50}, \
{3, TQ_x_16, 125}, \
{2, TQ_x_12, 250}, \
{1, TQ_x_12, 500}, \
{1, TQ_x_12, 0}, \
{1, TQ_x_12, 0}
#elif CO_FSYS == 16000
#define CO_CANbitRateDataInitializers \
{50, TQ_x_16, 10}, \
{25, TQ_x_16, 20}, \
{10, TQ_x_16, 50}, \
{4, TQ_x_16, 125}, \
{2, TQ_x_16, 250}, \
{1, TQ_x_16, 500}, \
{1, TQ_x_10, 800}, \

```

```

    {1, TQ_x_8 , 1000}
#elif CO_FSYS == 20000
    #define CO_CANbitRateDataInitializers \
    {50, TQ_x_20, 10}, \
    {25, TQ_x_20, 20}, \
    {10, TQ_x_20, 50}, \
    {5, TQ_x_16, 125}, \
    {2, TQ_x_20, 250}, \
    {1, TQ_x_20, 500}, \
    {1, TQ_x_20, 0}, \
    {1, TQ_x_10, 1000}
#elif CO_FSYS == 24000
    #define CO_CANbitRateDataInitializers \
    {63, TQ_x_19, 10}, \
    {40, TQ_x_15, 20}, \
    {15, TQ_x_16, 50}, \
    {6, TQ_x_16, 125}, \
    {3, TQ_x_16, 250}, \
    {2, TQ_x_12, 500}, \
    {1, TQ_x_15, 800}, \
    {1, TQ_x_12, 1000}
#elif CO_FSYS == 32000
    #define CO_CANbitRateDataInitializers \
    {64, TQ_x_25, 10}, \
    {50, TQ_x_16, 20}, \
    {20, TQ_x_16, 50}, \
    {8, TQ_x_16, 125}, \
    {4, TQ_x_16, 250}, \
    {2, TQ_x_16, 500}, \
    {2, TQ_x_10, 800}, \
    {1, TQ_x_16, 1000}
#elif CO_FSYS == 36000
    #define CO_CANbitRateDataInitializers \
    {50, TQ_x_18, 10}, \
    {50, TQ_x_18, 20}, \
    {20, TQ_x_18, 50}, \
    {8, TQ_x_18, 125}, \
    {4, TQ_x_18, 250}, \
    {2, TQ_x_18, 500}, \
    {2, TQ_x_18, 0}, \
    {1, TQ_x_18, 1000}
#elif CO_FSYS == 40000
    #define CO_CANbitRateDataInitializers \
    {50, TQ_x_20, 0}, \
    {50, TQ_x_20, 20}, \
    {25, TQ_x_16, 50}, \
    {10, TQ_x_16, 125}, \
    {5, TQ_x_16, 250}, \
    {2, TQ_x_20, 500}, \
    {1, TQ_x_25, 800}, \
    {1, TQ_x_20, 1000}
#elif CO_FSYS == 48000
    #define CO_CANbitRateDataInitializers \
    {63, TQ_x_19, 0}, \
    {63, TQ_x_19, 20}, \
    {30, TQ_x_16, 50}, \
    {12, TQ_x_16, 125}, \
    {6, TQ_x_16, 250}, \
    {3, TQ_x_16, 500}, \
    {2, TQ_x_15, 800}, \
    {2, TQ_x_12, 1000}
#elif CO_FSYS == 56000
    #define CO_CANbitRateDataInitializers \
    {61, TQ_x_23, 0}, \

```



```

    {61, TQ_x_23, 20}, \
    {35, TQ_x_16, 50}, \
    {14, TQ_x_16, 125}, \
    {7, TQ_x_16, 250}, \
    {4, TQ_x_14, 500}, \
    {5, TQ_x_7, 800}, \
    {2, TQ_x_14, 1000}
#elif CO_FSYS == 64000
#define CO_CANbitRateDataInitializers \
    {64, TQ_x_25, 0}, \
    {64, TQ_x_25, 20}, \
    {40, TQ_x_16, 50}, \
    {16, TQ_x_16, 125}, \
    {8, TQ_x_16, 250}, \
    {4, TQ_x_16, 500}, \
    {2, TQ_x_20, 800}, \
    {2, TQ_x_16, 1000}
#elif CO_FSYS == 72000
#define CO_CANbitRateDataInitializers \
    {40, TQ_x_18, 0}, \
    {40, TQ_x_18, 0}, \
    {40, TQ_x_18, 50}, \
    {16, TQ_x_18, 125}, \
    {8, TQ_x_18, 250}, \
    {4, TQ_x_18, 500}, \
    {3, TQ_x_15, 800}, \
    {2, TQ_x_18, 1000}
#elif CO_FSYS == 80000
#define CO_CANbitRateDataInitializers \
    {40, TQ_x_20, 0}, \
    {40, TQ_x_20, 0}, \
    {40, TQ_x_20, 50}, \
    {16, TQ_x_20, 125}, \
    {8, TQ_x_20, 250}, \
    {4, TQ_x_20, 500}, \
    {2, TQ_x_25, 800}, \
    {2, TQ_x_20, 1000}
#elif CO_FSYS == 100000
#define CO_CANbitRateDataInitializers \           //Pievienots darbam ar 200 MHz
mikrokontrolleri
    {50, TQ_x_20, 0}, /*Not possible*/ \
    {50, TQ_x_20, 0}, /*Not possible*/ \
    {50, TQ_x_20, 50}, /*CAN=50kbps*/ \
    {20, TQ_x_20, 125}, /*CAN=125kbps*/ \
    {10, TQ_x_20, 250}, /*CAN=250kbps*/ \
    {5, TQ_x_20, 500}, /*CAN=500kbps*/ \
    {2, TQ_x_25, 0}, /*Not possible*/ \
    {2, TQ_x_25, 1000} /*CAN=1000kbps*/
#else
#error define_CO_FSYS CO_FSYS not supported
#endif
#endif

/* Structure contains timing coefficients for CAN module.
*
* CAN baud rate is calculated from following equations:
* Fsys - System clock (MAX 80MHz for PIC32MX)
* TQ = 2 * BRP / Fsys - Time Quanta
* BaudRate = 1 / (TQ * K) - Can bus Baud Rate
* K = SJW + PROP + PhSeg1 + PhSeg2 - Number of Time Quantas
*/
typedef struct {
    uint8_t BRP; /* (1...64) Baud Rate Prescaler */
    uint8_t SJW; /* (1...4) SJW time */

```

```
    uint8_t  PROP;      /* (1...8) PROP time */
    uint8_t  phSeg1;    /* (1...8) Phase Segment 1 time */
    uint8_t  phSeg2;    /* (1...8) Phase Segment 2 time */
    uint16_t bitrate;   /* bitrate in kbps */
} CO_CANbitRateData_t;

#ifdef __cplusplus
}
#endif /* __cplusplus */

#endif /* CO_DRIVER_TARGET */
```

## Fails "CO\_OD.c"

```
// clang-format off
/*****

File - CO_OD.c/CO_OD.h
CANopen Object Dictionary.

This file was automatically generated with libedssharp Object
Dictionary Editor vUnknown DON'T EDIT THIS FILE MANUALLY !!!!
*****/

#include "CO_driver.h"
#include "CO_OD.h"
#include "CO_SDOserver.h"

/*****
DEFINITION AND INITIALIZATION OF OBJECT DICTIONARY VARIABLES
*****/

/***** Definition for ROM variables *****/
struct sCO_OD_ROM CO_OD_ROM = {
    CO_OD_FIRST_LAST_WORD,

    CO_OD_FIRST_LAST_WORD,
};

/***** Definition for RAM variables *****/
struct sCO_OD_RAM CO_OD_RAM = {
    CO_OD_FIRST_LAST_WORD,

/*0005*/ 0x0L,
/*1000*/ 0x0000L,
/*1001*/ 0x0L,
/*1002*/ 0x0000L,
/*1003*/ {0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L},
/*1006*/ 0x3A98L,
/*1007*/ 0x07D0L,
/*1008*/ {'C', 'A', 'N', 'o', 'p', 'e', 'n', 'N', 'o', 'd', 'e'},
/*1009*/ {'3', '.', '0', '0'},
/*100A*/ {'3', '.', '0', '0'},
/*1010*/ {0x0003L},
/*1011*/ {0x0001L},
/*1014*/ 0x0080L,
/*1015*/ 0x64,
/*1016*/ {0x503FFL, 0x603FFL, 0x0000L, 0x0000L},
/*1017*/ 0x00,
/*1018*/ {0x4L, 0x0000L, 0x0000L, 0x0000L, 0x0000L},
/*1019*/ 0x0L,
/*1029*/ {0x0L, 0x0L, 0x1L, 0x0L, 0x0L, 0x0L},
/*1200*/ {{0x2L, 0x0600L, 0x0580L}},
/*1280*/ {{0x3L, 0x0600L, 0x0580L, 0x5L}},
/*1400*/ {{0x2L, 0x0281L, 0xFFL},
/*1401*/ {0x2L, 0x0282L, 0xFFL},
/*1402*/ {0x2L, 0x0283L, 0xFFL},
/*1403*/ {0x2L, 0x0284L, 0xFFL},
/*1404*/ {0x2L, 0x0285L, 0xFFL},
/*1405*/ {0x2L, 0x0286L, 0xFFL},
/*1406*/ {0x2L, 0x028AL, 0xFEL},
/*1407*/ {0x2L, 0x028BL, 0xFFL},
```

```
/*1408*/ {0x2L, 0x028CL, 0xFFL},
/*1409*/ {0x2L, 0x028DL, 0xFFL}},
/*1600*/ {{0x2L, 0x60110010L, 0x60120020L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L,
0x0000L},
/*1601*/ {0x2L, 0x60210010L, 0x60220020L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L,
0x0000L},
/*1602*/ {0x2L, 0x60310010L, 0x60320020L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L,
0x0000L},
/*1603*/ {0x2L, 0x60410010L, 0x60420020L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L,
0x0000L},
/*1604*/ {0x2L, 0x60510010L, 0x60520020L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L,
0x0000L},
/*1605*/ {0x2L, 0x60610010L, 0x60620020L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L,
0x0000L},
/*1606*/ {0x2L, 0x60A10010L, 0x60A30010L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L,
0x0000L},
/*1607*/ {0x2L, 0x60B10010L, 0x60B30010L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L,
0x0000L},
/*1608*/ {0x2L, 0x60C10010L, 0x60C30010L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L,
0x0000L},
/*1609*/ {0x2L, 0x60D10010L, 0x60D30010L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L,
0x0000L}},
/*1800*/ {{0x6L, 0x0301L, 0xFFL, 0x64, 0x0L, 0x00, 0x0L},
/*1801*/ {0x6L, 0x0302L, 0xFFL, 0x00, 0x0L, 0x00, 0x0L},
/*1802*/ {0x6L, 0x0303L, 0xFFL, 0x00, 0x0L, 0x00, 0x0L},
/*1803*/ {0x6L, 0x0304L, 0xFFL, 0x00, 0x0L, 0x00, 0x0L},
/*1804*/ {0x6L, 0x0305L, 0xFFL, 0x00, 0x0L, 0x00, 0x0L},
/*1805*/ {0x6L, 0x0306L, 0xFFL, 0x00, 0x0L, 0x00, 0x0L},
/*1806*/ {0x6L, 0x030AL, 0xFFL, 0x00, 0x0L, 0x00, 0x0L},
/*1807*/ {0x6L, 0x030BL, 0xFFL, 0x00, 0x0L, 0x00, 0x0L},
/*1808*/ {0x6L, 0x030CL, 0xFFL, 0x00, 0x0L, 0x00, 0x0L},
/*1809*/ {0x6L, 0x030DL, 0xFFL, 0x00, 0x0L, 0x00, 0x0L}},
/*1A00*/ {{0x2L, 0x60100010L, 0x60140020L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L,
0x0000L},
/*1A01*/ {0x2L, 0x60200010L, 0x60240020L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L,
0x0000L},
/*1A02*/ {0x2L, 0x60300010L, 0x60340020L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L,
0x0000L},
/*1A03*/ {0x2L, 0x60400010L, 0x60440020L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L,
0x0000L},
/*1A04*/ {0x2L, 0x60500010L, 0x60540020L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L,
0x0000L},
/*1A05*/ {0x2L, 0x60600010L, 0x60640020L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L,
0x0000L},
/*1A06*/ {0x2L, 0x60A00010L, 0x60A20010L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L,
0x0000L},
/*1A07*/ {0x2L, 0x60B00010L, 0x60B20010L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L,
0x0000L},
/*1A08*/ {0x2L, 0x60C00010L, 0x60C20010L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L,
0x0000L},
/*1A09*/ {0x2L, 0x60D00010L, 0x60D20010L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L,
0x0000L}},
/*1F80*/ 0x0001L,
/*2100*/ {0x0L},
/*2101*/ 0x30L,
/*2102*/ 0xFA,
/*2103*/ 0x00,
/*2104*/ 0x00,
/*2106*/ 0x0000L,
/*2107*/ {0x3E8, 0x00, 0x00, 0x00, 0x00},
/*2108*/ {0x00},
/*2109*/ {0x00},
/*2110*/ {0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L,
0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L},
```

```
/*2111*/ {0x0001L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L,
0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L},
/*2112*/ {0x0001L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L,
0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0x0000L},
/*2120*/ {0x5L, 0x1234567890ABCDEF1L, 0x234567890ABCDEF1L, 12.345, 456.789, 0},
/*2130*/ {0x3L, {'-'}, 0x00000000L, 0x0000L},
/*2301*/ {{0x8L, 0x0064L, 0x1L, {'T', 'r', 'a', 'c', 'e', 'l'}, {'r', 'e', 'd'},
0x60000108L, 0x1L, 0x0L, 0x0000L},
/*2302*/ {0x8L, 0x0000L, 0x0L, {'T', 'r', 'a', 'c', 'e', '2'}, {'g', 'r', 'e', 'e', 'n'},
0x0000L, 0x0L, 0x0L, 0x0000L}},
/*2400*/ 0x0L,
/*2401*/ {{0x6L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0, 0x0000L},
/*2402*/ {0x6L, 0x0000L, 0x0000L, 0x0000L, 0x0000L, 0, 0x0000L}},
/*6010*/ 0x00,
/*6011*/ 0x00,
/*6012*/ 0x0000L,
/*6013*/ 0x0000L,
/*6014*/ 0x0000L,
/*6015*/ 0x0000L,
/*6020*/ 0x00,
/*6021*/ 0x00,
/*6022*/ 0x0000L,
/*6023*/ 0x0000L,
/*6024*/ 0x0000L,
/*6025*/ 0x0000L,
/*6030*/ 0x00,
/*6031*/ 0x00,
/*6032*/ 0x0000L,
/*6033*/ 0x0000L,
/*6034*/ 0x0000L,
/*6035*/ 0x0000L,
/*6040*/ 0x00,
/*6041*/ 0x00,
/*6042*/ 0x0000L,
/*6043*/ 0x0000L,
/*6044*/ 0x0000L,
/*6045*/ 0x0000L,
/*6050*/ 0x00,
/*6051*/ 0x00,
/*6052*/ 0x0000L,
/*6053*/ 0x0000L,
/*6054*/ 0x0000L,
/*6055*/ 0x0000L,
/*6060*/ 0x00,
/*6061*/ 0x00,
/*6062*/ 0x0000L,
/*6063*/ 0x0000L,
/*6064*/ 0x0000L,
/*6065*/ 0x0000L,
/*60A0*/ 0x00,
/*60A1*/ 0x00,
/*60A2*/ 0x00,
/*60A3*/ 0x00,
/*60B0*/ 0x00,
/*60B1*/ 0x00,
/*60B2*/ 0x00,
/*60B3*/ 0x00,
/*60C0*/ 0x00,
/*60C1*/ 0x00,
/*60C2*/ 0x00,
/*60C3*/ 0x00,
/*60D0*/ 0x00,
/*60D1*/ 0x00,
/*60D2*/ 0x00,
```

```

/*60D3*/ 0x00,

        CO_OD_FIRST_LAST_WORD,
};

/***** Definition for EEPROM variables *****/
struct sCO_OD_EEPROM CO_OD_EEPROM = {
        CO_OD_FIRST_LAST_WORD,

        CO_OD_FIRST_LAST_WORD,
};

/*****
STRUCTURES FOR RECORD TYPE OBJECTS
*****/

/*0x1018*/ const CO_OD_entryRecord_t OD_record1018[5] = {
        { (void*)&CO_OD_RAM.identity.maxSubIndex, 0x06, 0x1 },
        { (void*)&CO_OD_RAM.identity.vendorID, 0x86, 0x4 },
        { (void*)&CO_OD_RAM.identity.productCode, 0x86, 0x4 },
        { (void*)&CO_OD_RAM.identity.revisionNumber, 0x86, 0x4 },
        { (void*)&CO_OD_RAM.identity.serialNumber, 0x86, 0x4 },
};

/*0x1200*/ const CO_OD_entryRecord_t OD_record1200[3] = {
        { (void*)&CO_OD_RAM.SDOServerParameter[0].maxSubIndex, 0x06, 0x1 },
        { (void*)&CO_OD_RAM.SDOServerParameter[0].COB_IDClientToServer, 0x86, 0x4 },
        { (void*)&CO_OD_RAM.SDOServerParameter[0].COB_IDServerToClient, 0x86, 0x4 },
};

/*0x1280*/ const CO_OD_entryRecord_t OD_record1280[4] = {
        { (void*)&CO_OD_RAM.SDOClientParameter[0].maxSubIndex, 0x06, 0x1 },
        { (void*)&CO_OD_RAM.SDOClientParameter[0].COB_IDClientToServer, 0x86, 0x4 },
        { (void*)&CO_OD_RAM.SDOClientParameter[0].COB_IDServerToClient, 0x86, 0x4 },
        { (void*)&CO_OD_RAM.SDOClientParameter[0].nodeIDofTheSDOServer, 0x06, 0x1 },
};

/*0x1400*/ const CO_OD_entryRecord_t OD_record1400[3] = {
        { (void*)&CO_OD_RAM.RPDOCommunicationParameter[0].maxSubIndex, 0x06, 0x1 },
        { (void*)&CO_OD_RAM.RPDOCommunicationParameter[0].COB_IDUsedByRPDO, 0x8E, 0x4 },
        { (void*)&CO_OD_RAM.RPDOCommunicationParameter[0].transmissionType, 0x0E, 0x1 },
};

/*0x1401*/ const CO_OD_entryRecord_t OD_record1401[3] = {
        { (void*)&CO_OD_RAM.RPDOCommunicationParameter[1].maxSubIndex, 0x06, 0x1 },
        { (void*)&CO_OD_RAM.RPDOCommunicationParameter[1].COB_IDUsedByRPDO, 0x8E, 0x4 },
        { (void*)&CO_OD_RAM.RPDOCommunicationParameter[1].transmissionType, 0x0E, 0x1 },
};

/*0x1402*/ const CO_OD_entryRecord_t OD_record1402[3] = {
        { (void*)&CO_OD_RAM.RPDOCommunicationParameter[2].maxSubIndex, 0x06, 0x1 },
        { (void*)&CO_OD_RAM.RPDOCommunicationParameter[2].COB_IDUsedByRPDO, 0x8E, 0x4 },
        { (void*)&CO_OD_RAM.RPDOCommunicationParameter[2].transmissionType, 0x0E, 0x1 },
};

/*0x1403*/ const CO_OD_entryRecord_t OD_record1403[3] = {
        { (void*)&CO_OD_RAM.RPDOCommunicationParameter[3].maxSubIndex, 0x06, 0x1 },
        { (void*)&CO_OD_RAM.RPDOCommunicationParameter[3].COB_IDUsedByRPDO, 0x8E, 0x4 },
};

```

```

        { (void*)&CO_OD_RAM.RPDOCommunicationParameter[3].transmissionType, 0x0E, 0x1 },
};

/*0x1404*/ const CO_OD_entryRecord_t OD_record1404[3] = {
    { (void*)&CO_OD_RAM.RPDOCommunicationParameter[4].maxSubIndex, 0x06, 0x1 },
    { (void*)&CO_OD_RAM.RPDOCommunicationParameter[4].COB_IDUsedByRPDO, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.RPDOCommunicationParameter[4].transmissionType, 0x0E, 0x1 },
};

/*0x1405*/ const CO_OD_entryRecord_t OD_record1405[3] = {
    { (void*)&CO_OD_RAM.RPDOCommunicationParameter[5].maxSubIndex, 0x06, 0x1 },
    { (void*)&CO_OD_RAM.RPDOCommunicationParameter[5].COB_IDUsedByRPDO, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.RPDOCommunicationParameter[5].transmissionType, 0x0E, 0x1 },
};

/*0x1406*/ const CO_OD_entryRecord_t OD_record1406[3] = {
    { (void*)&CO_OD_RAM.RPDOCommunicationParameter[6].maxSubIndex, 0x06, 0x1 },
    { (void*)&CO_OD_RAM.RPDOCommunicationParameter[6].COB_IDUsedByRPDO, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.RPDOCommunicationParameter[6].transmissionType, 0x0E, 0x1 },
};

/*0x1407*/ const CO_OD_entryRecord_t OD_record1407[3] = {
    { (void*)&CO_OD_RAM.RPDOCommunicationParameter[7].maxSubIndex, 0x06, 0x1 },
    { (void*)&CO_OD_RAM.RPDOCommunicationParameter[7].COB_IDUsedByRPDO, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.RPDOCommunicationParameter[7].transmissionType, 0x0E, 0x1 },
};

/*0x1408*/ const CO_OD_entryRecord_t OD_record1408[3] = {
    { (void*)&CO_OD_RAM.RPDOCommunicationParameter[8].maxSubIndex, 0x06, 0x1 },
    { (void*)&CO_OD_RAM.RPDOCommunicationParameter[8].COB_IDUsedByRPDO, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.RPDOCommunicationParameter[8].transmissionType, 0x0E, 0x1 },
};

/*0x1409*/ const CO_OD_entryRecord_t OD_record1409[3] = {
    { (void*)&CO_OD_RAM.RPDOCommunicationParameter[9].maxSubIndex, 0x06, 0x1 },
    { (void*)&CO_OD_RAM.RPDOCommunicationParameter[9].COB_IDUsedByRPDO, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.RPDOCommunicationParameter[9].transmissionType, 0x0E, 0x1 },
};

/*0x1600*/ const CO_OD_entryRecord_t OD_record1600[9] = {
    { (void*)&CO_OD_RAM.RPDOMappingParameter[0].numberOfMappedObjects, 0x0E, 0x1 },
    { (void*)&CO_OD_RAM.RPDOMappingParameter[0].mappedObject1, 0xBE, 0x4 },
    { (void*)&CO_OD_RAM.RPDOMappingParameter[0].mappedObject2, 0xBE, 0x4 },
    { (void*)&CO_OD_RAM.RPDOMappingParameter[0].mappedObject3, 0xBE, 0x4 },
    { (void*)&CO_OD_RAM.RPDOMappingParameter[0].mappedObject4, 0xBE, 0x4 },
    { (void*)&CO_OD_RAM.RPDOMappingParameter[0].mappedObject5, 0xBE, 0x4 },
    { (void*)&CO_OD_RAM.RPDOMappingParameter[0].mappedObject6, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.RPDOMappingParameter[0].mappedObject7, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.RPDOMappingParameter[0].mappedObject8, 0x8E, 0x4 },
};

/*0x1601*/ const CO_OD_entryRecord_t OD_record1601[9] = {
    { (void*)&CO_OD_RAM.RPDOMappingParameter[1].numberOfMappedObjects, 0x0E, 0x1 },
    { (void*)&CO_OD_RAM.RPDOMappingParameter[1].mappedObject1, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.RPDOMappingParameter[1].mappedObject2, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.RPDOMappingParameter[1].mappedObject3, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.RPDOMappingParameter[1].mappedObject4, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.RPDOMappingParameter[1].mappedObject5, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.RPDOMappingParameter[1].mappedObject6, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.RPDOMappingParameter[1].mappedObject7, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.RPDOMappingParameter[1].mappedObject8, 0x8E, 0x4 },
};

/*0x1602*/ const CO_OD_entryRecord_t OD_record1602[9] = {

```





```

    { (void*) &CO_OD_RAM.RPDOMappingParameter[7].mappedObject4, 0x86, 0x4 },
    { (void*) &CO_OD_RAM.RPDOMappingParameter[7].mappedObject5, 0x86, 0x4 },
    { (void*) &CO_OD_RAM.RPDOMappingParameter[7].mappedObject6, 0x86, 0x4 },
    { (void*) &CO_OD_RAM.RPDOMappingParameter[7].mappedObject7, 0x86, 0x4 },
    { (void*) &CO_OD_RAM.RPDOMappingParameter[7].mappedObject8, 0x86, 0x4 },
};

/*0x1608*/ const CO_OD_entryRecord_t OD_record1608[9] = {
    { (void*) &CO_OD_RAM.RPDOMappingParameter[8].numberOfMappedObjects, 0x06, 0x1 },
    { (void*) &CO_OD_RAM.RPDOMappingParameter[8].mappedObject1, 0x86, 0x4 },
    { (void*) &CO_OD_RAM.RPDOMappingParameter[8].mappedObject2, 0x86, 0x4 },
    { (void*) &CO_OD_RAM.RPDOMappingParameter[8].mappedObject3, 0x86, 0x4 },
    { (void*) &CO_OD_RAM.RPDOMappingParameter[8].mappedObject4, 0x86, 0x4 },
    { (void*) &CO_OD_RAM.RPDOMappingParameter[8].mappedObject5, 0x86, 0x4 },
    { (void*) &CO_OD_RAM.RPDOMappingParameter[8].mappedObject6, 0x86, 0x4 },
    { (void*) &CO_OD_RAM.RPDOMappingParameter[8].mappedObject7, 0x86, 0x4 },
    { (void*) &CO_OD_RAM.RPDOMappingParameter[8].mappedObject8, 0x86, 0x4 },
};

/*0x1609*/ const CO_OD_entryRecord_t OD_record1609[9] = {
    { (void*) &CO_OD_RAM.RPDOMappingParameter[9].numberOfMappedObjects, 0x06, 0x1 },
    { (void*) &CO_OD_RAM.RPDOMappingParameter[9].mappedObject1, 0x86, 0x4 },
    { (void*) &CO_OD_RAM.RPDOMappingParameter[9].mappedObject2, 0x86, 0x4 },
    { (void*) &CO_OD_RAM.RPDOMappingParameter[9].mappedObject3, 0x86, 0x4 },
    { (void*) &CO_OD_RAM.RPDOMappingParameter[9].mappedObject4, 0x86, 0x4 },
    { (void*) &CO_OD_RAM.RPDOMappingParameter[9].mappedObject5, 0x86, 0x4 },
    { (void*) &CO_OD_RAM.RPDOMappingParameter[9].mappedObject6, 0x86, 0x4 },
    { (void*) &CO_OD_RAM.RPDOMappingParameter[9].mappedObject7, 0x86, 0x4 },
    { (void*) &CO_OD_RAM.RPDOMappingParameter[9].mappedObject8, 0x86, 0x4 },
};

/*0x1800*/ const CO_OD_entryRecord_t OD_record1800[7] = {
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[0].maxSubIndex, 0x06, 0x1 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[0].COB_IDUsedByTPDO, 0x8E, 0x4 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[0].transmissionType, 0x0E, 0x1 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[0].inhibitTime, 0x8E, 0x2 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[0].compatibilityEntry, 0x0E, 0x1
    },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[0].eventTimer, 0x8E, 0x2 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[0].SYNCStartValue, 0x0E, 0x1 },
};

/*0x1801*/ const CO_OD_entryRecord_t OD_record1801[7] = {
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[1].maxSubIndex, 0x06, 0x1 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[1].COB_IDUsedByTPDO, 0x8E, 0x4 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[1].transmissionType, 0x0E, 0x1 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[1].inhibitTime, 0x8E, 0x2 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[1].compatibilityEntry, 0x0E, 0x1
    },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[1].eventTimer, 0x8E, 0x2 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[1].SYNCStartValue, 0x0E, 0x1 },
};

/*0x1802*/ const CO_OD_entryRecord_t OD_record1802[7] = {
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[2].maxSubIndex, 0x06, 0x1 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[2].COB_IDUsedByTPDO, 0x8E, 0x4 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[2].transmissionType, 0x0E, 0x1 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[2].inhibitTime, 0x8E, 0x2 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[2].compatibilityEntry, 0x0E, 0x1
    },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[2].eventTimer, 0x8E, 0x2 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[2].SYNCStartValue, 0x0E, 0x1 },
};

```

```

/*0x1803*/ const CO_OD_entryRecord_t OD_record1803[7] = {
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[3].maxSubIndex, 0x06, 0x1 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[3].COB_IDUsedByTPDO, 0x8E, 0x4 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[3].transmissionType, 0x0E, 0x1 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[3].inhibitTime, 0x8E, 0x2 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[3].compatibilityEntry, 0x0E, 0x1
},
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[3].eventTimer, 0x8E, 0x2 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[3].SYNCStartValue, 0x0E, 0x1 },
};

/*0x1804*/ const CO_OD_entryRecord_t OD_record1804[7] = {
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[4].maxSubIndex, 0x06, 0x1 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[4].COB_IDUsedByTPDO, 0x8E, 0x4 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[4].transmissionType, 0x0E, 0x1 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[4].inhibitTime, 0x8E, 0x2 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[4].compatibilityEntry, 0x06, 0x1
},
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[4].eventTimer, 0x8E, 0x2 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[4].SYNCStartValue, 0x0E, 0x1 },
};

/*0x1805*/ const CO_OD_entryRecord_t OD_record1805[7] = {
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[5].maxSubIndex, 0x06, 0x1 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[5].COB_IDUsedByTPDO, 0x8E, 0x4 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[5].transmissionType, 0x0E, 0x1 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[5].inhibitTime, 0x8E, 0x2 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[5].compatibilityEntry, 0x06, 0x1
},
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[5].eventTimer, 0x8E, 0x2 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[5].SYNCStartValue, 0x0E, 0x1 },
};

/*0x1806*/ const CO_OD_entryRecord_t OD_record1806[7] = {
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[6].maxSubIndex, 0x06, 0x1 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[6].COB_IDUsedByTPDO, 0x8E, 0x4 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[6].transmissionType, 0x0E, 0x1 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[6].inhibitTime, 0x8E, 0x2 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[6].compatibilityEntry, 0x06, 0x1
},
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[6].eventTimer, 0x8E, 0x2 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[6].SYNCStartValue, 0x0E, 0x1 },
};

/*0x1807*/ const CO_OD_entryRecord_t OD_record1807[7] = {
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[7].maxSubIndex, 0x06, 0x1 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[7].COB_IDUsedByTPDO, 0x8E, 0x4 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[7].transmissionType, 0x0E, 0x1 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[7].inhibitTime, 0x8E, 0x2 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[7].compatibilityEntry, 0x06, 0x1
},
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[7].eventTimer, 0x8E, 0x2 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[7].SYNCStartValue, 0x0E, 0x1 },
};

/*0x1808*/ const CO_OD_entryRecord_t OD_record1808[7] = {
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[8].maxSubIndex, 0x06, 0x1 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[8].COB_IDUsedByTPDO, 0x8E, 0x4 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[8].transmissionType, 0x0E, 0x1 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[8].inhibitTime, 0x8E, 0x2 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[8].compatibilityEntry, 0x06, 0x1
},
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[8].eventTimer, 0x8E, 0x2 },
    { (void*) &CO_OD_RAM.TPDOCommunicationParameter[8].SYNCStartValue, 0x0E, 0x1 },
};

```

```

};

/*0x1809*/ const CO_OD_entryRecord_t OD_record1809[7] = {
    { (void*)&CO_OD_RAM.TPDOCommunicationParameter[9].maxSubIndex, 0x06, 0x1 },
    { (void*)&CO_OD_RAM.TPDOCommunicationParameter[9].COB_IDUsedByTPDO, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.TPDOCommunicationParameter[9].transmissionType, 0x0E, 0x1 },
    { (void*)&CO_OD_RAM.TPDOCommunicationParameter[9].inhibitTime, 0x8E, 0x2 },
    { (void*)&CO_OD_RAM.TPDOCommunicationParameter[9].compatibilityEntry, 0x06, 0x1
},
    { (void*)&CO_OD_RAM.TPDOCommunicationParameter[9].eventTimer, 0x8E, 0x2 },
    { (void*)&CO_OD_RAM.TPDOCommunicationParameter[9].SYNCStartValue, 0x0E, 0x1 },
};

/*0x1A00*/ const CO_OD_entryRecord_t OD_record1A00[9] = {
    { (void*)&CO_OD_RAM.TPDOMappingParameter[0].numberOfMappedObjects, 0x0E, 0x1 },
    { (void*)&CO_OD_RAM.TPDOMappingParameter[0].mappedObject1, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.TPDOMappingParameter[0].mappedObject2, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.TPDOMappingParameter[0].mappedObject3, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.TPDOMappingParameter[0].mappedObject4, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.TPDOMappingParameter[0].mappedObject5, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.TPDOMappingParameter[0].mappedObject6, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.TPDOMappingParameter[0].mappedObject7, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.TPDOMappingParameter[0].mappedObject8, 0x8E, 0x4 },
};

/*0x1A01*/ const CO_OD_entryRecord_t OD_record1A01[9] = {
    { (void*)&CO_OD_RAM.TPDOMappingParameter[1].numberOfMappedObjects, 0x0E, 0x1 },
    { (void*)&CO_OD_RAM.TPDOMappingParameter[1].mappedObject1, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.TPDOMappingParameter[1].mappedObject2, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.TPDOMappingParameter[1].mappedObject3, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.TPDOMappingParameter[1].mappedObject4, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.TPDOMappingParameter[1].mappedObject5, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.TPDOMappingParameter[1].mappedObject6, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.TPDOMappingParameter[1].mappedObject7, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.TPDOMappingParameter[1].mappedObject8, 0x8E, 0x4 },
};

/*0x1A02*/ const CO_OD_entryRecord_t OD_record1A02[9] = {
    { (void*)&CO_OD_RAM.TPDOMappingParameter[2].numberOfMappedObjects, 0x0E, 0x1 },
    { (void*)&CO_OD_RAM.TPDOMappingParameter[2].mappedObject1, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.TPDOMappingParameter[2].mappedObject2, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.TPDOMappingParameter[2].mappedObject3, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.TPDOMappingParameter[2].mappedObject4, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.TPDOMappingParameter[2].mappedObject5, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.TPDOMappingParameter[2].mappedObject6, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.TPDOMappingParameter[2].mappedObject7, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.TPDOMappingParameter[2].mappedObject8, 0x8E, 0x4 },
};

/*0x1A03*/ const CO_OD_entryRecord_t OD_record1A03[9] = {
    { (void*)&CO_OD_RAM.TPDOMappingParameter[3].numberOfMappedObjects, 0x0E, 0x1 },
    { (void*)&CO_OD_RAM.TPDOMappingParameter[3].mappedObject1, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.TPDOMappingParameter[3].mappedObject2, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.TPDOMappingParameter[3].mappedObject3, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.TPDOMappingParameter[3].mappedObject4, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.TPDOMappingParameter[3].mappedObject5, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.TPDOMappingParameter[3].mappedObject6, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.TPDOMappingParameter[3].mappedObject7, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.TPDOMappingParameter[3].mappedObject8, 0x8E, 0x4 },
};

/*0x1A04*/ const CO_OD_entryRecord_t OD_record1A04[9] = {
    { (void*)&CO_OD_RAM.TPDOMappingParameter[4].numberOfMappedObjects, 0x06, 0x1 },
    { (void*)&CO_OD_RAM.TPDOMappingParameter[4].mappedObject1, 0x86, 0x4 },
};

```



```

        { (void*)&CO_OD_RAM.TPDOMappingParameter[9].mappedObject6, 0x86, 0x4 },
        { (void*)&CO_OD_RAM.TPDOMappingParameter[9].mappedObject7, 0x86, 0x4 },
        { (void*)&CO_OD_RAM.TPDOMappingParameter[9].mappedObject8, 0x86, 0x4 },
};

/*0x2120*/ const CO_OD_entryRecord_t OD_record2120[6] = {
    { (void*)&CO_OD_RAM.testVar.maxSubIndex, 0x06, 0x1 },
    { (void*)&CO_OD_RAM.testVar.I64, 0x9E, 0x8 },
    { (void*)&CO_OD_RAM.testVar.U64, 0x9E, 0x8 },
    { (void*)&CO_OD_RAM.testVar.R32, 0x9E, 0x4 },
    { (void*)&CO_OD_RAM.testVar.R64, 0x9E, 0x8 },
    { (void*)0, 0x0E, 0x0 },
};

/*0x2130*/ const CO_OD_entryRecord_t OD_record2130[4] = {
    { (void*)&CO_OD_RAM.time.maxSubIndex, 0x06, 0x1 },
    { (void*)&CO_OD_RAM.time.string, 0x06, 0x1 },
    { (void*)&CO_OD_RAM.time.epochTimeBaseMs, 0x8E, 0x8 },
    { (void*)&CO_OD_RAM.time.epochTimeOffsetMs, 0x9E, 0x4 },
};

/*0x2301*/ const CO_OD_entryRecord_t OD_record2301[9] = {
    { (void*)&CO_OD_RAM.traceConfig[0].maxSubIndex, 0x06, 0x1 },
    { (void*)&CO_OD_RAM.traceConfig[0].size, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.traceConfig[0].axisNo, 0x0E, 0x1 },
    { (void*)&CO_OD_RAM.traceConfig[0].name, 0x0E, 0x6 },
    { (void*)&CO_OD_RAM.traceConfig[0].color, 0x0E, 0x3 },
    { (void*)&CO_OD_RAM.traceConfig[0].map, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.traceConfig[0].format, 0x0E, 0x1 },
    { (void*)&CO_OD_RAM.traceConfig[0].trigger, 0x0E, 0x1 },
    { (void*)&CO_OD_RAM.traceConfig[0].threshold, 0x8E, 0x4 },
};

/*0x2302*/ const CO_OD_entryRecord_t OD_record2302[9] = {
    { (void*)&CO_OD_RAM.traceConfig[1].maxSubIndex, 0x06, 0x1 },
    { (void*)&CO_OD_RAM.traceConfig[1].size, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.traceConfig[1].axisNo, 0x0E, 0x1 },
    { (void*)&CO_OD_RAM.traceConfig[1].name, 0x0E, 0x6 },
    { (void*)&CO_OD_RAM.traceConfig[1].color, 0x0E, 0x5 },
    { (void*)&CO_OD_RAM.traceConfig[1].map, 0x8E, 0x4 },
    { (void*)&CO_OD_RAM.traceConfig[1].format, 0x0E, 0x1 },
    { (void*)&CO_OD_RAM.traceConfig[1].trigger, 0x0E, 0x1 },
    { (void*)&CO_OD_RAM.traceConfig[1].threshold, 0x8E, 0x4 },
};

/*0x2401*/ const CO_OD_entryRecord_t OD_record2401[7] = {
    { (void*)&CO_OD_RAM.trace[0].maxSubIndex, 0x06, 0x1 },
    { (void*)&CO_OD_RAM.trace[0].size, 0x9E, 0x4 },
    { (void*)&CO_OD_RAM.trace[0].value, 0x9E, 0x4 },
    { (void*)&CO_OD_RAM.trace[0].min, 0x9E, 0x4 },
    { (void*)&CO_OD_RAM.trace[0].max, 0x9E, 0x4 },
    { (void*)0, 0x06, 0x0 },
    { (void*)&CO_OD_RAM.trace[0].triggerTime, 0x9E, 0x4 },
};

/*0x2402*/ const CO_OD_entryRecord_t OD_record2402[7] = {
    { (void*)&CO_OD_RAM.trace[1].maxSubIndex, 0x06, 0x1 },
    { (void*)&CO_OD_RAM.trace[1].size, 0x9E, 0x4 },
    { (void*)&CO_OD_RAM.trace[1].value, 0x9E, 0x4 },
    { (void*)&CO_OD_RAM.trace[1].min, 0x9E, 0x4 },
    { (void*)&CO_OD_RAM.trace[1].max, 0x9E, 0x4 },
    { (void*)0, 0x06, 0x0 },
    { (void*)&CO_OD_RAM.trace[1].triggerTime, 0x9E, 0x4 },
};

```

/\*\*\*\*\*\*

OBJECT DICTIONARY

\*\*\*\*\*/

```
const CO_OD_entry_t CO_OD[CO_OD_NoOfElements] = {
{0x0005, 0x00, 0x06, 1, (void*)&CO_OD_RAM.compatibilityEntry},
{0x1000, 0x00, 0x86, 4, (void*)&CO_OD_RAM.deviceType},
{0x1001, 0x00, 0x26, 1, (void*)&CO_OD_RAM.errorRegister},
{0x1002, 0x00, 0xA6, 4, (void*)&CO_OD_RAM.manufacturerStatusRegister},
{0x1003, 0x08, 0x06, 0, (void*)&CO_OD_RAM.preDefinedErrorField[0]},
{0x1006, 0x00, 0x8E, 4, (void*)&CO_OD_RAM.communicationCyclePeriod},
{0x1007, 0x00, 0x8E, 4, (void*)&CO_OD_RAM.synchronousWindowLength},
{0x1008, 0x00, 0x06, 11, (void*)&CO_OD_RAM.manufacturerDeviceName},
{0x1009, 0x00, 0x06, 4, (void*)&CO_OD_RAM.manufacturerHardwareVersion},
{0x100A, 0x00, 0x06, 4, (void*)&CO_OD_RAM.manufacturerSoftwareVersion},
{0x1010, 0x01, 0x06, 0, (void*)&CO_OD_RAM.storeParameters[0]},
{0x1011, 0x01, 0x06, 0, (void*)&CO_OD_RAM.restoreDefaultParameters[0]},
{0x1014, 0x00, 0x86, 4, (void*)&CO_OD_RAM.COB_ID_EMCY},
{0x1015, 0x00, 0x8E, 2, (void*)&CO_OD_RAM.inhibitTimeEMCY},
{0x1016, 0x04, 0x06, 0, (void*)&CO_OD_RAM.consumerHeartbeatTime[0]},
{0x1017, 0x00, 0x8E, 2, (void*)&CO_OD_RAM.producerHeartbeatTime},
{0x1018, 0x04, 0x00, 0, (void*)&OD_record1018},
{0x1019, 0x00, 0x0E, 1, (void*)&CO_OD_RAM.synchronousCounterOverflowValue},
{0x1029, 0x06, 0x06, 0, (void*)&CO_OD_RAM.errorBehavior[0]},
{0x1200, 0x02, 0x00, 0, (void*)&OD_record1200},
{0x1280, 0x03, 0x00, 0, (void*)&OD_record1280},
{0x1400, 0x02, 0x00, 0, (void*)&OD_record1400},
{0x1401, 0x02, 0x00, 0, (void*)&OD_record1401},
{0x1402, 0x02, 0x00, 0, (void*)&OD_record1402},
{0x1403, 0x02, 0x00, 0, (void*)&OD_record1403},
{0x1404, 0x02, 0x00, 0, (void*)&OD_record1404},
{0x1405, 0x02, 0x00, 0, (void*)&OD_record1405},
{0x1406, 0x02, 0x00, 4, (void*)&OD_record1406},
{0x1407, 0x02, 0x00, 4, (void*)&OD_record1407},
{0x1408, 0x02, 0x00, 4, (void*)&OD_record1408},
{0x1409, 0x02, 0x00, 4, (void*)&OD_record1409},
{0x1600, 0x08, 0x00, 0, (void*)&OD_record1600},
{0x1601, 0x08, 0x00, 0, (void*)&OD_record1601},
{0x1602, 0x08, 0x00, 0, (void*)&OD_record1602},
{0x1603, 0x08, 0x00, 0, (void*)&OD_record1603},
{0x1604, 0x08, 0x00, 0, (void*)&OD_record1604},
{0x1605, 0x08, 0x00, 0, (void*)&OD_record1605},
{0x1606, 0x08, 0x00, 4, (void*)&OD_record1606},
{0x1607, 0x08, 0x00, 4, (void*)&OD_record1607},
{0x1608, 0x08, 0x00, 4, (void*)&OD_record1608},
{0x1609, 0x08, 0x00, 4, (void*)&OD_record1609},
{0x1800, 0x06, 0x00, 0, (void*)&OD_record1800},
{0x1801, 0x06, 0x00, 0, (void*)&OD_record1801},
{0x1802, 0x06, 0x00, 0, (void*)&OD_record1802},
{0x1803, 0x06, 0x00, 0, (void*)&OD_record1803},
{0x1804, 0x06, 0x00, 0, (void*)&OD_record1804},
{0x1805, 0x06, 0x00, 0, (void*)&OD_record1805},
{0x1806, 0x06, 0x00, 4, (void*)&OD_record1806},
{0x1807, 0x06, 0x00, 4, (void*)&OD_record1807},
{0x1808, 0x06, 0x00, 4, (void*)&OD_record1808},
{0x1809, 0x06, 0x00, 4, (void*)&OD_record1809},
{0x1A00, 0x08, 0x00, 0, (void*)&OD_record1A00},
{0x1A01, 0x08, 0x00, 0, (void*)&OD_record1A01},
{0x1A02, 0x08, 0x00, 0, (void*)&OD_record1A02},
{0x1A03, 0x08, 0x00, 0, (void*)&OD_record1A03},
{0x1A04, 0x08, 0x00, 0, (void*)&OD_record1A04},
{0x1A05, 0x08, 0x00, 0, (void*)&OD_record1A05},
{0x1A06, 0x08, 0x00, 4, (void*)&OD_record1A06},
{0x1A07, 0x08, 0x00, 4, (void*)&OD_record1A07},
```

```
{0x1A08, 0x08, 0x00, 4, (void*)&OD_record1A08},
{0x1A09, 0x08, 0x00, 4, (void*)&OD_record1A09},
{0x1F80, 0x00, 0x8E, 4, (void*)&CO_OD_RAM.NMTStartup},
{0x2100, 0x00, 0x26, 10, (void*)&CO_OD_RAM.errorStatusBits},
{0x2101, 0x00, 0x0E, 1, (void*)&CO_OD_RAM.CANNodeID},
{0x2102, 0x00, 0x8E, 2, (void*)&CO_OD_RAM.CANBitRate},
{0x2103, 0x00, 0x8E, 2, (void*)&CO_OD_RAM.SYNCCounter},
{0x2104, 0x00, 0x86, 2, (void*)&CO_OD_RAM.SYNCTime},
{0x2106, 0x00, 0x86, 4, (void*)&CO_OD_RAM.powerOnCounter},
{0x2107, 0x05, 0x06, 0, (void*)&CO_OD_RAM.performance[0]},
{0x2108, 0x01, 0x06, 0, (void*)&CO_OD_RAM.temperature[0]},
{0x2109, 0x01, 0x06, 0, (void*)&CO_OD_RAM.voltage[0]},
{0x2110, 0x10, 0x06, 0, (void*)&CO_OD_RAM.variableInt32[0]},
{0x2111, 0x10, 0x06, 0, (void*)&CO_OD_RAM.variableROM_Int32[0]},
{0x2112, 0x10, 0x06, 0, (void*)&CO_OD_RAM.variableNV_Int32[0]},
{0x2120, 0x05, 0x00, 0, (void*)&OD_record2120},
{0x2130, 0x03, 0x00, 0, (void*)&OD_record2130},
{0x2301, 0x08, 0x00, 0, (void*)&OD_record2301},
{0x2302, 0x08, 0x00, 0, (void*)&OD_record2302},
{0x2400, 0x00, 0x1E, 1, (void*)&CO_OD_RAM.traceEnable},
{0x2401, 0x06, 0x00, 0, (void*)&OD_record2401},
{0x2402, 0x06, 0x00, 0, (void*)&OD_record2402},
{0x6010, 0x00, 0xFE, 2, (void*)&CO_OD_RAM.IGUS_D1Controlword1},
{0x6011, 0x00, 0xBE, 2, (void*)&CO_OD_RAM.IGUS_D1Statusword1},
{0x6012, 0x00, 0xBE, 4, (void*)&CO_OD_RAM.IGUS_D1PositionActualValue1},
{0x6013, 0x00, 0xBE, 4, (void*)&CO_OD_RAM.IGUS_D1VelocityActualValue1},
{0x6014, 0x00, 0xFE, 4, (void*)&CO_OD_RAM.IGUS_D1TargetPosition1},
{0x6015, 0x00, 0xBE, 4, (void*)&CO_OD_RAM.IGUS_D1TargetVelocity1},
{0x6020, 0x00, 0xFE, 2, (void*)&CO_OD_RAM.IGUS_D1Controlword2},
{0x6021, 0x00, 0xBE, 2, (void*)&CO_OD_RAM.IGUS_D1Statusword2},
{0x6022, 0x00, 0xBE, 4, (void*)&CO_OD_RAM.IGUS_D1PositionActualValue2},
{0x6023, 0x00, 0xBE, 4, (void*)&CO_OD_RAM.IGUS_D1VelocityActualValue2},
{0x6024, 0x00, 0xFE, 4, (void*)&CO_OD_RAM.IGUS_D1TargetPosition2},
{0x6025, 0x00, 0xBE, 4, (void*)&CO_OD_RAM.IGUS_D1TargetVelocity2},
{0x6030, 0x00, 0xFE, 2, (void*)&CO_OD_RAM.IGUS_D1Controlword3},
{0x6031, 0x00, 0xBE, 2, (void*)&CO_OD_RAM.IGUS_D1Statusword3},
{0x6032, 0x00, 0xBE, 4, (void*)&CO_OD_RAM.IGUS_D1PositionActualValue3},
{0x6033, 0x00, 0xBE, 4, (void*)&CO_OD_RAM.IGUS_D1VelocityActualValue3},
{0x6034, 0x00, 0xFE, 4, (void*)&CO_OD_RAM.IGUS_D1TargetPosition3},
{0x6035, 0x00, 0xBE, 4, (void*)&CO_OD_RAM.IGUS_D1TargetVelocity3},
{0x6040, 0x00, 0xFE, 2, (void*)&CO_OD_RAM.IGUS_D1Controlword4},
{0x6041, 0x00, 0xBE, 2, (void*)&CO_OD_RAM.IGUS_D1Statusword4},
{0x6042, 0x00, 0xBE, 4, (void*)&CO_OD_RAM.IGUS_D1PositionActualValue4},
{0x6043, 0x00, 0xBE, 4, (void*)&CO_OD_RAM.IGUS_D1VelocityActualValue4},
{0x6044, 0x00, 0xFE, 4, (void*)&CO_OD_RAM.IGUS_D1TargetPosition4},
{0x6045, 0x00, 0xBE, 4, (void*)&CO_OD_RAM.IGUS_D1TargetVelocity4},
{0x6050, 0x00, 0xFE, 2, (void*)&CO_OD_RAM.IGUS_D1Controlword5},
{0x6051, 0x00, 0xBE, 2, (void*)&CO_OD_RAM.IGUS_D1Statusword5},
{0x6052, 0x00, 0xBE, 4, (void*)&CO_OD_RAM.IGUS_D1PositionActualValue5},
{0x6053, 0x00, 0xBE, 4, (void*)&CO_OD_RAM.IGUS_D1VelocityActualValue5},
{0x6054, 0x00, 0xFE, 4, (void*)&CO_OD_RAM.IGUS_D1TargetPosition5},
{0x6055, 0x00, 0xBE, 4, (void*)&CO_OD_RAM.IGUS_D1TargetVelocity5},
{0x6060, 0x00, 0xFE, 2, (void*)&CO_OD_RAM.IGUS_D1Controlword6},
{0x6061, 0x00, 0xBE, 2, (void*)&CO_OD_RAM.IGUS_D1Statusword6},
{0x6062, 0x00, 0xBE, 4, (void*)&CO_OD_RAM.IGUS_D1PositionActualValue6},
{0x6063, 0x00, 0xBE, 4, (void*)&CO_OD_RAM.IGUS_D1VelocityActualValue6},
{0x6064, 0x00, 0xFE, 4, (void*)&CO_OD_RAM.IGUS_D1TargetPosition6},
{0x6065, 0x00, 0xBE, 4, (void*)&CO_OD_RAM.IGUS_D1TargetVelocity6},
{0x60A0, 0x00, 0xFE, 2, (void*)&CO_OD_RAM.motorControlword1},
{0x60A1, 0x00, 0xBE, 2, (void*)&CO_OD_RAM.motorStatusword1},
{0x60A2, 0x00, 0xFE, 2, (void*)&CO_OD_RAM.motorTargetVelocity1},
{0x60A3, 0x00, 0xBE, 2, (void*)&CO_OD_RAM.motorActualVelocity1},
{0x60B0, 0x00, 0xFE, 2, (void*)&CO_OD_RAM.motorControlword2},
{0x60B1, 0x00, 0xBE, 2, (void*)&CO_OD_RAM.motorStatusword2},
```

```
{0x60B2, 0x00, 0xFE, 2, (void*)&CO_OD_RAM.motorTargetVelocity2},
{0x60B3, 0x00, 0xBE, 2, (void*)&CO_OD_RAM.motorActualVelocity2},
{0x60C0, 0x00, 0xFE, 2, (void*)&CO_OD_RAM.motorControlword3},
{0x60C1, 0x00, 0xBE, 2, (void*)&CO_OD_RAM.motorStatusword3},
{0x60C2, 0x00, 0xFE, 2, (void*)&CO_OD_RAM.motorTargetVelocity3},
{0x60C3, 0x00, 0xBE, 2, (void*)&CO_OD_RAM.motorActualVelocity3},
{0x60D0, 0x00, 0xFE, 2, (void*)&CO_OD_RAM.motorControlword4},
{0x60D1, 0x00, 0xBE, 2, (void*)&CO_OD_RAM.motorStatusword4},
{0x60D2, 0x00, 0xFE, 2, (void*)&CO_OD_RAM.motorTargetVelocity4},
{0x60D3, 0x00, 0xBE, 2, (void*)&CO_OD_RAM.motorActualVelocity4},
};
// clang-format on
```



## Fails "CO\_OD.h"

```
// clang-format off
/*****

File - CO_OD.c/CO_OD.h
CANopen Object Dictionary.

This file was automatically generated with libedssharp Object
Dictionary Editor vUnknown DON'T EDIT THIS FILE MANUALLY !!!!
*****/

#ifndef CO_OD_H_
#define CO_OD_H_

/*****
CANopen DATA TYPES
*****/
typedef bool_t          BOOLEAN;
typedef uint8_t         UNSIGNED8;
typedef uint16_t        UNSIGNED16;
typedef uint32_t        UNSIGNED32;
typedef uint64_t        UNSIGNED64;
typedef int8_t          INTEGER8;
typedef int16_t         INTEGER16;
typedef int32_t         INTEGER32;
typedef int64_t         INTEGER64;
typedef float32_t       REAL32;
typedef float64_t       REAL64;
typedef char_t          VISIBLE_STRING;
typedef oChar_t         OCTET_STRING;

#ifdef DOMAIN
#undef DOMAIN
#endif

typedef domain_t        DOMAIN;

#ifndef timeOfDay_t
typedef union {
    unsigned long ullValue;
    struct {
        unsigned long ms:28;
        unsigned reserved:4;
        unsigned days:16;
        unsigned reserved2:16;
    };
}timeOfDay_t;
#endif

typedef timeOfDay_t     TIME_OF_DAY;
typedef timeOfDay_t     TIME_DIFFERENCE;

/*****
FILE INFO:
FileName:
FileVersion: 0
CreationTime: 5:01pēcp.
CreationDate: 08-29-2018
CreatedBy: Vitalijs
*****/
```

```

/*****
DEVICE INFO:
VendorName:    Mitavas_roboti
VendorNumber:  0
ProductName:   RONINMainController
ProductNumber: 0
*****/

```

```

/*****
FEATURES
*****/
#define CO_NO_SYNC                0    //Associated objects: 1005-1007
#define CO_NO_EMERGENCY          1    //Associated objects: 1014, 1015
#define CO_NO_TIME                0    //Associated objects: 1012, 1013
#define CO_NO_SDO_SERVER         1    //Associated objects: 1200-127F
#define CO_NO_SDO_CLIENT         1    //Associated objects: 1280-12FF
#define CO_NO_LSS_SERVER         0    //LSS Slave
#define CO_NO_LSS_CLIENT         0    //LSS Master
#define CO_NO_GFC                0
#define CO_NO_SRDO               0
#define CO_NO_RPDO               10   //Associated objects: 14xx, 16xx
#define CO_NO_TPDO               10   //Associated objects: 18xx, 1Axx
#define CO_NO_NMT_MASTER         1
#define CO_NO_TRACE              0

```

```

/*****
OBJECT DICTIONARY
*****/
#define CO_OD_NoOfElements        133

```

```

/*****
TYPE DEFINITIONS FOR RECORDS
*****/
/*1018      */ typedef struct {
    UNSIGNED8      maxSubIndex;
    UNSIGNED32     vendorID;
    UNSIGNED32     productCode;
    UNSIGNED32     revisionNumber;
    UNSIGNED32     serialNumber;
    }              OD_identity_t;
/*1200      */ typedef struct {
    UNSIGNED8      maxSubIndex;
    UNSIGNED32     COB_IDClientToServer;
    UNSIGNED32     COB_IDServerToClient;
    }              OD_SDOServerParameter_t;
/*1280      */ typedef struct {
    UNSIGNED8      maxSubIndex;
    UNSIGNED32     COB_IDClientToServer;
    UNSIGNED32     COB_IDServerToClient;
    UNSIGNED8      nodeIDOfTheSDOServer;
    }              OD_SDOClientParameter_t;
/*1400      */ typedef struct {
    UNSIGNED8      maxSubIndex;
    UNSIGNED32     COB_IDUsedByRPDO;
    UNSIGNED8      transmissionType;
    }              OD_RPDOCommunicationParameter_t;
/*1600      */ typedef struct {
    UNSIGNED8      numberOfMappedObjects;
    UNSIGNED32     mappedObject1;
    UNSIGNED32     mappedObject2;

```

```

        UNSIGNED32    mappedObject3;
        UNSIGNED32    mappedObject4;
        UNSIGNED32    mappedObject5;
        UNSIGNED32    mappedObject6;
        UNSIGNED32    mappedObject7;
        UNSIGNED32    mappedObject8;
    }
    OD_RPDOMappingParameter_t;
/*1800 */ typedef struct {
    UNSIGNED8        maxSubIndex;
    UNSIGNED32       COB_IDUsedByTPDO;
    UNSIGNED8        transmissionType;
    UNSIGNED16       inhibitTime;
    UNSIGNED8        compatibilityEntry;
    UNSIGNED16       eventTimer;
    UNSIGNED8        SYNCStartValue;
    }
    OD_TPDOCommunicationParameter_t;
/*1A00 */ typedef struct {
    UNSIGNED8        numberOfMappedObjects;
    UNSIGNED32       mappedObject1;
    UNSIGNED32       mappedObject2;
    UNSIGNED32       mappedObject3;
    UNSIGNED32       mappedObject4;
    UNSIGNED32       mappedObject5;
    UNSIGNED32       mappedObject6;
    UNSIGNED32       mappedObject7;
    UNSIGNED32       mappedObject8;
    }
    OD_TPDOMappingParameter_t;
/*2120 */ typedef struct {
    UNSIGNED8        maxSubIndex;
    INTEGER64        I64;
    UNSIGNED64       U64;
    REAL32           R32;
    REAL64           R64;
    DOMAIN           domain;
    }
    OD_testVar_t;
/*2130 */ typedef struct {
    UNSIGNED8        maxSubIndex;
    VISIBLE_STRING  string[1];
    UNSIGNED64       epochTimeBaseMs;
    UNSIGNED32       epochTimeOffsetMs;
    }
    OD_time_t;
/*2301 */ typedef struct {
    UNSIGNED8        maxSubIndex;
    UNSIGNED32       size;
    UNSIGNED8        axisNo;
    VISIBLE_STRING  name[6];
    VISIBLE_STRING  color[3];
    UNSIGNED32       map;
    UNSIGNED8        format;
    UNSIGNED8        trigger;
    INTEGER32        threshold;
    }
    OD_traceConfig_t;
/*2401 */ typedef struct {
    UNSIGNED8        maxSubIndex;
    UNSIGNED32       size;
    INTEGER32        value;
    INTEGER32        min;
    INTEGER32        max;
    DOMAIN           plot;
    UNSIGNED32       triggerTime;
    }
    OD_trace_t;

```

```

/*****
TYPE DEFINITIONS FOR OBJECT DICTIONARY INDEXES

```

some of those are redundant with CO\_SDO.h CO\_ObjDicId\_t <Common CiA301 object dictionary entries>

```
*****/
/*0005 */
    #define OD_0005_compatibilityEntry 0x0005

/*1000 */
    #define OD_1000_deviceType 0x1000

/*1001 */
    #define OD_1001_errorRegister 0x1001

/*1002 */
    #define OD_1002_manufacturerStatusRegister 0x1002

/*1003 */
    #define OD_1003_preDefinedErrorField 0x1003

    #define OD_1003_0_preDefinedErrorField_maxSubIndex 0
    #define OD_1003_1_preDefinedErrorField_standardErrorField 1
    #define OD_1003_2_preDefinedErrorField_standardErrorField 2
    #define OD_1003_3_preDefinedErrorField_standardErrorField 3
    #define OD_1003_4_preDefinedErrorField_standardErrorField 4
    #define OD_1003_5_preDefinedErrorField_standardErrorField 5
    #define OD_1003_6_preDefinedErrorField_standardErrorField 6
    #define OD_1003_7_preDefinedErrorField_standardErrorField 7
    #define OD_1003_8_preDefinedErrorField_standardErrorField 8

/*1006 */
    #define OD_1006_communicationCyclePeriod 0x1006

/*1007 */
    #define OD_1007_synchronousWindowLength 0x1007

/*1008 */
    #define OD_1008_manufacturerDeviceName 0x1008

/*1009 */
    #define OD_1009_manufacturerHardwareVersion 0x1009

/*100A */
    #define OD_100A_manufacturerSoftwareVersion 0x100A

/*1010 */
    #define OD_1010_storeParameters 0x1010

    #define OD_1010_0_storeParameters_maxSubIndex 0
    #define OD_1010_1_storeParameters_saveAllParameters 1

/*1011 */
    #define OD_1011_restoreDefaultParameters 0x1011

    #define OD_1011_0_restoreDefaultParameters_maxSubIndex 0
    #define OD_1011_1_restoreDefaultParameters_restoreAllDefaultParameters 1

/*1014 */
    #define OD_1014_COB_ID_EMCY 0x1014

/*1015 */
    #define OD_1015_inhibitTimeEMCY 0x1015

/*1016 */
    #define OD_1016_consumerHeartbeatTime 0x1016
```

```

#define OD_1016_0_consumerHeartbeatTime_maxSubIndex      0
#define OD_1016_1_consumerHeartbeatTime_consumerHeartbeatTime 1
#define OD_1016_2_consumerHeartbeatTime_consumerHeartbeatTime 2
#define OD_1016_3_consumerHeartbeatTime_consumerHeartbeatTime 3
#define OD_1016_4_consumerHeartbeatTime_consumerHeartbeatTime 4

/*1017 */
#define OD_1017_producerHeartbeatTime                    0x1017

/*1018 */
#define OD_1018_identity                                 0x1018

#define OD_1018_0_identity_maxSubIndex                  0
#define OD_1018_1_identity_vendorID                    1
#define OD_1018_2_identity_productCode                 2
#define OD_1018_3_identity_revisionNumber              3
#define OD_1018_4_identity_serialNumber                4

/*1019 */
#define OD_1019_synchronousCounterOverflowValue         0x1019

/*1029 */
#define OD_1029_errorBehavior                           0x1029

#define OD_1029_0_errorBehavior_maxSubIndex            0
#define OD_1029_1_errorBehavior_communication          1
#define OD_1029_2_errorBehavior_communicationOther    2
#define OD_1029_3_errorBehavior_communicationPassive  3
#define OD_1029_4_errorBehavior_generic               4
#define OD_1029_5_errorBehavior_deviceProfile         5
#define OD_1029_6_errorBehavior_manufacturerSpecific  6

/*1200 */
#define OD_1200_SDOServerParameter                     0x1200

#define OD_1200_0_SDOServerParameter_maxSubIndex      0
#define OD_1200_1_SDOServerParameter_COB_IDClientToServer 1
#define OD_1200_2_SDOServerParameter_COB_IDServerToClient 2

/*1280 */
#define OD_1280_SDOClientParameter                    0x1280

#define OD_1280_0_SDOClientParameter_maxSubIndex     0
#define OD_1280_1_SDOClientParameter_COB_IDClientToServer 1
#define OD_1280_2_SDOClientParameter_COB_IDServerToClient 2
#define OD_1280_3_SDOClientParameter_nodeIDOfTheSDOserver 3

/*1400 */
#define OD_1400_RPDOCommunicationParameter            0x1400

#define OD_1400_0_RPDOCommunicationParameter_maxSubIndex 0
#define OD_1400_1_RPDOCommunicationParameter_COB_IDUsedByRPDO 1
#define OD_1400_2_RPDOCommunicationParameter_transmissionType 2

/*1401 */
#define OD_1401_RPDOCommunicationParameter           0x1401

#define OD_1401_0_RPDOCommunicationParameter_maxSubIndex 0
#define OD_1401_1_RPDOCommunicationParameter_COB_IDUsedByRPDO 1
#define OD_1401_2_RPDOCommunicationParameter_transmissionType 2

/*1402 */
#define OD_1402_RPDOCommunicationParameter           0x1402

```

```

#define OD_1402_0_RPDOCommunicationParameter_maxSubIndex    0
#define OD_1402_1_RPDOCommunicationParameter_COB_IDUsedByRPDO 1
#define OD_1402_2_RPDOCommunicationParameter_transmissionType 2

/*1403 */
#define OD_1403_RPDOCommunicationParameter                0x1403

#define OD_1403_0_RPDOCommunicationParameter_maxSubIndex    0
#define OD_1403_1_RPDOCommunicationParameter_COB_IDUsedByRPDO 1
#define OD_1403_2_RPDOCommunicationParameter_transmissionType 2

/*1404 */
#define OD_1404_RPDOCommunicationParameter                0x1404

#define OD_1404_0_RPDOCommunicationParameter_maxSubIndex    0
#define OD_1404_1_RPDOCommunicationParameter_COB_IDUsedByRPDO 1
#define OD_1404_2_RPDOCommunicationParameter_transmissionType 2

/*1405 */
#define OD_1405_RPDOCommunicationParameter                0x1405

#define OD_1405_0_RPDOCommunicationParameter_maxSubIndex    0
#define OD_1405_1_RPDOCommunicationParameter_COB_IDUsedByRPDO 1
#define OD_1405_2_RPDOCommunicationParameter_transmissionType 2

/*1406 */
#define OD_1406_RPDOCommunicationParameter                0x1406

#define OD_1406_0_RPDOCommunicationParameter_maxSubIndex    0
#define OD_1406_1_RPDOCommunicationParameter_COB_IDUsedByRPDO 1
#define OD_1406_2_RPDOCommunicationParameter_transmissionType 2

/*1407 */
#define OD_1407_RPDOCommunicationParameter                0x1407

#define OD_1407_0_RPDOCommunicationParameter_maxSubIndex    0
#define OD_1407_1_RPDOCommunicationParameter_COB_IDUsedByRPDO 1
#define OD_1407_2_RPDOCommunicationParameter_transmissionType 2

/*1408 */
#define OD_1408_RPDOCommunicationParameter                0x1408

#define OD_1408_0_RPDOCommunicationParameter_maxSubIndex    0
#define OD_1408_1_RPDOCommunicationParameter_COB_IDUsedByRPDO 1
#define OD_1408_2_RPDOCommunicationParameter_transmissionType 2

/*1409 */
#define OD_1409_RPDOCommunicationParameter                0x1409

#define OD_1409_0_RPDOCommunicationParameter_maxSubIndex    0
#define OD_1409_1_RPDOCommunicationParameter_COB_IDUsedByRPDO 1
#define OD_1409_2_RPDOCommunicationParameter_transmissionType 2

/*1600 */
#define OD_1600_RPDOMappingParameter                    0x1600

#define OD_1600_0_RPDOMappingParameter_maxSubIndex        0
#define OD_1600_1_RPDOMappingParameter_mappedObject1      1
#define OD_1600_2_RPDOMappingParameter_mappedObject2      2
#define OD_1600_3_RPDOMappingParameter_mappedObject3      3
#define OD_1600_4_RPDOMappingParameter_mappedObject4      4
#define OD_1600_5_RPDOMappingParameter_mappedObject5      5
#define OD_1600_6_RPDOMappingParameter_mappedObject6      6

```

```

#define OD_1600_7_RPDOMappingParameter_mappedObject7 7
#define OD_1600_8_RPDOMappingParameter_mappedObject8 8

/*1601 */
#define OD_1601_RPDOMappingParameter 0x1601

#define OD_1601_0_RPDOMappingParameter_maxSubIndex 0
#define OD_1601_1_RPDOMappingParameter_mappedObject1 1
#define OD_1601_2_RPDOMappingParameter_mappedObject2 2
#define OD_1601_3_RPDOMappingParameter_mappedObject3 3
#define OD_1601_4_RPDOMappingParameter_mappedObject4 4
#define OD_1601_5_RPDOMappingParameter_mappedObject5 5
#define OD_1601_6_RPDOMappingParameter_mappedObject6 6
#define OD_1601_7_RPDOMappingParameter_mappedObject7 7
#define OD_1601_8_RPDOMappingParameter_mappedObject8 8

/*1602 */
#define OD_1602_RPDOMappingParameter 0x1602

#define OD_1602_0_RPDOMappingParameter_maxSubIndex 0
#define OD_1602_1_RPDOMappingParameter_mappedObject1 1
#define OD_1602_2_RPDOMappingParameter_mappedObject2 2
#define OD_1602_3_RPDOMappingParameter_mappedObject3 3
#define OD_1602_4_RPDOMappingParameter_mappedObject4 4
#define OD_1602_5_RPDOMappingParameter_mappedObject5 5
#define OD_1602_6_RPDOMappingParameter_mappedObject6 6
#define OD_1602_7_RPDOMappingParameter_mappedObject7 7
#define OD_1602_8_RPDOMappingParameter_mappedObject8 8

/*1603 */
#define OD_1603_RPDOMappingParameter 0x1603

#define OD_1603_0_RPDOMappingParameter_maxSubIndex 0
#define OD_1603_1_RPDOMappingParameter_mappedObject1 1
#define OD_1603_2_RPDOMappingParameter_mappedObject2 2
#define OD_1603_3_RPDOMappingParameter_mappedObject3 3
#define OD_1603_4_RPDOMappingParameter_mappedObject4 4
#define OD_1603_5_RPDOMappingParameter_mappedObject5 5
#define OD_1603_6_RPDOMappingParameter_mappedObject6 6
#define OD_1603_7_RPDOMappingParameter_mappedObject7 7
#define OD_1603_8_RPDOMappingParameter_mappedObject8 8

/*1604 */
#define OD_1604_RPDOMappingParameter 0x1604

#define OD_1604_0_RPDOMappingParameter_maxSubIndex 0
#define OD_1604_1_RPDOMappingParameter_mappedObject1 1
#define OD_1604_2_RPDOMappingParameter_mappedObject2 2
#define OD_1604_3_RPDOMappingParameter_mappedObject3 3
#define OD_1604_4_RPDOMappingParameter_mappedObject4 4
#define OD_1604_5_RPDOMappingParameter_mappedObject5 5
#define OD_1604_6_RPDOMappingParameter_mappedObject6 6
#define OD_1604_7_RPDOMappingParameter_mappedObject7 7
#define OD_1604_8_RPDOMappingParameter_mappedObject8 8

/*1605 */
#define OD_1605_RPDOMappingParameter 0x1605

#define OD_1605_0_RPDOMappingParameter_maxSubIndex 0
#define OD_1605_1_RPDOMappingParameter_mappedObject1 1
#define OD_1605_2_RPDOMappingParameter_mappedObject2 2
#define OD_1605_3_RPDOMappingParameter_mappedObject3 3
#define OD_1605_4_RPDOMappingParameter_mappedObject4 4
#define OD_1605_5_RPDOMappingParameter_mappedObject5 5

```

```

#define OD_1605_6_RPDOMappingParameter_mappedObject6      6
#define OD_1605_7_RPDOMappingParameter_mappedObject7      7
#define OD_1605_8_RPDOMappingParameter_mappedObject8      8

/*1606 */
#define OD_1606_RPDOMappingParameter                      0x1606

#define OD_1606_0_RPDOMappingParameter_maxSubIndex      0
#define OD_1606_1_RPDOMappingParameter_mappedObject1    1
#define OD_1606_2_RPDOMappingParameter_mappedObject2    2
#define OD_1606_3_RPDOMappingParameter_mappedObject3    3
#define OD_1606_4_RPDOMappingParameter_mappedObject4    4
#define OD_1606_5_RPDOMappingParameter_mappedObject5    5
#define OD_1606_6_RPDOMappingParameter_mappedObject6    6
#define OD_1606_7_RPDOMappingParameter_mappedObject7    7
#define OD_1606_8_RPDOMappingParameter_mappedObject8    8

/*1607 */
#define OD_1607_RPDOMappingParameter                      0x1607

#define OD_1607_0_RPDOMappingParameter_maxSubIndex      0
#define OD_1607_1_RPDOMappingParameter_mappedObject1    1
#define OD_1607_2_RPDOMappingParameter_mappedObject2    2
#define OD_1607_3_RPDOMappingParameter_mappedObject3    3
#define OD_1607_4_RPDOMappingParameter_mappedObject4    4
#define OD_1607_5_RPDOMappingParameter_mappedObject5    5
#define OD_1607_6_RPDOMappingParameter_mappedObject6    6
#define OD_1607_7_RPDOMappingParameter_mappedObject7    7
#define OD_1607_8_RPDOMappingParameter_mappedObject8    8

/*1608 */
#define OD_1608_RPDOMappingParameter                      0x1608

#define OD_1608_0_RPDOMappingParameter_maxSubIndex      0
#define OD_1608_1_RPDOMappingParameter_mappedObject1    1
#define OD_1608_2_RPDOMappingParameter_mappedObject2    2
#define OD_1608_3_RPDOMappingParameter_mappedObject3    3
#define OD_1608_4_RPDOMappingParameter_mappedObject4    4
#define OD_1608_5_RPDOMappingParameter_mappedObject5    5
#define OD_1608_6_RPDOMappingParameter_mappedObject6    6
#define OD_1608_7_RPDOMappingParameter_mappedObject7    7
#define OD_1608_8_RPDOMappingParameter_mappedObject8    8

/*1609 */
#define OD_1609_RPDOMappingParameter                      0x1609

#define OD_1609_0_RPDOMappingParameter_maxSubIndex      0
#define OD_1609_1_RPDOMappingParameter_mappedObject1    1
#define OD_1609_2_RPDOMappingParameter_mappedObject2    2
#define OD_1609_3_RPDOMappingParameter_mappedObject3    3
#define OD_1609_4_RPDOMappingParameter_mappedObject4    4
#define OD_1609_5_RPDOMappingParameter_mappedObject5    5
#define OD_1609_6_RPDOMappingParameter_mappedObject6    6
#define OD_1609_7_RPDOMappingParameter_mappedObject7    7
#define OD_1609_8_RPDOMappingParameter_mappedObject8    8

/*1800 */
#define OD_1800_TPDOCommunicationParameter                0x1800

#define OD_1800_0_TPDOCommunicationParameter_maxSubIndex 0
#define OD_1800_1_TPDOCommunicationParameter_COB_IDUsedByTPDO 1
#define OD_1800_2_TPDOCommunicationParameter_transmissionType 2
#define OD_1800_3_TPDOCommunicationParameter_inhibitTime 3
#define OD_1800_4_TPDOCommunicationParameter_compatibilityEntry 4

```



```

#define OD_1800_5_TPDOCommunicationParameter_eventTimer      5
#define OD_1800_6_TPDOCommunicationParameter_SYNCStartValue 6

/*1801 */
#define OD_1801_TPDOCommunicationParameter                    0x1801

#define OD_1801_0_TPDOCommunicationParameter_maxSubIndex    0
#define OD_1801_1_TPDOCommunicationParameter_COB_IDUsedByTPDO 1
#define OD_1801_2_TPDOCommunicationParameter_transmissionType 2
#define OD_1801_3_TPDOCommunicationParameter_inhibitTime    3
#define OD_1801_4_TPDOCommunicationParameter_compatibilityEntry 4
#define OD_1801_5_TPDOCommunicationParameter_eventTimer      5
#define OD_1801_6_TPDOCommunicationParameter_SYNCStartValue 6

/*1802 */
#define OD_1802_TPDOCommunicationParameter                    0x1802

#define OD_1802_0_TPDOCommunicationParameter_maxSubIndex    0
#define OD_1802_1_TPDOCommunicationParameter_COB_IDUsedByTPDO 1
#define OD_1802_2_TPDOCommunicationParameter_transmissionType 2
#define OD_1802_3_TPDOCommunicationParameter_inhibitTime    3
#define OD_1802_4_TPDOCommunicationParameter_compatibilityEntry 4
#define OD_1802_5_TPDOCommunicationParameter_eventTimer      5
#define OD_1802_6_TPDOCommunicationParameter_SYNCStartValue 6

/*1803 */
#define OD_1803_TPDOCommunicationParameter                    0x1803

#define OD_1803_0_TPDOCommunicationParameter_maxSubIndex    0
#define OD_1803_1_TPDOCommunicationParameter_COB_IDUsedByTPDO 1
#define OD_1803_2_TPDOCommunicationParameter_transmissionType 2
#define OD_1803_3_TPDOCommunicationParameter_inhibitTime    3
#define OD_1803_4_TPDOCommunicationParameter_compatibilityEntry 4
#define OD_1803_5_TPDOCommunicationParameter_eventTimer      5
#define OD_1803_6_TPDOCommunicationParameter_SYNCStartValue 6

/*1804 */
#define OD_1804_TPDOCommunicationParameter                    0x1804

#define OD_1804_0_TPDOCommunicationParameter_maxSubIndex    0
#define OD_1804_1_TPDOCommunicationParameter_COB_IDUsedByTPDO 1
#define OD_1804_2_TPDOCommunicationParameter_transmissionType 2
#define OD_1804_3_TPDOCommunicationParameter_inhibitTime    3
#define OD_1804_4_TPDOCommunicationParameter_compatibilityEntry 4
#define OD_1804_5_TPDOCommunicationParameter_eventTimer      5
#define OD_1804_6_TPDOCommunicationParameter_SYNCStartValue 6

/*1805 */
#define OD_1805_TPDOCommunicationParameter                    0x1805

#define OD_1805_0_TPDOCommunicationParameter_maxSubIndex    0
#define OD_1805_1_TPDOCommunicationParameter_COB_IDUsedByTPDO 1
#define OD_1805_2_TPDOCommunicationParameter_transmissionType 2
#define OD_1805_3_TPDOCommunicationParameter_inhibitTime    3
#define OD_1805_4_TPDOCommunicationParameter_compatibilityEntry 4
#define OD_1805_5_TPDOCommunicationParameter_eventTimer      5
#define OD_1805_6_TPDOCommunicationParameter_SYNCStartValue 6

/*1806 */
#define OD_1806_TPDOCommunicationParameter                    0x1806

#define OD_1806_0_TPDOCommunicationParameter_maxSubIndex    0
#define OD_1806_1_TPDOCommunicationParameter_COB_IDUsedByTPDO 1
#define OD_1806_2_TPDOCommunicationParameter_transmissionType 2

```

```

#define OD_1806_3_TPDOCommunicationParameter_inhibitTime      3
#define OD_1806_4_TPDOCommunicationParameter_compatibilityEntry 4
#define OD_1806_5_TPDOCommunicationParameter_eventTimer      5
#define OD_1806_6_TPDOCommunicationParameter_SYNCStartValue 6

/*1807 */
#define OD_1807_TPDOCommunicationParameter                    0x1807

#define OD_1807_0_TPDOCommunicationParameter_maxSubIndex     0
#define OD_1807_1_TPDOCommunicationParameter_COB_IDUsedByTPDO 1
#define OD_1807_2_TPDOCommunicationParameter_transmissionType 2
#define OD_1807_3_TPDOCommunicationParameter_inhibitTime     3
#define OD_1807_4_TPDOCommunicationParameter_compatibilityEntry 4
#define OD_1807_5_TPDOCommunicationParameter_eventTimer      5
#define OD_1807_6_TPDOCommunicationParameter_SYNCStartValue 6

/*1808 */
#define OD_1808_TPDOCommunicationParameter                    0x1808

#define OD_1808_0_TPDOCommunicationParameter_maxSubIndex     0
#define OD_1808_1_TPDOCommunicationParameter_COB_IDUsedByTPDO 1
#define OD_1808_2_TPDOCommunicationParameter_transmissionType 2
#define OD_1808_3_TPDOCommunicationParameter_inhibitTime     3
#define OD_1808_4_TPDOCommunicationParameter_compatibilityEntry 4
#define OD_1808_5_TPDOCommunicationParameter_eventTimer      5
#define OD_1808_6_TPDOCommunicationParameter_SYNCStartValue 6

/*1809 */
#define OD_1809_TPDOCommunicationParameter                    0x1809

#define OD_1809_0_TPDOCommunicationParameter_maxSubIndex     0
#define OD_1809_1_TPDOCommunicationParameter_COB_IDUsedByTPDO 1
#define OD_1809_2_TPDOCommunicationParameter_transmissionType 2
#define OD_1809_3_TPDOCommunicationParameter_inhibitTime     3
#define OD_1809_4_TPDOCommunicationParameter_compatibilityEntry 4
#define OD_1809_5_TPDOCommunicationParameter_eventTimer      5
#define OD_1809_6_TPDOCommunicationParameter_SYNCStartValue 6

/*1A00 */
#define OD_1A00_TPDOMappingParameter                          0x1A00

#define OD_1A00_0_TPDOMappingParameter_maxSubIndex           0
#define OD_1A00_1_TPDOMappingParameter_mappedObject1         1
#define OD_1A00_2_TPDOMappingParameter_mappedObject2         2
#define OD_1A00_3_TPDOMappingParameter_mappedObject3         3
#define OD_1A00_4_TPDOMappingParameter_mappedObject4         4
#define OD_1A00_5_TPDOMappingParameter_mappedObject5         5
#define OD_1A00_6_TPDOMappingParameter_mappedObject6         6
#define OD_1A00_7_TPDOMappingParameter_mappedObject7         7
#define OD_1A00_8_TPDOMappingParameter_mappedObject8         8

/*1A01 */
#define OD_1A01_TPDOMappingParameter                          0x1A01

#define OD_1A01_0_TPDOMappingParameter_maxSubIndex           0
#define OD_1A01_1_TPDOMappingParameter_mappedObject1         1
#define OD_1A01_2_TPDOMappingParameter_mappedObject2         2
#define OD_1A01_3_TPDOMappingParameter_mappedObject3         3
#define OD_1A01_4_TPDOMappingParameter_mappedObject4         4
#define OD_1A01_5_TPDOMappingParameter_mappedObject5         5
#define OD_1A01_6_TPDOMappingParameter_mappedObject6         6
#define OD_1A01_7_TPDOMappingParameter_mappedObject7         7
#define OD_1A01_8_TPDOMappingParameter_mappedObject8         8

```

```

/*1A02 */
#define OD_1A02_TPDOMappingParameter 0x1A02

#define OD_1A02_0_TPDOMappingParameter_maxSubIndex 0
#define OD_1A02_1_TPDOMappingParameter_mappedObject1 1
#define OD_1A02_2_TPDOMappingParameter_mappedObject2 2
#define OD_1A02_3_TPDOMappingParameter_mappedObject3 3
#define OD_1A02_4_TPDOMappingParameter_mappedObject4 4
#define OD_1A02_5_TPDOMappingParameter_mappedObject5 5
#define OD_1A02_6_TPDOMappingParameter_mappedObject6 6
#define OD_1A02_7_TPDOMappingParameter_mappedObject7 7
#define OD_1A02_8_TPDOMappingParameter_mappedObject8 8

/*1A03 */
#define OD_1A03_TPDOMappingParameter 0x1A03

#define OD_1A03_0_TPDOMappingParameter_maxSubIndex 0
#define OD_1A03_1_TPDOMappingParameter_mappedObject1 1
#define OD_1A03_2_TPDOMappingParameter_mappedObject2 2
#define OD_1A03_3_TPDOMappingParameter_mappedObject3 3
#define OD_1A03_4_TPDOMappingParameter_mappedObject4 4
#define OD_1A03_5_TPDOMappingParameter_mappedObject5 5
#define OD_1A03_6_TPDOMappingParameter_mappedObject6 6
#define OD_1A03_7_TPDOMappingParameter_mappedObject7 7
#define OD_1A03_8_TPDOMappingParameter_mappedObject8 8

/*1A04 */
#define OD_1A04_TPDOMappingParameter 0x1A04

#define OD_1A04_0_TPDOMappingParameter_maxSubIndex 0
#define OD_1A04_1_TPDOMappingParameter_mappedObject1 1
#define OD_1A04_2_TPDOMappingParameter_mappedObject2 2
#define OD_1A04_3_TPDOMappingParameter_mappedObject3 3
#define OD_1A04_4_TPDOMappingParameter_mappedObject4 4
#define OD_1A04_5_TPDOMappingParameter_mappedObject5 5
#define OD_1A04_6_TPDOMappingParameter_mappedObject6 6
#define OD_1A04_7_TPDOMappingParameter_mappedObject7 7
#define OD_1A04_8_TPDOMappingParameter_mappedObject8 8

/*1A05 */
#define OD_1A05_TPDOMappingParameter 0x1A05

#define OD_1A05_0_TPDOMappingParameter_maxSubIndex 0
#define OD_1A05_1_TPDOMappingParameter_mappedObject1 1
#define OD_1A05_2_TPDOMappingParameter_mappedObject2 2
#define OD_1A05_3_TPDOMappingParameter_mappedObject3 3
#define OD_1A05_4_TPDOMappingParameter_mappedObject4 4
#define OD_1A05_5_TPDOMappingParameter_mappedObject5 5
#define OD_1A05_6_TPDOMappingParameter_mappedObject6 6
#define OD_1A05_7_TPDOMappingParameter_mappedObject7 7
#define OD_1A05_8_TPDOMappingParameter_mappedObject8 8

/*1A06 */
#define OD_1A06_TPDOMappingParameter 0x1A06

#define OD_1A06_0_TPDOMappingParameter_maxSubIndex 0
#define OD_1A06_1_TPDOMappingParameter_mappedObject1 1
#define OD_1A06_2_TPDOMappingParameter_mappedObject2 2
#define OD_1A06_3_TPDOMappingParameter_mappedObject3 3
#define OD_1A06_4_TPDOMappingParameter_mappedObject4 4
#define OD_1A06_5_TPDOMappingParameter_mappedObject5 5
#define OD_1A06_6_TPDOMappingParameter_mappedObject6 6
#define OD_1A06_7_TPDOMappingParameter_mappedObject7 7
#define OD_1A06_8_TPDOMappingParameter_mappedObject8 8

```

```

/*1A07 */
#define OD_1A07_TPDOMappingParameter 0x1A07

#define OD_1A07_0_TPDOMappingParameter_maxSubIndex 0
#define OD_1A07_1_TPDOMappingParameter_mappedObject1 1
#define OD_1A07_2_TPDOMappingParameter_mappedObject2 2
#define OD_1A07_3_TPDOMappingParameter_mappedObject3 3
#define OD_1A07_4_TPDOMappingParameter_mappedObject4 4
#define OD_1A07_5_TPDOMappingParameter_mappedObject5 5
#define OD_1A07_6_TPDOMappingParameter_mappedObject6 6
#define OD_1A07_7_TPDOMappingParameter_mappedObject7 7
#define OD_1A07_8_TPDOMappingParameter_mappedObject8 8

/*1A08 */
#define OD_1A08_TPDOMappingParameter 0x1A08

#define OD_1A08_0_TPDOMappingParameter_maxSubIndex 0
#define OD_1A08_1_TPDOMappingParameter_mappedObject1 1
#define OD_1A08_2_TPDOMappingParameter_mappedObject2 2
#define OD_1A08_3_TPDOMappingParameter_mappedObject3 3
#define OD_1A08_4_TPDOMappingParameter_mappedObject4 4
#define OD_1A08_5_TPDOMappingParameter_mappedObject5 5
#define OD_1A08_6_TPDOMappingParameter_mappedObject6 6
#define OD_1A08_7_TPDOMappingParameter_mappedObject7 7
#define OD_1A08_8_TPDOMappingParameter_mappedObject8 8

/*1A09 */
#define OD_1A09_TPDOMappingParameter 0x1A09

#define OD_1A09_0_TPDOMappingParameter_maxSubIndex 0
#define OD_1A09_1_TPDOMappingParameter_mappedObject1 1
#define OD_1A09_2_TPDOMappingParameter_mappedObject2 2
#define OD_1A09_3_TPDOMappingParameter_mappedObject3 3
#define OD_1A09_4_TPDOMappingParameter_mappedObject4 4
#define OD_1A09_5_TPDOMappingParameter_mappedObject5 5
#define OD_1A09_6_TPDOMappingParameter_mappedObject6 6
#define OD_1A09_7_TPDOMappingParameter_mappedObject7 7
#define OD_1A09_8_TPDOMappingParameter_mappedObject8 8

/*1F80 */
#define OD_1F80_NMTStartup 0x1F80

/*2100 */
#define OD_2100_errorStatusBits 0x2100

/*2101 */
#define OD_2101_CANNodeID 0x2101

/*2102 */
#define OD_2102_CANBitRate 0x2102

/*2103 */
#define OD_2103_SYNCCounter 0x2103

/*2104 */
#define OD_2104_SYNCTime 0x2104

/*2106 */
#define OD_2106_powerOnCounter 0x2106

/*2107 */
#define OD_2107_performance 0x2107

```

```

#define OD_2107_0_performance_maxSubIndex          0
#define OD_2107_1_performance_cyclesPerSecond     1
#define OD_2107_2_performance_timerCycleTime      2
#define OD_2107_3_performance_timerCycleMaxTime   3
#define OD_2107_4_performance_mainCycleTime       4
#define OD_2107_5_performance_mainCycleMaxTime    5

/*2108 */
#define OD_2108_temperature                        0x2108

#define OD_2108_0_temperature_maxSubIndex         0
#define OD_2108_1_temperature_mainPCB            1

/*2109 */
#define OD_2109_voltage                            0x2109

#define OD_2109_0_voltage_maxSubIndex             0
#define OD_2109_1_voltage_mainPCBSupply           1

/*2110 */
#define OD_2110_variableInt32                     0x2110

#define OD_2110_0_variableInt32_maxSubIndex       0
#define OD_2110_1_variableInt32_int32             1
#define OD_2110_2_variableInt32_int32             2
#define OD_2110_3_variableInt32_int32             3
#define OD_2110_4_variableInt32_int32             4
#define OD_2110_5_variableInt32_int32             5
#define OD_2110_6_variableInt32_int32             6
#define OD_2110_7_variableInt32_int32             7
#define OD_2110_8_variableInt32_int32             8
#define OD_2110_9_variableInt32_int32             9
#define OD_2110_10_variableInt32_int32            10
#define OD_2110_11_variableInt32_int32            11
#define OD_2110_12_variableInt32_int32            12
#define OD_2110_13_variableInt32_int32            13
#define OD_2110_14_variableInt32_int32            14
#define OD_2110_15_variableInt32_int32            15
#define OD_2110_16_variableInt32_int32            16

/*2111 */
#define OD_2111_variableROM_Int32                  0x2111

#define OD_2111_0_variableROM_Int32_maxSubIndex   0
#define OD_2111_1_variableROM_Int32_int32         1
#define OD_2111_2_variableROM_Int32_int32         2
#define OD_2111_3_variableROM_Int32_int32         3
#define OD_2111_4_variableROM_Int32_int32         4
#define OD_2111_5_variableROM_Int32_int32         5
#define OD_2111_6_variableROM_Int32_int32         6
#define OD_2111_7_variableROM_Int32_int32         7
#define OD_2111_8_variableROM_Int32_int32         8
#define OD_2111_9_variableROM_Int32_int32         9
#define OD_2111_10_variableROM_Int32_int32        10
#define OD_2111_11_variableROM_Int32_int32        11
#define OD_2111_12_variableROM_Int32_int32        12
#define OD_2111_13_variableROM_Int32_int32        13
#define OD_2111_14_variableROM_Int32_int32        14
#define OD_2111_15_variableROM_Int32_int32        15
#define OD_2111_16_variableROM_Int32_int32        16

/*2112 */
#define OD_2112_variableNV_Int32                   0x2112

```

```

#define OD_2112_0_variableNV_Int32_maxSubIndex      0
#define OD_2112_1_variableNV_Int32_int32           1
#define OD_2112_2_variableNV_Int32_int32           2
#define OD_2112_3_variableNV_Int32_int32           3
#define OD_2112_4_variableNV_Int32_int32           4
#define OD_2112_5_variableNV_Int32_int32           5
#define OD_2112_6_variableNV_Int32_int32           6
#define OD_2112_7_variableNV_Int32_int32           7
#define OD_2112_8_variableNV_Int32_int32           8
#define OD_2112_9_variableNV_Int32_int32           9
#define OD_2112_10_variableNV_Int32_int32          10
#define OD_2112_11_variableNV_Int32_int32          11
#define OD_2112_12_variableNV_Int32_int32          12
#define OD_2112_13_variableNV_Int32_int32          13
#define OD_2112_14_variableNV_Int32_int32          14
#define OD_2112_15_variableNV_Int32_int32          15
#define OD_2112_16_variableNV_Int32_int32          16

/*2120 */
#define OD_2120_testVar                             0x2120

#define OD_2120_0_testVar_maxSubIndex               0
#define OD_2120_1_testVar_I64                       1
#define OD_2120_2_testVar_U64                       2
#define OD_2120_3_testVar_R32                       3
#define OD_2120_4_testVar_R64                       4
#define OD_2120_5_testVar_domain                    5

/*2130 */
#define OD_2130_time                                 0x2130

#define OD_2130_0_time_maxSubIndex                  0
#define OD_2130_1_time_string                       1
#define OD_2130_2_time_epochTimeBaseMs             2
#define OD_2130_3_time_epochTimeOffsetMs           3

/*2301 */
#define OD_2301_traceConfig                          0x2301

#define OD_2301_0_traceConfig_maxSubIndex           0
#define OD_2301_1_traceConfig_size                  1
#define OD_2301_2_traceConfig_axisNo                2
#define OD_2301_3_traceConfig_name                  3
#define OD_2301_4_traceConfig_color                 4
#define OD_2301_5_traceConfig_map                   5
#define OD_2301_6_traceConfig_format                 6
#define OD_2301_7_traceConfig_trigger                7
#define OD_2301_8_traceConfig_threshold              8

/*2302 */
#define OD_2302_traceConfig                          0x2302

#define OD_2302_0_traceConfig_maxSubIndex           0
#define OD_2302_1_traceConfig_size                  1
#define OD_2302_2_traceConfig_axisNo                2
#define OD_2302_3_traceConfig_name                  3
#define OD_2302_4_traceConfig_color                 4
#define OD_2302_5_traceConfig_map                   5
#define OD_2302_6_traceConfig_format                 6
#define OD_2302_7_traceConfig_trigger                7
#define OD_2302_8_traceConfig_threshold              8

/*2400 */
#define OD_2400_traceEnable                          0x2400

```

```

/*2401 */
#define OD_2401_trace 0x2401

#define OD_2401_0_trace_maxSubIndex 0
#define OD_2401_1_trace_size 1
#define OD_2401_2_trace_value 2
#define OD_2401_3_trace_min 3
#define OD_2401_4_trace_max 4
#define OD_2401_5_trace_plot 5
#define OD_2401_6_trace_triggerTime 6

/*2402 */
#define OD_2402_trace 0x2402

#define OD_2402_0_trace_maxSubIndex 0
#define OD_2402_1_trace_size 1
#define OD_2402_2_trace_value 2
#define OD_2402_3_trace_min 3
#define OD_2402_4_trace_max 4
#define OD_2402_5_trace_plot 5
#define OD_2402_6_trace_triggerTime 6

/*6010 */
#define OD_6010_IGUS_D1Controlword1 0x6010

/*6011 */
#define OD_6011_IGUS_D1Statusword1 0x6011

/*6012 */
#define OD_6012_IGUS_D1PositionActualValue1 0x6012

/*6013 */
#define OD_6013_IGUS_D1VelocityActualValue1 0x6013

/*6014 */
#define OD_6014_IGUS_D1TargetPosition1 0x6014

/*6015 */
#define OD_6015_IGUS_D1TargetVelocity1 0x6015

/*6020 */
#define OD_6020_IGUS_D1Controlword2 0x6020

/*6021 */
#define OD_6021_IGUS_D1Statusword2 0x6021

/*6022 */
#define OD_6022_IGUS_D1PositionActualValue2 0x6022

/*6023 */
#define OD_6023_IGUS_D1VelocityActualValue2 0x6023

/*6024 */
#define OD_6024_IGUS_D1TargetPosition2 0x6024

/*6025 */
#define OD_6025_IGUS_D1TargetVelocity2 0x6025

/*6030 */
#define OD_6030_IGUS_D1Controlword3 0x6030

/*6031 */
#define OD_6031_IGUS_D1Statusword3 0x6031

```

```
/*6032 */
    #define OD_6032_IGUS_D1PositionActualValue3          0x6032
/*6033 */
    #define OD_6033_IGUS_D1VelocityActualValue3        0x6033
/*6034 */
    #define OD_6034_IGUS_D1TargetPosition3             0x6034
/*6035 */
    #define OD_6035_IGUS_D1TargetVelocity3             0x6035
/*6040 */
    #define OD_6040_IGUS_D1Controlword4                0x6040
/*6041 */
    #define OD_6041_IGUS_D1Statusword4                 0x6041
/*6042 */
    #define OD_6042_IGUS_D1PositionActualValue4        0x6042
/*6043 */
    #define OD_6043_IGUS_D1VelocityActualValue4        0x6043
/*6044 */
    #define OD_6044_IGUS_D1TargetPosition4             0x6044
/*6045 */
    #define OD_6045_IGUS_D1TargetVelocity4             0x6045
/*6050 */
    #define OD_6050_IGUS_D1Controlword5                0x6050
/*6051 */
    #define OD_6051_IGUS_D1Statusword5                 0x6051
/*6052 */
    #define OD_6052_IGUS_D1PositionActualValue5        0x6052
/*6053 */
    #define OD_6053_IGUS_D1VelocityActualValue5        0x6053
/*6054 */
    #define OD_6054_IGUS_D1TargetPosition5             0x6054
/*6055 */
    #define OD_6055_IGUS_D1TargetVelocity5             0x6055
/*6060 */
    #define OD_6060_IGUS_D1Controlword6                0x6060
/*6061 */
    #define OD_6061_IGUS_D1Statusword6                 0x6061
/*6062 */
    #define OD_6062_IGUS_D1PositionActualValue6        0x6062
/*6063 */
    #define OD_6063_IGUS_D1VelocityActualValue6        0x6063
/*6064 */
    #define OD_6064_IGUS_D1TargetPosition6             0x6064
```



```

/*6065 */
    #define OD_6065_IGUS_D1TargetVelocity6          0x6065

/*60A0 */
    #define OD_60A0_motorControlword1              0x60A0

/*60A1 */
    #define OD_60A1_motorStatusword1              0x60A1

/*60A2 */
    #define OD_60A2_motorTargetVelocity1          0x60A2

/*60A3 */
    #define OD_60A3_motorActualVelocity1          0x60A3

/*60B0 */
    #define OD_60B0_motorControlword2              0x60B0

/*60B1 */
    #define OD_60B1_motorStatusword2              0x60B1

/*60B2 */
    #define OD_60B2_motorTargetVelocity2          0x60B2

/*60B3 */
    #define OD_60B3_motorActualVelocity2          0x60B3

/*60C0 */
    #define OD_60C0_motorControlword3              0x60C0

/*60C1 */
    #define OD_60C1_motorStatusword3              0x60C1

/*60C2 */
    #define OD_60C2_motorTargetVelocity3          0x60C2

/*60C3 */
    #define OD_60C3_motorActualVelocity3          0x60C3

/*60D0 */
    #define OD_60D0_motorControlword4              0x60D0

/*60D1 */
    #define OD_60D1_motorStatusword4              0x60D1

/*60D2 */
    #define OD_60D2_motorTargetVelocity4          0x60D2

/*60D3 */
    #define OD_60D3_motorActualVelocity4          0x60D3

/*****
    STRUCTURES FOR VARIABLES IN DIFFERENT MEMORY LOCATIONS
    *****/
#define CO_OD_FIRST_LAST_WORD      0x55 //Any value from 0x01 to 0xFE. If changed, EEPROM
will be reinitialized.

/***** Structure for ROM variables *****/
struct sCO_OD_ROM{
    UNSIGNED32      FirstWord;

    UNSIGNED32      LastWord;
};

```

```

/***** Structure for RAM variables *****/
struct sCO_OD_RAM{
    UNSIGNED32    FirstWord;

/*0005    */ UNSIGNED8    compatabilityEntry;
/*1000    */ UNSIGNED32   deviceType;
/*1001    */ UNSIGNED8    errorRegister;
/*1002    */ UNSIGNED32   manufacturerStatusRegister;
/*1003    */ UNSIGNED32   preDefinedErrorField[8];
/*1006    */ UNSIGNED32   communicationCyclePeriod;
/*1007    */ UNSIGNED32   synchronousWindowLength;
/*1008    */ VISIBLE_STRING manufacturerDeviceName[11];
/*1009    */ VISIBLE_STRING manufacturerHardwareVersion[4];
/*100A    */ VISIBLE_STRING manufacturerSoftwareVersion[4];
/*1010    */ UNSIGNED32   storeParameters[1];
/*1011    */ UNSIGNED32   restoreDefaultParameters[1];
/*1014    */ UNSIGNED32   COB_ID_EMICY;
/*1015    */ UNSIGNED16   inhibitTimeEMCY;
/*1016    */ UNSIGNED32   consumerHeartbeatTime[4];
/*1017    */ UNSIGNED16   producerHeartbeatTime;
/*1018    */ OD_identity_t identity;
/*1019    */ UNSIGNED8    synchronousCounterOverflowValue;
/*1029    */ UNSIGNED8    errorBehavior[6];
/*1200    */ OD_SDOServerParameter_t SDOServerParameter[1];
/*1280    */ OD_SDOClientParameter_t SDOClientParameter[1];
/*1400    */ OD_RPDOCommunicationParameter_t RPDOCommunicationParameter[10];
/*1600    */ OD_RPDOMappingParameter_t RPDOMappingParameter[10];
/*1800    */ OD_TPDOCCommunicationParameter_t TPDOCCommunicationParameter[10];
/*1A00    */ OD_TPDOMappingParameter_t TPDOMappingParameter[10];
/*1F80    */ UNSIGNED32   NMTStartup;
/*2100    */ OCTET_STRING  errorStatusBits[10];
/*2101    */ UNSIGNED8    CANNodeID;
/*2102    */ UNSIGNED16   CANBitRate;
/*2103    */ UNSIGNED16   SYNCCounter;
/*2104    */ UNSIGNED16   SYNCTime;
/*2106    */ UNSIGNED32   powerOnCounter;
/*2107    */ UNSIGNED16   performance[5];
/*2108    */ INTEGER16    temperature[1];
/*2109    */ INTEGER16    voltage[1];
/*2110    */ INTEGER32    variableInt32[16];
/*2111    */ INTEGER32    variableROM_Int32[16];
/*2112    */ INTEGER32    variableNV_Int32[16];
/*2120    */ OD_testVar_t testVar;
/*2130    */ OD_time_t    time;
/*2301    */ OD_traceConfig_t traceConfig[2];
/*2400    */ UNSIGNED8    traceEnable;
/*2401    */ OD_trace_t    trace[2];
/*6010    */ UNSIGNED16   IGUS_D1Controlword1;
/*6011    */ UNSIGNED16   IGUS_D1Statusword1;
/*6012    */ INTEGER32    IGUS_D1PositionActualValue1;
/*6013    */ INTEGER32    IGUS_D1VelocityActualValue1;
/*6014    */ INTEGER32    IGUS_D1TargetPosition1;
/*6015    */ INTEGER32    IGUS_D1TargetVelocity1;
/*6020    */ UNSIGNED16   IGUS_D1Controlword2;
/*6021    */ UNSIGNED16   IGUS_D1Statusword2;
/*6022    */ INTEGER32    IGUS_D1PositionActualValue2;
/*6023    */ INTEGER32    IGUS_D1VelocityActualValue2;
/*6024    */ INTEGER32    IGUS_D1TargetPosition2;
/*6025    */ INTEGER32    IGUS_D1TargetVelocity2;
/*6030    */ UNSIGNED16   IGUS_D1Controlword3;
/*6031    */ UNSIGNED16   IGUS_D1Statusword3;
/*6032    */ INTEGER32    IGUS_D1PositionActualValue3;
/*6033    */ INTEGER32    IGUS_D1VelocityActualValue3;

```

```

/*6034      */ INTEGER32      IGUS_D1TargetPosition3;
/*6035      */ INTEGER32      IGUS_D1TargetVelocity3;
/*6040      */ UNSIGNED16     IGUS_D1Controlword4;
/*6041      */ UNSIGNED16     IGUS_D1Statusword4;
/*6042      */ INTEGER32      IGUS_D1PositionActualValue4;
/*6043      */ INTEGER32      IGUS_D1VelocityActualValue4;
/*6044      */ INTEGER32      IGUS_D1TargetPosition4;
/*6045      */ INTEGER32      IGUS_D1TargetVelocity4;
/*6050      */ UNSIGNED16     IGUS_D1Controlword5;
/*6051      */ UNSIGNED16     IGUS_D1Statusword5;
/*6052      */ INTEGER32      IGUS_D1PositionActualValue5;
/*6053      */ INTEGER32      IGUS_D1VelocityActualValue5;
/*6054      */ INTEGER32      IGUS_D1TargetPosition5;
/*6055      */ INTEGER32      IGUS_D1TargetVelocity5;
/*6060      */ UNSIGNED16     IGUS_D1Controlword6;
/*6061      */ UNSIGNED16     IGUS_D1Statusword6;
/*6062      */ INTEGER32      IGUS_D1PositionActualValue6;
/*6063      */ INTEGER32      IGUS_D1VelocityActualValue6;
/*6064      */ INTEGER32      IGUS_D1TargetPosition6;
/*6065      */ INTEGER32      IGUS_D1TargetVelocity6;
/*60A0      */ UNSIGNED16     motorControlword1;
/*60A1      */ UNSIGNED16     motorStatusword1;
/*60A2      */ INTEGER16      motorTargetVelocity1;
/*60A3      */ INTEGER16      motorActualVelocity1;
/*60B0      */ UNSIGNED16     motorControlword2;
/*60B1      */ UNSIGNED16     motorStatusword2;
/*60B2      */ INTEGER16      motorTargetVelocity2;
/*60B3      */ INTEGER16      motorActualVelocity2;
/*60C0      */ UNSIGNED16     motorControlword3;
/*60C1      */ UNSIGNED16     motorStatusword3;
/*60C2      */ INTEGER16      motorTargetVelocity3;
/*60C3      */ INTEGER16      motorActualVelocity3;
/*60D0      */ UNSIGNED16     motorControlword4;
/*60D1      */ UNSIGNED16     motorStatusword4;
/*60D2      */ INTEGER16      motorTargetVelocity4;
/*60D3      */ INTEGER16      motorActualVelocity4;

                UNSIGNED32      LastWord;
};

/***** Structure for EEPROM variables *****/
struct sCO_OD_EEPROM{
                UNSIGNED32      FirstWord;

                UNSIGNED32      LastWord;
};

/***** Declaration of Object Dictionary variables *****/
extern struct sCO_OD_ROM CO_OD_ROM;

extern struct sCO_OD_RAM CO_OD_RAM;

extern struct sCO_OD_EEPROM CO_OD_EEPROM;

/***** ALIASES FOR OBJECT DICTIONARY VARIABLES *****/
/*0005, Data Type: UNSIGNED8 */
#define OD_compatibilityEntry
CO_OD_RAM.compatibilityEntry

/*1000, Data Type: UNSIGNED32 */
#define OD_deviceType
CO_OD_RAM.deviceType

```

```

/*1001, Data Type: UNSIGNED8 */
    #define OD_errorRegister
CO_OD_RAM.errorRegister

/*1002, Data Type: UNSIGNED32 */
    #define OD_manufacturerStatusRegister
CO_OD_RAM.manufacturerStatusRegister

/*1003, Data Type: UNSIGNED32, Array[8] */
    #define OD_preDefinedErrorField
CO_OD_RAM.preDefinedErrorField
    #define ODL_preDefinedErrorField_arrayLength      8
    #define ODA_preDefinedErrorField_standardErrorField 0

/*1006, Data Type: UNSIGNED32 */
    #define OD_communicationCyclePeriod
CO_OD_RAM.communicationCyclePeriod

/*1007, Data Type: UNSIGNED32 */
    #define OD_synchronousWindowLength
CO_OD_RAM.synchronousWindowLength

/*1008, Data Type: VISIBLE_STRING */
    #define OD_manufacturerDeviceName
CO_OD_RAM.manufacturerDeviceName
    #define ODL_manufacturerDeviceName_stringLength  11

/*1009, Data Type: VISIBLE_STRING */
    #define OD_manufacturerHardwareVersion
CO_OD_RAM.manufacturerHardwareVersion
    #define ODL_manufacturerHardwareVersion_stringLength  4

/*100A, Data Type: VISIBLE_STRING */
    #define OD_manufacturerSoftwareVersion
CO_OD_RAM.manufacturerSoftwareVersion
    #define ODL_manufacturerSoftwareVersion_stringLength  4

/*1010, Data Type: UNSIGNED32, Array[1] */
    #define OD_storeParameters
CO_OD_RAM.storeParameters
    #define ODL_storeParameters_arrayLength            1
    #define ODA_storeParameters_saveAllParameters     0

/*1011, Data Type: UNSIGNED32, Array[1] */
    #define OD_restoreDefaultParameters
CO_OD_RAM.restoreDefaultParameters
    #define ODL_restoreDefaultParameters_arrayLength  1
    #define ODA_restoreDefaultParameters_restoreAllDefaultParameters 0

/*1014, Data Type: UNSIGNED32 */
    #define OD_COB_ID_EMCY
CO_OD_RAM.COB_ID_EMCY

/*1015, Data Type: UNSIGNED16 */
    #define OD_inhibitTimeEMCY
CO_OD_RAM.inhibitTimeEMCY

/*1016, Data Type: UNSIGNED32, Array[4] */
    #define OD_consumerHeartbeatTime
CO_OD_RAM.consumerHeartbeatTime
    #define ODL_consumerHeartbeatTime_arrayLength     4
    #define ODA_consumerHeartbeatTime_consumerHeartbeatTime 0

/*1017, Data Type: UNSIGNED16 */

```

```

        #define OD_producerHeartbeatTime
CO_OD_RAM.producerHeartbeatTime

/*1018, Data Type: identity_t */
        #define OD_identity                                CO_OD_RAM.identity

/*1019, Data Type: UNSIGNED8 */
        #define OD_synchronousCounterOverflowValue
CO_OD_RAM.synchronousCounterOverflowValue

/*1029, Data Type: UNSIGNED8, Array[6] */
        #define OD_errorBehavior
CO_OD_RAM.errorBehavior
        #define ODL_errorBehavior_arrayLength            6
        #define ODA_errorBehavior_communication          0
        #define ODA_errorBehavior_communicationOther    1
        #define ODA_errorBehavior_communicationPassive  2
        #define ODA_errorBehavior_generic               3
        #define ODA_errorBehavior_deviceProfile         4
        #define ODA_errorBehavior_manufacturerSpecific  5

/*1200, Data Type: SDOServerParameter_t */
        #define OD_SDOServerParameter
CO_OD_RAM.SDOServerParameter

/*1280, Data Type: SDOClientParameter_t */
        #define OD_SDOClientParameter
CO_OD_RAM.SDOClientParameter

/*1400, Data Type: RPDOCommunicationParameter_t */
        #define OD_RPDOCommunicationParameter
CO_OD_RAM.RPDOCommunicationParameter

/*1600, Data Type: RPDOMappingParameter_t */
        #define OD_RPDOMappingParameter
CO_OD_RAM.RPDOMappingParameter

/*1800, Data Type: TPDOCommunicationParameter_t */
        #define OD_TPDOCommunicationParameter
CO_OD_RAM.TPDOCommunicationParameter

/*1A00, Data Type: TPDOMappingParameter_t */
        #define OD_TPDOMappingParameter
CO_OD_RAM.TPDOMappingParameter

/*1F80, Data Type: UNSIGNED32 */
        #define OD_NMTStartup                            CO_OD_RAM.NMTStartup

/*2100, Data Type: OCTET_STRING */
        #define OD_errorStatusBits
CO_OD_RAM.errorStatusBits
        #define ODL_errorStatusBits_stringLength        10

/*2101, Data Type: UNSIGNED8 */
        #define OD_CANNodeID                            CO_OD_RAM.CANNodeID

/*2102, Data Type: UNSIGNED16 */
        #define OD_CANBitRate                            CO_OD_RAM.CANBitRate

/*2103, Data Type: UNSIGNED16 */
        #define OD_SYNCCounter                          CO_OD_RAM.SYNCCounter

/*2104, Data Type: UNSIGNED16 */
        #define OD_SYNCTime                              CO_OD_RAM.SYNCTime

```

```

/*2106, Data Type: UNSIGNED32 */
    #define OD_powerOnCounter
CO_OD_RAM.powerOnCounter

/*2107, Data Type: UNSIGNED16, Array[5] */
    #define OD_performance                                CO_OD_RAM.performance
    #define ODL_performance_arrayLength                  5
    #define ODA_performance_cyclesPerSecond             0
    #define ODA_performance_timerCycleTime              1
    #define ODA_performance_timerCycleMaxTime           2
    #define ODA_performance_mainCycleTime               3
    #define ODA_performance_mainCycleMaxTime            4

/*2108, Data Type: INTEGER16, Array[1] */
    #define OD_temperature                                CO_OD_RAM.temperature
    #define ODL_temperature_arrayLength                  1
    #define ODA_temperature_mainPCB                      0

/*2109, Data Type: INTEGER16, Array[1] */
    #define OD_voltage                                    CO_OD_RAM.voltage
    #define ODL_voltage_arrayLength                      1
    #define ODA_voltage_mainPCBSupply                    0

/*2110, Data Type: INTEGER32, Array[16] */
    #define OD_variableInt32
CO_OD_RAM.variableInt32
    #define ODL_variableInt32_arrayLength                16
    #define ODA_variableInt32_int32                      0

/*2111, Data Type: INTEGER32, Array[16] */
    #define OD_variableROM_Int32
CO_OD_RAM.variableROM_Int32
    #define ODL_variableROM_Int32_arrayLength            16
    #define ODA_variableROM_Int32_int32                  0

/*2112, Data Type: INTEGER32, Array[16] */
    #define OD_variableNV_Int32
CO_OD_RAM.variableNV_Int32
    #define ODL_variableNV_Int32_arrayLength              16
    #define ODA_variableNV_Int32_int32                    0

/*2120, Data Type: testVar_t */
    #define OD_testVar                                    CO_OD_RAM.testVar

/*2130, Data Type: time_t */
    #define OD_time                                        CO_OD_RAM.time

/*2301, Data Type: traceConfig_t */
    #define OD_traceConfig                                CO_OD_RAM.traceConfig

/*2400, Data Type: UNSIGNED8 */
    #define OD_traceEnable                                CO_OD_RAM.traceEnable

/*2401, Data Type: trace_t */
    #define OD_trace                                        CO_OD_RAM.trace

/*6010, Data Type: UNSIGNED16 */
    #define OD_IGUS_D1Controlword1
CO_OD_RAM.IGUS_D1Controlword1

/*6011, Data Type: UNSIGNED16 */
    #define OD_IGUS_D1Statusword1
CO_OD_RAM.IGUS_D1Statusword1

```

```
/*6012, Data Type: INTEGER32 */
    #define OD_IGUS_D1PositionActualValue1
CO_OD_RAM.IGUS_D1PositionActualValue1

/*6013, Data Type: INTEGER32 */
    #define OD_IGUS_D1VelocityActualValue1
CO_OD_RAM.IGUS_D1VelocityActualValue1

/*6014, Data Type: INTEGER32 */
    #define OD_IGUS_D1TargetPosition1
CO_OD_RAM.IGUS_D1TargetPosition1

/*6015, Data Type: INTEGER32 */
    #define OD_IGUS_D1TargetVelocity1
CO_OD_RAM.IGUS_D1TargetVelocity1

/*6020, Data Type: UNSIGNED16 */
    #define OD_IGUS_D1Controlword2
CO_OD_RAM.IGUS_D1Controlword2

/*6021, Data Type: UNSIGNED16 */
    #define OD_IGUS_D1Statusword2
CO_OD_RAM.IGUS_D1Statusword2

/*6022, Data Type: INTEGER32 */
    #define OD_IGUS_D1PositionActualValue2
CO_OD_RAM.IGUS_D1PositionActualValue2

/*6023, Data Type: INTEGER32 */
    #define OD_IGUS_D1VelocityActualValue2
CO_OD_RAM.IGUS_D1VelocityActualValue2

/*6024, Data Type: INTEGER32 */
    #define OD_IGUS_D1TargetPosition2
CO_OD_RAM.IGUS_D1TargetPosition2

/*6025, Data Type: INTEGER32 */
    #define OD_IGUS_D1TargetVelocity2
CO_OD_RAM.IGUS_D1TargetVelocity2

/*6030, Data Type: UNSIGNED16 */
    #define OD_IGUS_D1Controlword3
CO_OD_RAM.IGUS_D1Controlword3

/*6031, Data Type: UNSIGNED16 */
    #define OD_IGUS_D1Statusword3
CO_OD_RAM.IGUS_D1Statusword3

/*6032, Data Type: INTEGER32 */
    #define OD_IGUS_D1PositionActualValue3
CO_OD_RAM.IGUS_D1PositionActualValue3

/*6033, Data Type: INTEGER32 */
    #define OD_IGUS_D1VelocityActualValue3
CO_OD_RAM.IGUS_D1VelocityActualValue3

/*6034, Data Type: INTEGER32 */
    #define OD_IGUS_D1TargetPosition3
CO_OD_RAM.IGUS_D1TargetPosition3

/*6035, Data Type: INTEGER32 */
    #define OD_IGUS_D1TargetVelocity3
CO_OD_RAM.IGUS_D1TargetVelocity3
```

```
/*6040, Data Type: UNSIGNED16 */
    #define OD_IGUS_D1Controlword4
CO_OD_RAM.IGUS_D1Controlword4

/*6041, Data Type: UNSIGNED16 */
    #define OD_IGUS_D1Statusword4
CO_OD_RAM.IGUS_D1Statusword4

/*6042, Data Type: INTEGER32 */
    #define OD_IGUS_D1PositionActualValue4
CO_OD_RAM.IGUS_D1PositionActualValue4

/*6043, Data Type: INTEGER32 */
    #define OD_IGUS_D1VelocityActualValue4
CO_OD_RAM.IGUS_D1VelocityActualValue4

/*6044, Data Type: INTEGER32 */
    #define OD_IGUS_D1TargetPosition4
CO_OD_RAM.IGUS_D1TargetPosition4

/*6045, Data Type: INTEGER32 */
    #define OD_IGUS_D1TargetVelocity4
CO_OD_RAM.IGUS_D1TargetVelocity4

/*6050, Data Type: UNSIGNED16 */
    #define OD_IGUS_D1Controlword5
CO_OD_RAM.IGUS_D1Controlword5

/*6051, Data Type: UNSIGNED16 */
    #define OD_IGUS_D1Statusword5
CO_OD_RAM.IGUS_D1Statusword5

/*6052, Data Type: INTEGER32 */
    #define OD_IGUS_D1PositionActualValue5
CO_OD_RAM.IGUS_D1PositionActualValue5

/*6053, Data Type: INTEGER32 */
    #define OD_IGUS_D1VelocityActualValue5
CO_OD_RAM.IGUS_D1VelocityActualValue5

/*6054, Data Type: INTEGER32 */
    #define OD_IGUS_D1TargetPosition5
CO_OD_RAM.IGUS_D1TargetPosition5

/*6055, Data Type: INTEGER32 */
    #define OD_IGUS_D1TargetVelocity5
CO_OD_RAM.IGUS_D1TargetVelocity5

/*6060, Data Type: UNSIGNED16 */
    #define OD_IGUS_D1Controlword6
CO_OD_RAM.IGUS_D1Controlword6

/*6061, Data Type: UNSIGNED16 */
    #define OD_IGUS_D1Statusword6
CO_OD_RAM.IGUS_D1Statusword6

/*6062, Data Type: INTEGER32 */
    #define OD_IGUS_D1PositionActualValue6
CO_OD_RAM.IGUS_D1PositionActualValue6

/*6063, Data Type: INTEGER32 */
    #define OD_IGUS_D1VelocityActualValue6
CO_OD_RAM.IGUS_D1VelocityActualValue6
```



```
/*6064, Data Type: INTEGER32 */
    #define OD_IGUS_D1TargetPosition6
CO_OD_RAM.IGUS_D1TargetPosition6

/*6065, Data Type: INTEGER32 */
    #define OD_IGUS_D1TargetVelocity6
CO_OD_RAM.IGUS_D1TargetVelocity6

/*60A0, Data Type: UNSIGNED16 */
    #define OD_motorControlword1
CO_OD_RAM.motorControlword1

/*60A1, Data Type: UNSIGNED16 */
    #define OD_motorStatusword1
CO_OD_RAM.motorStatusword1

/*60A2, Data Type: INTEGER16 */
    #define OD_motorTargetVelocity1
CO_OD_RAM.motorTargetVelocity1

/*60A3, Data Type: INTEGER16 */
    #define OD_motorActualVelocity1
CO_OD_RAM.motorActualVelocity1

/*60B0, Data Type: UNSIGNED16 */
    #define OD_motorControlword2
CO_OD_RAM.motorControlword2

/*60B1, Data Type: UNSIGNED16 */
    #define OD_motorStatusword2
CO_OD_RAM.motorStatusword2

/*60B2, Data Type: INTEGER16 */
    #define OD_motorTargetVelocity2
CO_OD_RAM.motorTargetVelocity2

/*60B3, Data Type: INTEGER16 */
    #define OD_motorActualVelocity2
CO_OD_RAM.motorActualVelocity2

/*60C0, Data Type: UNSIGNED16 */
    #define OD_motorControlword3
CO_OD_RAM.motorControlword3

/*60C1, Data Type: UNSIGNED16 */
    #define OD_motorStatusword3
CO_OD_RAM.motorStatusword3

/*60C2, Data Type: INTEGER16 */
    #define OD_motorTargetVelocity3
CO_OD_RAM.motorTargetVelocity3

/*60C3, Data Type: INTEGER16 */
    #define OD_motorActualVelocity3
CO_OD_RAM.motorActualVelocity3

/*60D0, Data Type: UNSIGNED16 */
    #define OD_motorControlword4
CO_OD_RAM.motorControlword4

/*60D1, Data Type: UNSIGNED16 */
    #define OD_motorStatusword4
CO_OD_RAM.motorStatusword4
```

```
/*60D2, Data Type: INTEGER16 */
    #define OD_motorTargetVelocity4
CO_OD_RAM.motorTargetVelocity4

/*60D3, Data Type: INTEGER16 */
    #define OD_motorActualVelocity4
CO_OD_RAM.motorActualVelocity4

#endif
// clang-format on
```

## Fails "communicatorApp.c"

```

/*****
MPLAB Harmony Application Source File

```

```

Company:
  Microchip Technology Inc.

```

```

File Name:
  communicatorapp.c

```

```

Summary:
  This file contains the source code for the MPLAB Harmony application.

```

```

Description:
  This file contains the source code for the MPLAB Harmony application. It
  implements the logic of the application's state machine and it may call
  API routines of other MPLAB Harmony modules in the system, such as drivers,
  system services, and middleware. However, it does not call any of the
  system interfaces (such as the "Initialize" and "Tasks" functions) of any of
  the modules in the system or make any assumptions about when those functions
  are called. That is the responsibility of the configuration-specific system
  files.

```

```

*****/

```

```

// DOM-IGNORE-BEGIN
/*****
Copyright (c) 2013-2014 released Microchip Technology Inc. All rights reserved.

```

Microchip licenses to you the right to use, modify, copy and distribute Software only when embedded on a Microchip microcontroller or digital signal controller that is integrated into your product or third party product (pursuant to the sublicense terms in the accompanying license agreement).

You should refer to the license agreement accompanying this Software for additional information regarding your rights and obligations.

```

SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND,
EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF
MERCHANTABILITY, TITLE, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE.
IN NO EVENT SHALL MICROCHIP OR ITS LICENSORS BE LIABLE OR OBLIGATED UNDER
CONTRACT, NEGLIGENCE, STRICT LIABILITY, CONTRIBUTION, BREACH OF WARRANTY, OR
OTHER LEGAL EQUITABLE THEORY ANY DIRECT OR INDIRECT DAMAGES OR EXPENSES
INCLUDING BUT NOT LIMITED TO ANY INCIDENTAL, SPECIAL, INDIRECT, PUNITIVE OR
CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF PROCUREMENT OF
SUBSTITUTE GOODS, TECHNOLOGY, SERVICES, OR ANY CLAIMS BY THIRD PARTIES
(INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF), OR OTHER SIMILAR COSTS.

```

```

*****/

```

```

// DOM-IGNORE-END

```

```

// *****/
// *****/
// Section: Included Files
// *****/
// *****/

```

```

#include "communicatorapp.h"
#include "UDPComm/PIC32_Harmony/udpcomm.h"

```

```

// *****/
// *****/
// Section: Global Data Definitions
// *****/

```

```

// *****
// *****
/* Application Data

Summary:
    Holds application data

Description:
    This structure holds the application's data.

Remarks:
    This structure should be initialized by the APP_Initialize function.

    Application strings and buffers are be defined outside this structure.
*/

COMMUNICATORAPP_DATA commAppData;

// *****
// *****
// Section: Application Callback Functions
// *****
// *****

/* TODO: Add any necessary callback functions.
*/

// *****
// *****
// Section: Application Local Functions
// *****
// *****

/* TODO: Add any necessary local functions.
*/

// *****
// *****
// Section: Application Initialization and State Machine Functions
// *****
// *****

/*****
Function:
    void COMMUNICATORAPP_Initialize ( void )

Remarks:
    See prototype in communicatorapp.h.
*/

void COMMUNICATORAPP_Initialize ( void )
{
    UDPCOMM_Initialise();

    /* Place the App state machine in its initial state. */
    commAppData.state = COMMUNICATORAPP_WAIT_INIT;
    commAppData.Flags.Word = 0;

    /* TODO: Initialize your application's state machine and other
    * parameters.
    */
}

```

```

}

/*****
Function:
    void COMMUNICATORAPP_Tasks ( void )

Remarks:
    See prototype in communicatorapp.h.
*/

void COMMUNICATORAPP_Tasks ( void )
{
    SYS_STATUS          tcpipStat;
    int                 i, nNets, byteCnt, charCnt;
    int16_t             p1, p2, p3, p4, p5;
    uint8_t *ptr, *c;
    int8_t *s;
    TCPIP_NET_HANDLE    netH;
    const char          *netName;
    IPV4_ADDR           ipAddr;
    bool res;
    RONIN_COMM_INTERFACE_TYPE comInterfaceType;

    UDPCOMM_Tasks();
    /* Check the application's current state. */
    switch ( commAppData.state )
    {
        /* Application's initial state. */
        case COMMUNICATORAPP_WAIT_INIT:
        {
            if (UDPCOMM_GetState() == UDPCOMM_READY)
                commAppData.state = COMMUNICATORAPP_IDLE;
            break;
        }

        case COMMUNICATORAPP_IDLE:

            if (byteCnt = UDPCOMM_GetBytes( commAppData.RemoteCommandBuff,
                                           sizeof(commAppData.RemoteCommandBuff)))
            {
                commAppData.RemoteCommandBuff[byteCnt] = '\0';
                RONIN_COMMAND_PROC_ParseString(commAppData.RemoteCommandBuff);
            }

            charCnt = sprintf(commAppData.TxBuff, "%d;", commAppData.PacketsSent);
            comInterfaceType = RONIN_COMMAND_PROC_GetStringToSend(commAppData.TxBuff +
charCnt);
            switch (comInterfaceType)
            {
                case RONIN_COMM_NO_DATA:
                    break;
                case RONIN_COMM_INTERFACE_FAST:
                    UDPCOMM_PutBytes(commAppData.TxBuff, strlen(commAppData.TxBuff));
                    commAppData.PacketsSent++;
                    break;
                case RONIN_COMM_INTERFACE_RELIABLE:
                    //Not implemented
                    break;
            }
            break;
        /* TODO: implement your application state machine.*/
    }
}

```

```
/* The default state should never be executed. */
default:
{
    /* TODO: Handle error in application's state machine. */
    break;
}
}
if (commAppData.state > COMMUNICATORAPP_WAIT_FOR_CONNECTION)
{
//     if (!TCPIP_UDP_IsConnected(commAppData.ClientSocketHandle))
//     {
//         SYS_CONSOLE_MESSAGE("Connection closed!\r\n");
//         commAppData.state = COMMUNICATORAPP_WAIT_FOR_IP;
//     }
}
}

/*****
End of File
*/
```

## Fails "communicatorApp.h"

```

/*****
MPLAB Harmony Application Header File

```

```

Company:
  Microchip Technology Inc.

```

```

File Name:
  communicatorapp.h

```

```

Summary:
  This header file provides prototypes and definitions for the application.

```

```

Description:
  This header file provides function prototypes and data type definitions for
  the application. Some of these are required by the system (such as the
  "APP_Initialize" and "APP_Tasks" prototypes) and some of them are only used
  internally by the application (such as the "APP_STATES" definition). Both
  are defined here for convenience.

```

```

*****/

```

```

//DOM-IGNORE-BEGIN

```

```

/*****
Copyright (c) 2013-2014 released Microchip Technology Inc. All rights reserved.

```

Microchip licenses to you the right to use, modify, copy and distribute Software only when embedded on a Microchip microcontroller or digital signal controller that is integrated into your product or third party product (pursuant to the sublicense terms in the accompanying license agreement).

You should refer to the license agreement accompanying this Software for additional information regarding your rights and obligations.

SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY, TITLE, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL MICROCHIP OR ITS LICENSORS BE LIABLE OR OBLIGATED UNDER CONTRACT, NEGLIGENCE, STRICT LIABILITY, CONTRIBUTION, BREACH OF WARRANTY, OR OTHER LEGAL EQUITABLE THEORY ANY DIRECT OR INDIRECT DAMAGES OR EXPENSES INCLUDING BUT NOT LIMITED TO ANY INCIDENTAL, SPECIAL, INDIRECT, PUNITIVE OR CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF PROCUREMENT OF SUBSTITUTE GOODS, TECHNOLOGY, SERVICES, OR ANY CLAIMS BY THIRD PARTIES (INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF), OR OTHER SIMILAR COSTS.

```

*****/
//DOM-IGNORE-END

```

```

#ifndef _COMMUNICATORAPP_H
#define _COMMUNICATORAPP_H

```

```

// *****/
// *****/
// Section: Included Files
// *****/
// *****/

```

```

#include <stdint.h>
#include <stdbool.h>
#include <stddef.h>
#include <stdlib.h>
#include "system_config.h"
#include "system_definitions.h"
#include "M2M_interface.h"

```

```

// DOM-IGNORE-BEGIN
#ifdef __cplusplus // Provide C++ Compatibility

extern "C" {

#endif
// DOM-IGNORE-END

// *****
// *****
// Section: Type Definitions
// *****
// *****

/* Application states

Summary:
    Application states enumeration

Description:
    This enumeration defines the valid application states. These states
    determine the behavior of the application at various times.
*/

typedef enum
{
    /* Application's state machine's initial state. */
    COMMUNICATORAPP_WAIT_INIT=0,
    COMMUNICATORAPP_WAIT_FOR_IP,
    COMMUNICATORAPP_WAIT_FOR_CONNECTION,
    COMMUNICATORAPP_IDLE

    /* TODO: Define states used by the application state machine. */
} COMMUNICATORAPP_STATES;

// *****
/* Application Data

Summary:
    Holds application data

Description:
    This structure holds the application's data.

Remarks:
    Application strings and buffers are be defined outside this structure.
*/

typedef struct
{
    /* The application's current state */
    COMMUNICATORAPP_STATES state;

    IPV4_ADDR    dwLastIP;
    IPV4_ADDR DataServerIP;
    UDP_PORT DataServerPort;
    UDP_PORT DVSpatformUDPServerPort;
    UDP_PORT DVSpatformTCPServerPort;
    UDP_SOCKET UDPServerSocketHandle;
    UDP_SOCKET TCPServerSocketHandle;
    uint16_t DataLength;

```



```

uint32_t PacketsSent;
uint32_t PacketsReceived;
uint8_t *DataPtr;
uint8_t RemoteCommandBuff[4096];
int8_t TxBuff[512];
union
{
    struct
    {
        uint32_t TxDataReady:1;
        uint32_t EnableTelemetryUDP:1;
        uint32_t EnableTelemetryConsole:1;
    };
    uint32_t Word;
} Flags;

/* TODO: Define any additional data used by the application. */
} COMMUNICATORAPP_DATA;

// *****
// *****
// Section: Application Callback Routines
// *****
// *****
/* These routines are called by drivers when certain events occur.
*/

// *****
// *****
// Section: Application Initialization and State Machine Functions
// *****
// *****

/*****
Function:
    void COMMUNICATORAPP_Initialize ( void )

Summary:
    MPLAB Harmony application initialization routine.

Description:
    This function initializes the Harmony application. It places the
    application in its initial state and prepares it to run so that its
    APP_Tasks function can be called.

Precondition:
    All other system initialization routines should be called before calling
    this routine (in "SYS_Initialize").

Parameters:
    None.

Returns:
    None.

Example:
    <code>
    COMMUNICATORAPP_Initialize();
    </code>

Remarks:
    This routine must be called from the SYS_Initialize function.

```

```

*/
void COMMUNICATORAPP_Initialize ( void );

/*****
Function:
    void COMMUNICATORAPP_Tasks ( void )

Summary:
    MPLAB Harmony Demo application tasks function

Description:
    This routine is the Harmony Demo application's tasks function. It
    defines the application's state machine and core logic.

Precondition:
    The system and application initialization ("SYS_Initialize") should be
    called before calling this.

Parameters:
    None.

Returns:
    None.

Example:
    <code>
    COMMUNICATORAPP_Tasks();
    </code>

Remarks:
    This routine must be called from SYS_Tasks() routine.
*/
void COMMUNICATORAPP_Tasks( void );

#endif /* _COMMUNICATORAPP_H */

//DOM-IGNORE-BEGIN
#ifdef __cplusplus
}
#endif
//DOM-IGNORE-END

/*****
End of File
*/

```

## Fails "igusD1.c"

```

#include "igusD1.h"

#define IGUSD1_POSITIONING_MODE_ABSOLUTE 0
#define IGUSD1_POSITIONING_MODE_RELATIVE 1

typedef struct
{
    void* ControlwordPtr;

    CO_OBJECT *Settings;
} IGUSD1_INIT_DATA;

void IGUSD1_Initialize( IGUSD1_t *id,
                       int8_t nodeId,
                       CO_OBJECT *settings,
                       void* ControlwordPtr,
                       void* StatuswordPtr,
                       void* TargetPosPtr)
{
    id->Status = IGUSD1_Init;
    id->Tag = nodeId;
    id->Controlword = ControlwordPtr;
    id->StatusWord = StatuswordPtr;
    id->TargetPos = TargetPosPtr;
    id->Settings = settings;
    id->CANnodeID = nodeId;
    id->ErrorCount = 0;
    id->SettingCount = IGUSD1_SETTING_CO_OBJECT_COUNT;
    memset(id->ProcessTimers, 0, IGUSD1_PROCESS_TIMER_COUNT * sizeof(uint32_t));
}

void IGUSD1_SetPos(IGUSD1_t *id, int32_t pos)
{
    if (id->Status == IGUSD1_ServiceTasks)
    {
        id->Controlword->SwitchOn = 1;
        id->Controlword->EnableVoltage = 1;
        id->Controlword->QuickStop = 1;
        id->Controlword->EnableOperation = 1;
        id->Controlword->StartMove = 0; /* Do not start yet, data adoption by
                                         * IGUSD1 takes time, starting performed
                                         * in IGUSD1_Tasks function */
        id->Controlword->PositioningMode = IGUSD1_POSITIONING_MODE_ABSOLUTE;
        id->ProcessTimers[0] = 0;
        *id->TargetPos = pos;
        id->Status = IGUSD1_MoveTest;
    }
}

bool IGUSD1_IsTargetReached(IGUSD1_t *id)
{
    bool ret = false;
    if (id->Status == IGUSD1_ServiceTasks && id->StatusWord->TargetReached)
    {
        ret = true;
    }
    return ret;
}

```

```

}

//Works only after power up
void IGUSD1_StartHoming(IGUSD1_t *id)
{
    id->Flags.HomingRequired = 1;
}

void IGUSD1_Tasks(IGUSD1_t *id, uint32_t elapsedMiliSec)
{
    CANOPENAPP_SDO_ClientOpeResult_t res;
    int i;
    for (i = 0; i < IGUSD1_PROCESS_TIMER_COUNT; i++)
    {
        id->ProcessTimers[i] += elapsedMiliSec;
    }
    switch(id->Status)
    {
        case IGUSD1_Init:
            id->Flags.HomingRequired = 1;
            id->Status = IGUSD1_InitSetup;
        case IGUSD1_InitSetup:
            if (id->ProcessTimers[1] < 1000)
                break;
            if (id->Flags.HomingRequired)
            {
                id->Settings[1].Value = 6; //Homing mode
            }
            else
            {
                id->Settings[1].Value = 1; //Profile Position Mode
            }
        case IGUSD1_Setup:
            if (CANOPENAPP_TRANSFER_Initiate( id->CANnodeID,
                                              id->Settings,
                                              id->SettingCount,
                                              CANOPENAPP_SDO_TRANSFER_DOWNLOAD, true) ==
                CANOPENAPP_SDO_ClientOpeScheduled)
                id->Status = IGUSD1_ConfigSetting;
            break;
        case IGUSD1_ConfigSetting:
            res = CANOPENAPP_TRANSFER_Result(id->Settings);
            switch (res)
            {
                case CANOPENAPP_SDO_ClientOpeResultOK:
                    SYS_CONSOLE_PRINT("IGUS D1 [%d]: ", id->Tag);
                    SYS_CONSOLE_PRINT("Enabling voltage\r\n");
                    id->Controlword->EnableVoltage = 1;
                    id->Controlword->QuickStop = 1;           /*Send control word Enable
voltage
                                                                * and Quick stop to enter
                                                                * "Ready to switch on" state
*/
                    id->Status = IGUSD1_EnablingVoltage;
                    break;
                case CANOPENAPP_SDO_ClientCANError:
                case CANOPENAPP_SDO_ClientArgError:
                    id->Status = IGUSD1_Error;
                    break;
                case CANOPENAPP_SDO_ClientOpeScheduled:
                default:
                    break;
            }
        }
    }
    break;
}

```

```

case IGUSD1_EnablingVoltage:
    if (id->StatusWord->ReadyToSwitchOn)
    {
        SYS_CONSOLE_PRINT("IGUS D1 [%d]: ", id->Tag);
        SYS_CONSOLE_PRINT("Switching on\r\n");
        id->Controlword->SwitchOn = 1;
        id->Controlword->EnableVoltage = 1;
        id->Controlword->QuickStop = 1;          /*Send control word Enable voltage
                                                * and Quick stop
                                                * and Switch on to enter
                                                * "Switched on" state */

        id->Status = IGUSD1_SwitchingOn;
    }
    //TODO: else add timeout?
    break;
case IGUSD1_SwitchingOn:
    if (id->StatusWord->SwitchedOn)
    {
        SYS_CONSOLE_PRINT("IGUS D1 [%d]: ", id->Tag);
        SYS_CONSOLE_PRINT("Enabling operation\r\n");
        id->Controlword->SwitchOn = 1;
        id->Controlword->EnableVoltage = 1;
        id->Controlword->EnableOperation = 1;
        id->Controlword->QuickStop = 1;          /*Send control word Enable voltage
                                                * and Quick stop
                                                * and Switch on
                                                * and Enable operation to enter
                                                * "Operation enabled" state */

        id->Status = IGUSD1_EnablingOperation;
    }
    break;
case IGUSD1_EnablingOperation:
    if (id->StatusWord->OperationEnabled)
    {
        if (id->Flags.HomingRequired)
        {
            SYS_CONSOLE_PRINT("IGUS D1 [%d]: ", id->Tag);
            SYS_CONSOLE_PRINT("Homing starting\r\n");
            id->Status = IGUSD1_HomingStarting;
            id->ProcessTimers[0] = 0;
            id->Flags.HomingRequired = 0;
        }
        else
        {
            SYS_CONSOLE_PRINT("IGUS D1 [%d]: ", id->Tag);
            SYS_CONSOLE_PRINT("Operation enabled\r\n");
            id->Status = IGUSD1_ServiceTasks;
        }
    }
    break;
case IGUSD1_ServiceTasks:
    if (id->Flags.HomingRequired)
    {
        id->Status = IGUSD1_InitSetup;
    }
    break;
case IGUSD1_HomingStarting:
    if (id->ProcessTimers[0] > 10)
    {
        id->Controlword->SwitchOn = 1;
        id->Controlword->EnableVoltage = 1;
        id->Controlword->QuickStop = 1;
        id->Controlword->EnableOperation = 1;
        id->Controlword->StartMove = 1;          //Start of homing movement
    }

```

```

        id->ProcessTimers[0] = 0;
        id->Status = IGUSD1_HomingProgress;
    }
    break;
case IGUSD1_HomingProgress:
    if (id->ProcessTimers[0] < 50)
        break;
    if (id->StatusWord->TargetReached)
    {
        id->Controlword->StartMove = 0;
        SYS_CONSOLE_PRINT("IGUS D1 [%d]: ", id->Tag);
        SYS_CONSOLE_PRINT("Homing complete\r\n");
        id->Status = IGUSD1_InitSetup;        //Setup normal operation after homing
    }
    break;
case IGUSD1_MoveTest:
    if (id->ProcessTimers[0] > 10)
    {
        id->Controlword->StartMove = 1;        //Start of movement
        id->ProcessTimers[0] = 0;
        id->Status = IGUSD1_Moving;
    }
    break;
case IGUSD1_Moving:
    if (id->StatusWord->SetpointApplied == 1)
    {
        id->Controlword->StartMove = 0;
    }
    if (id->ProcessTimers[0] > 10)
    {
        id->Controlword->StartMove = 0;
        if (id->StatusWord->TargetReached == 1)
        {
            id->Status = IGUSD1_ServiceTasks;
        }
    }
    break;
case IGUSD1_Error:
    if (id->ErrorCount < 100)
    {
        SYS_CONSOLE_PRINT("IGUS D1 [%d]: ", id->Tag);
        SYS_CONSOLE_PRINT("CAN open error, attempt to continue\r\n\r");
        id->ErrorCount++;
        id->Status = IGUSD1_ServiceTasks;
    }
    else if (id->ErrorCount == 100)
    {
        SYS_CONSOLE_PRINT("IGUS D1 [%d]: ", id->Tag);
        SYS_CONSOLE_PRINT("No response\r\n\r");
        id->ErrorCount++;
    }
    break;
}
}
}

```

## Fails "igusD1.h"

```

/*
 * File:   igusD1.h
 * Author: Vitalijs
 *
 * Created on otrdiena, 2020, 14 j?lijs, 12:22
 */

#ifndef IGUSD1_H
#define IGUSD1_H

#ifdef __cplusplus
extern "C" {
#endif

#include <stdint.h>
#include <string.h>
#include <stddef.h>
#include <errno.h>
#include "system_definitions.h"
#include "system_config.h"
#include "osal/osal.h"

#define IGUSD1_SETTING_CO_OBJECT_COUNT 10
#define IGUSD1_PROCESS_TIMER_COUNT 2

typedef enum
{
    IGUSD1_Init = 0,
    IGUSD1_InitSetup,
    IGUSD1_Setup,
    IGUSD1_ConfigSetting,
    IGUSD1_EnablingVoltage,
    IGUSD1_SwitchingOn,
    IGUSD1_EnablingOperation,
    IGUSD1_ServiceTasks,
    IGUSD1_MoveTest,
    IGUSD1_Moving,
    IGUSD1_HomingStarting,
    IGUSD1_HomingProgress,
    IGUSD1_Error
} IGUSD1_STATUS;

typedef struct
{
    uint16_t SwitchOn:1;
    uint16_t EnableVoltage:1;
    uint16_t QuickStop:1;
    uint16_t EnableOperation:1;
    uint16_t StartMove:1;
    uint16_t ModeSpec2:1;
    uint16_t PositioningMode:1;
    uint16_t FaultReset:1;
    uint16_t Halt:1;
    uint16_t ModeSpec4:1;
    uint16_t :6;
} IGUSD1_CONTROLWORD;

typedef struct
{
    uint16_t ReadyToSwitchOn:1;
    uint16_t SwitchedOn:1;
    uint16_t OperationEnabled:1;

```

```

uint16_t Fault:1;
uint16_t VoltageEnable:1;
uint16_t QuickStop:1;
uint16_t SwitchOnDisabled:1;
uint16_t Warning:1;
uint16_t :1;
uint16_t Remote:1;
uint16_t TargetReached:1;
uint16_t InternalLimitActive:1;
uint16_t SetpointApplied:1;
uint16_t :3;

} IGUSD1_STATUSWORD;

typedef struct
{
    IGUSD1_STATUS Status;
    int32_t Tag;
    IGUSD1_CONTROLWORD *Controlword;
    IGUSD1_STATUSWORD *StatusWord;
    uint8_t CANnodeID;

    int32_t *TargetPos;
    CO_OBJECT *Settings;
    uint32_t CurrSetting;
    uint32_t SettingCount;
    uint32_t ErrorCount;

    uint32_t ProcessTimers[IGUSD1_PROCESS_TIMER_COUNT];
    struct
    {
        uint32_t HomingRequired:1;
    } Flags;
} IGUSD1_t;

void IGUSD1_Initialize( IGUSD1_t *id,
                       int8_t nodeId,
                       CO_OBJECT *settings,
                       void* ControlwordPtr,
                       void* StatuswordPtr,
                       void* TargetPosPtr);

void IGUSD1_Tasks(IGUSD1_t *id, uint32_t elapsedMiliSec);
void IGUSD1_SetPos(IGUSD1_t *id, int32_t pos);
void IGUSD1_ReadCANObj(uint16_t address, uint16_t subIdx);
void IGUSD1_StartHoming(IGUSD1_t *id);
bool IGUSD1_IsTargetReached(IGUSD1_t *id);

#ifdef    __cplusplus
}
#endif

#endif    /* IGUSD1_H */

```



## Fails "main.c"

```

/*****
MPLAB Harmony Project Main Source File

Company:
  Microchip Technology Inc.

File Name:
  main.c

Summary:
  This file contains the "main" function for an MPLAB Harmony project.

Description:
  This file contains the "main" function for an MPLAB Harmony project. The
  "main" function calls the "SYS_Initialize" function to initialize the state
  machines of all MPLAB Harmony modules in the system and it calls the
  "SYS_Tasks" function from within a system-wide "super" loop to maintain
  their correct operation. These two functions are implemented in
  configuration-specific files (usually "system_init.c" and "system_tasks.c")
  in a configuration-specific folder under the "src/system_config" folder
  within this project's top-level folder. An MPLAB Harmony project may have
  more than one configuration, each contained within it's own folder under
  the "system_config" folder.
*****/

// DOM-IGNORE-BEGIN
/*****
Copyright (c) 2013-2014 released Microchip Technology Inc. All rights reserved.

//Microchip licenses to you the right to use, modify, copy and distribute
Software only when embedded on a Microchip microcontroller or digital signal
controller that is integrated into your product or third party product
(pursuant to the sublicense terms in the accompanying license agreement).

You should refer to the license agreement accompanying this Software for
additional information regarding your rights and obligations.

SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND,
EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF
MERCHANTABILITY, TITLE, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE.
IN NO EVENT SHALL MICROCHIP OR ITS LICENSORS BE LIABLE OR OBLIGATED UNDER
CONTRACT, NEGLIGENCE, STRICT LIABILITY, CONTRIBUTION, BREACH OF WARRANTY, OR
OTHER LEGAL EQUITABLE THEORY ANY DIRECT OR INDIRECT DAMAGES OR EXPENSES
INCLUDING BUT NOT LIMITED TO ANY INCIDENTAL, SPECIAL, INDIRECT, PUNITIVE OR
CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF PROCUREMENT OF
SUBSTITUTE GOODS, TECHNOLOGY, SERVICES, OR ANY CLAIMS BY THIRD PARTIES
(INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF), OR OTHER SIMILAR COSTS.
*****/
// DOM-IGNORE-END

// *****/
// *****/
// Section: Included Files
// *****/
// *****/

#include <stddef.h>           // Defines NULL
#include <stdbool.h>         // Defines true
#include <stdlib.h>          // Defines EXIT_FAILURE
#include "system/common/sys_module.h"
#include "RONINCommandProc.h" // SYS function prototypes

```

```

// *****
// *****
// Section: Main Entry Point
// *****
// *****

void delay_lms(void)
{
    int i = 40000;
    while(i--);
}

void delay_ms(int ms)
{
    while(ms--)
    {
        delay_lms();
    }
}

int main ( void )
{
    /* Initialize all MPLAB Harmony modules, including application(s). */
    SYS_Initialize ( NULL );

    delay_ms(2000);

    while ( true )
    {
        /* Maintain state machines of all polled MPLAB Harmony modules. */
        SYS_Tasks ( );
    }

    /* Execution should not come here during normal operation */

    return ( EXIT_FAILURE );
}

/*****
End of File
*/

```

## Fails "Ronin\_CNC.c"

```

#include "RONIN_CNC.h"
#include "system_config.h"
#include "system_definitions.h"
#include "igusD1.h"
#include "UDPCComm/PIC32_Harmony/udpcomm.h"
#include "RONIN_laser.h"

/*****/
/*****/
/*****/
int8_t IigusD1NodeIds[] = { 1, 2, 3, 4, 5, 6};

#define RONINCANC_AXIS_COUNT sizeof(IigusD1NodeIds)

CO_OBJECT AxisSettings[RONINCANC_AXIS_COUNT][IGUSD1_SETTING_CO_OBJECT_COUNT] =
{
    {
        { 0x1801, 5, 255, sizeof(uint16_t) }, //Communication parameter (send on demand)
        { 0x6060, 0, 1, sizeof(uint8_t) }, //Profile position mode
        { 0x6081, 0, 200, sizeof(uint32_t) }, //Profile velocity (mm/s)
        { 0x6083, 0, 200, sizeof(uint32_t) }, //Profile acceleration (mm/s^2)
        { 0x6084, 0, 200, sizeof(uint32_t) }, //Profile deceleration (mm/s^2)
        { 0x6092, 1, 70, sizeof(uint32_t) }, //Feed rate (mm)
        { 0x6092, 2, 1, sizeof(uint32_t) }, //Shaft revolutions
        { 0x6099, 1, 5, sizeof(uint32_t) }, //Switch Search Speed (mm/s)
        { 0x6099, 2, 5, sizeof(uint32_t) }, //Zero Search Speed (mm/s)
        { 0x609A, 0, 5, sizeof(uint32_t) }, //Homing acceleration/deceleration
    },
    {
        { 0x1801, 5, 255, sizeof(uint16_t) }, //Communication parameter (send on demand)
        { 0x6060, 0, 1, sizeof(uint8_t) }, //Profile position mode
        { 0x6081, 0, 25, sizeof(uint32_t) }, //Profile velocity (mm/s)
        { 0x6083, 0, 25, sizeof(uint32_t) }, //Profile acceleration (mm/s^2)
        { 0x6084, 0, 200, sizeof(uint32_t) }, //Profile deceleration (mm/s^2)
        { 0x6092, 1, 70, sizeof(uint32_t) }, //Feed rate (mm)
        { 0x6092, 2, 1, sizeof(uint32_t) }, //Shaft revolutions
        { 0x6099, 1, 5, sizeof(uint32_t) }, //Switch Search Speed (mm/s)
        { 0x6099, 2, 5, sizeof(uint32_t) }, //Zero Search Speed (mm/s)
        { 0x609A, 0, 5, sizeof(uint32_t) }, //Homing acceleration/deceleration
    },
    {
        { 0x1801, 5, 255, sizeof(uint16_t) }, //Communication parameter (send on demand)
        { 0x6060, 0, 1, sizeof(uint8_t) }, //Profile position mode
        { 0x6081, 0, 200, sizeof(uint32_t) }, //Profile velocity (mm/s)
        { 0x6083, 0, 200, sizeof(uint32_t) }, //Profile acceleration (mm/s^2)
        { 0x6084, 0, 200, sizeof(uint32_t) }, //Profile deceleration (mm/s^2)
        { 0x6092, 1, 70, sizeof(uint32_t) }, //Feed rate (mm)
        { 0x6092, 2, 1, sizeof(uint32_t) }, //Shaft revolutions
        { 0x6099, 1, 5, sizeof(uint32_t) }, //Switch Search Speed (mm/s)
        { 0x6099, 2, 5, sizeof(uint32_t) }, //Zero Search Speed (mm/s)
        { 0x609A, 0, 5, sizeof(uint32_t) }, //Homing acceleration/deceleration
    },
    {
        { 0x1801, 5, 255, sizeof(uint16_t) }, //Communication parameter (send on demand)
        { 0x6060, 0, 1, sizeof(uint8_t) }, //Profile position mode
        { 0x6081, 0, 50, sizeof(uint32_t) }, //Profile velocity (mm/s)
        { 0x6083, 0, 100, sizeof(uint32_t) }, //Profile acceleration (mm/s^2)
    },
}

```

```

        { 0x6084, 0, 100, sizeof(uint32_t) }, //Profile deceleration (mm/s^2)
        { 0x6092, 1, 70, sizeof(uint32_t) }, //Feed rate (mm)
        { 0x6092, 2, 1, sizeof(uint32_t) }, //Shaft revolutions
        { 0x6099, 1, 5, sizeof(uint32_t) }, //Switch Search Speed (mm/s)
        { 0x6099, 2, 5, sizeof(uint32_t) }, //Zero Search Speed (mm/s)
        { 0x609A, 0, 5, sizeof(uint32_t) }, //Homing acceleration/deceleration
(mm/s^2)
    },
    {
        { 0x1801, 5, 255, sizeof(uint16_t) }, //Communication parameter (send on demand)
        { 0x6060, 0, 1, sizeof(uint8_t) }, //Profile position mode
        { 0x6081, 0, 40, sizeof(uint32_t) }, //Profile velocity (mm/s)
        { 0x6083, 0, 100, sizeof(uint32_t) }, //Profile acceleration (mm/s^2)
        { 0x6084, 0, 100, sizeof(uint32_t) }, //Profile deceleration (mm/s^2)
        { 0x6092, 1, 70, sizeof(uint32_t) }, //Feed rate (mm)
        { 0x6092, 2, 1, sizeof(uint32_t) }, //Shaft revolutions
        { 0x6099, 1, 5, sizeof(uint32_t) }, //Switch Search Speed (mm/s)
        { 0x6099, 2, 5, sizeof(uint32_t) }, //Zero Search Speed (mm/s)
        { 0x609A, 0, 5, sizeof(uint32_t) }, //Homing acceleration/deceleration
(mm/s^2)
    },
    {
        { 0x1801, 5, 255, sizeof(uint16_t) }, //Communication parameter (send on demand)
        { 0x6060, 0, 1, sizeof(uint8_t) }, //Profile position mode
        { 0x6081, 0, 50, sizeof(uint32_t) }, //Profile velocity (mm/s)
        { 0x6083, 0, 100, sizeof(uint32_t) }, //Profile acceleration (mm/s^2)
        { 0x6084, 0, 100, sizeof(uint32_t) }, //Profile deceleration (mm/s^2)
        { 0x6092, 1, 70, sizeof(uint32_t) }, //Feed rate (mm)
        { 0x6092, 2, 1, sizeof(uint32_t) }, //Shaft revolutions
        { 0x6099, 1, 5, sizeof(uint32_t) }, //Switch Search Speed (mm/s)
        { 0x6099, 2, 5, sizeof(uint32_t) }, //Zero Search Speed (mm/s)
        { 0x609A, 0, 5, sizeof(uint32_t) }, //Homing acceleration/deceleration
(mm/s^2)
    }
};

```

```

void *COPointers[RONINCANC_AXIS_COUNT][3] =
{
    {
        (void*)&CO_OD_RAM.IGUS_D1Controlword1,
        (void*)&CO_OD_RAM.IGUS_D1Statusword1,
        (void*)&CO_OD_RAM.IGUS_D1TargetPosition1,
    },
    {
        (void*)&CO_OD_RAM.IGUS_D1Controlword2,
        (void*)&CO_OD_RAM.IGUS_D1Statusword2,
        (void*)&CO_OD_RAM.IGUS_D1TargetPosition2,
    },
    {
        (void*)&CO_OD_RAM.IGUS_D1Controlword3,
        (void*)&CO_OD_RAM.IGUS_D1Statusword3,
        (void*)&CO_OD_RAM.IGUS_D1TargetPosition3,
    },
    {
        (void*)&CO_OD_RAM.IGUS_D1Controlword4,
        (void*)&CO_OD_RAM.IGUS_D1Statusword4,
        (void*)&CO_OD_RAM.IGUS_D1TargetPosition4,
    },
    {
        (void*)&CO_OD_RAM.IGUS_D1Controlword5,
        (void*)&CO_OD_RAM.IGUS_D1Statusword5,
        (void*)&CO_OD_RAM.IGUS_D1TargetPosition5,
    },
    {

```

```

        (void*)&CO_OD_RAM.IGUS_D1Controlword6,
        (void*)&CO_OD_RAM.IGUS_D1Statusword6,
        (void*)&CO_OD_RAM.IGUS_D1TargetPosition6,
    }
};
/*****/

#define RONINCANC_MILISECOND_TIMER_MAX 0xFFFFFFFF

#define RONINCANC_MAX_LASERPOINTS 1024

typedef struct
{
    int32_t X;
    int32_t Y;
    int32_t Z;
    struct
    {
        /* If 1, laser will be on when heading to this point */
        uint32_t LaserOn:1;
    } Flags;
} RONINCANC_LASERPOINT;

typedef enum
{
    RONINCANC_Init = 0,
    RONINCANC_SetupAxis,
    RONINCANC_ServiceTasks
} RONINCANC_STATUS;

typedef struct
{
    IGUSD1_t AxisMotors[RONINCANC_AXIS_COUNT];
    uint32_t MilisecondTimer;
    SYS_TMR_HANDLE TmrlmsHandle;
    RONINCANC_LASERPOINT TotalPoints[RONINCANC_MAX_LASERPOINTS];
    uint32_t PointCount;
    uint32_t CurrentPoint;
    RONINCANC_STATUS Status;
    struct
    {
        uint32_t LaserAxisMoveStart:1;
        uint32_t MillAxisMoveStart:1;
    } Flags;
} RONINCANC_DATA;

RONINCANC_DATA _tcnc;
void RONINCANC_Tmrlms_Callback ( uintptr_t context, uint32_t currTick );

void RONINCANC_Tmrlms_Callback ( uintptr_t context, uint32_t currTick )
{
    _tcnc.MilisecondTimer++;
}

void RONINCANC_Initialize(void)
{
    _tcnc.MilisecondTimer = 0;
    _tcnc.TmrlmsHandle = DRV_HANDLE_INVALID;
    _tcnc.Status = RONINCANC_Init;
    _tcnc.CurrentPoint = 0;
    _tcnc.PointCount = 0;
    _tcnc.Flags.LaserAxisMoveStart = 0;
    _tcnc.Flags.MillAxisMoveStart = 0;
}

```

```

void _addLaserPoint(int32_t x, int32_t y, int32_t z, bool powerOn)
{
    if (_tcnc.PointCount < RONINCANC_MAX_LASERPOINTS)
    {
        _tcnc.TotalPoints[_tcnc.PointCount].X = x;
        _tcnc.TotalPoints[_tcnc.PointCount].Y = y;
        _tcnc.TotalPoints[_tcnc.PointCount].Z = z;
        if (powerOn)
            _tcnc.TotalPoints[_tcnc.PointCount].Flags.LaserOn = 1;
        else
            _tcnc.TotalPoints[_tcnc.PointCount].Flags.LaserOn = 0;
        _tcnc.PointCount++;
    }
}

void RONINCNC_SetLaserPos(int32_t x, int32_t y, int32_t z)
{
    if (_tcnc.PointCount == 0)
    {
        _tcnc.CurrentPoint = 0;
        _addLaserPoint(x, y, z, false);
    }
}

void _setLaserPos(int32_t x, int32_t y, int32_t z)
{
    IGUSD1_SetPos(&(_tcnc.AxisMotors[0]), x);
    IGUSD1_SetPos(&(_tcnc.AxisMotors[1]), y);
    IGUSD1_SetPos(&(_tcnc.AxisMotors[2]), z);
    _tcnc.Flags.LaserAxisMoveStart = 1;
}

void RONINCNC_SetMillPos(int32_t x, int32_t y, int32_t z)
{
    _setMillPos(x, y, z);
}

void _setMillPos(int32_t x, int32_t y, int32_t z)
{
    IGUSD1_SetPos(&(_tcnc.AxisMotors[3]), x);
    IGUSD1_SetPos(&(_tcnc.AxisMotors[4]), y);
    IGUSD1_SetPos(&(_tcnc.AxisMotors[5]), z);
    _tcnc.Flags.MillAxisMoveStart = 1;
}

/*
 * Draw rectangle with laser on at constant height z
 * x,y -> x+w,y
 * x+w -> x+w,y+h
 * x+w,y+h -> x,y+h
 * x,y+h -> x,y
 */
void RONINCNC_DrawLaserPattern1(int32_t x, int32_t y, int32_t z, int32_t w, int32_t h)
{
    if (_tcnc.PointCount == 0)
    {
        _tcnc.CurrentPoint = 0;
        _addLaserPoint(x, y, z, false);
        _addLaserPoint(x, y + h, z, true);
        _addLaserPoint(x + w, y + h, z, true);
        _addLaserPoint(x + w, y, z, true);
        _addLaserPoint(x, y, z, true);
        _addLaserPoint(x, y, z, false);
    }
}

```

```

    }
}

#define PATTERN2_STEP 3

void RONINCNC_DrawLaserPattern2(int32_t x, int32_t y, int32_t z, int32_t w, int32_t h)
{
    int32_t hSum = 0;
    if (_tcnc.PointCount == 0)
    {
        _tcnc.CurrentPoint = 0;
        _addLaserPoint(x, y, z, false);    //Move to start (laser off)

        while (hSum < h)
        {
            hSum += PATTERN2_STEP;
            _addLaserPoint(x + w, y + hSum, z, true);    //to the right (laser on)
            _addLaserPoint(x, y + hSum, z, true);    //to the left (laser on)
        }
        _addLaserPoint(x, y + hSum, z, false);    //turn off laser
    }
}

void RONINCNC_Tasks(void)
{
    int i;
    static int currAxis = 0;
    static uint32_t prevTime = 0;
    //New time difference after last call
    uint32_t timeDiff = _tcnc.MilisecondTimer
        + RONINCANC_MILISECOND_TIMER_MAX - prevTime + 1;
    prevTime = _tcnc.MilisecondTimer;
    switch (_tcnc.Status)
    {
        case RONINCANC_Init:
            if (SYS_TMR_Status(sysObj.sysTmr) == SYS_STATUS_READY && (UDPCOMM_GetState()
== UDPCOMM_READY))
            {
                if (_tcnc.TmrlmsHandle == DRV_HANDLE_INVALID)
                    _tcnc.TmrlmsHandle = SYS_TMR_CallbackPeriodic(1, 0,
&RONINCNC_Tmrlms_Callback);
                if (_tcnc.TmrlmsHandle == DRV_HANDLE_INVALID)
                    break;
                for (i = 0; i < RONINCANC_AXIS_COUNT; i++)
                {
                    IGUSD1_Initialize(&(_tcnc.AxisMotors[i]), IgusD1NodeIds[i],
AxisSettings[i], COPointers[i][0], COPointers[i][1], COPointers[i][2]);
                    IGUSD1_StartHoming(&(_tcnc.AxisMotors[i]));
                }
                RONIN_LASER_Initialize();
                RONIN_MILL_Initialize();
                _tcnc.Status = RONINCANC_SetupAxis;
            }
            break;
        case RONINCANC_SetupAxis:
            _tcnc.Status = RONINCANC_ServiceTasks;
            break;
        case RONINCANC_ServiceTasks:

            if (!_tcnc.Flags.LaserAxisMoveStart)
            {
                if (_tcnc.CurrentPoint < _tcnc.PointCount)
                {

```

```

        _setLaserPos(
            _tcnc.TotalPoints[_tcnc.CurrentPoint].X,
            _tcnc.TotalPoints[_tcnc.CurrentPoint].Y,
            _tcnc.TotalPoints[_tcnc.CurrentPoint].Z);
    if (_tcnc.TotalPoints[_tcnc.CurrentPoint].Flags.LaserOn)
        RONIN_LASER_Enable();
    else
        RONIN_LASER_Disable();
}
}

if (_tcnc.Flags.LaserAxisMoveStart)
{
    if (
        IGUSD1_IsTargetReached(&(_tcnc.AxisMotors[0])) &&
        IGUSD1_IsTargetReached(&(_tcnc.AxisMotors[1])) &&
        IGUSD1_IsTargetReached(&(_tcnc.AxisMotors[2])))
    {
        _tcnc.Flags.LaserAxisMoveStart = 0;
        _tcnc.CurrentPoint++;
        if (_tcnc.CurrentPoint == _tcnc.PointCount)
        {
            _tcnc.CurrentPoint = 0;
            _tcnc.PointCount = 0;
        }
    }
}
RONIN_COMMAND_PROC_ReplyPatternOk(RONIN_RequestPatternCompleteLaser);
}
}

if (_tcnc.Flags.MillAxisMoveStart)
{
    if (
        IGUSD1_IsTargetReached(&(_tcnc.AxisMotors[3])) &&
        IGUSD1_IsTargetReached(&(_tcnc.AxisMotors[4])) &&
        IGUSD1_IsTargetReached(&(_tcnc.AxisMotors[5])))
    {
        RONIN_COMMAND_PROC_ReplyPatternOk(RONIN_RequestPatternCompleteMill);
        _tcnc.Flags.MillAxisMoveStart = 0;
    }
}
for (i = 0; i < RONINCANC_AXIS_COUNT; i++)
{
    IGUSD1_Tasks(&_tcnc.AxisMotors[i], timeDiff);
}
RONIN_LASER_Tasks();
RONIN_MILL_Tasks();
break;
}
}
}

```



## Fails "Ronin\_CNC.h"

```
/*
 * File:   RONIN_CNC.h
 * Author: Vitalijs
 *
 * Created on tre?diena, 2020, 12 augusts, 08:45
 */

#ifndef RONIN_CNC_H
#define RONIN_CNC_H

#ifdef __cplusplus
extern "C" {
#endif

#include <stdint.h>
#include <stdbool.h>
#include "system_config.h"
#include "system_definitions.h"

void RONINCNC_Initialize(void);
void RONINCNC_Tasks(void);

void RONINCNC_SetLaserPos(int32_t x, int32_t y, int32_t z);
void RONINCNC_SetMillPos(int32_t x, int32_t y, int32_t z);
void RONINCNC_DrawLaserPattern1(int32_t x, int32_t y, int32_t z, int32_t w, int32_t h);
void RONINCNC_DrawLaserPattern2(int32_t x, int32_t y, int32_t z, int32_t w, int32_t h);

#ifdef __cplusplus
}
#endif

#endif /* RONIN_CNC_H */
```

## Fails "Ronin\_command\_proc.c"

```

#include "RONINCommandProc.h"
#include "RONIN_motor.h"
#include "RONIN_CNC.h"
#include "RONIN_laser.h"
#include "RONIN_mill.h"
#include <stdlib.h>

#define RONIN_COMMAND_WDT_PERIOD 200

#define RONIN_MOTOR_COUNT 4

#ifdef LEDRStateSet
#define WDT_TO_LED_ON() LEDRStateSet(0)
#define WDT_TO_LED_OFF() LEDRStateSet(1)
#endif
#ifdef BSP_LED_3On
#define WDT_TO_LED_ON() BSP_LED_3On()
#define WDT_TO_LED_OFF() BSP_LED_3Off()
#endif

#define MAX_TOKENS 32
const int8_t seps[] = ";";

typedef enum
{
    RONIN_INIT_READ = 0,
    RONIN_INIT_WRITE,
    RONIN_INIT_CHECK,
} RONIN_INITIT_PHASE;

typedef struct
{
    RONIN_STATE State;
    RONIN_INITIT_PHASE InitPhase;

    uint32_t CurrMotorID;
    int32_t SpeedL;
    int32_t SpeedR;
    uint32_t RPMtotal;
    uint64_t Etotal;
    uint64_t LaserTtotal;
    uint32_t LaserPower;
    uint32_t BumperStatus;
    uint32_t LaserStatus;
    uint32_t BattVolatge;
    uint32_t MotorCurrent[RONIN_MOTOR_COUNT];
    uint8_t DataBuff[32];
    SYS_TMR_HANDLE Tmr200msHandle;
    SYS_TMR_HANDLE Tmr1sHandle;
    SYS_TMR_HANDLE TmrWDTHandle;
    SIMPLE_QUEUE IncomeCommandQueue;
    SIMPLE_QUEUE ProcessedCommandQueue;
} RONIN_DATA;

int8_t *_RONIN_COMMAND_PROC_CommandStrings[] =
{
    RONIN_COMMAND_ASCII_STRINGS
};

```

```

#define RONIN_COMMAND_PROC_CMD_STR_COUNT      (sizeof(_RONIN_COMMAND_PROC_CommandStrings) /
\
                                                sizeof(_RONIN_COMMAND_PROC_CommandStrings[0]))

void RONIN_COMMAND_PROC_200ms_Callback( uintptr_t context, uint32_t currTick );
void RONIN_COMMAND_PROC_1s_Callback( uintptr_t context, uint32_t currTick );
void RONIN_COMMAND_PROC_WDT_Callback( uintptr_t context, uint32_t currTick );
void RONIN_SDO_ClientOpCallback( CANOPENAPP_SDO_ClientOpeResult_t opResult);

RONIN_DATA _t;

void RONIN_COMMAND_PROC_Initialize(void)
{
    memset(&_t, 0, sizeof(RONIN_DATA));
    SimpleQueue_Init(&_t.IncomeCommandQueue);
    SimpleQueue_Init(&_t.ProcessedCommandQueue);
    _t.Tmr200msHandle = SYS_TMR_CallbackPeriodic(200, 0,
&RONIN_COMMAND_PROC_200ms_Callback);
    _t.Tmr1sHandle = SYS_TMR_CallbackPeriodic(1000, 0, &RONIN_COMMAND_PROC_1s_Callback);
    _t.TmrWDTHandle = SYS_TMR_CallbackPeriodic(RONIN_COMMAND_WDT_PERIOD, 0,
&RONIN_COMMAND_PROC_WDT_Callback);
    _t.State = RONIN_INITIALIZE;
}

void RONIN_COMMAND_PROC_WDT_Timeout(void)
{
    WDT_TO_LED_ON();
    RONIN_MOTOR_SetSpeed(RONIN_MOTOR_FL, 0);
    RONIN_MOTOR_SetSpeed(RONIN_MOTOR_RL, 0);
    RONIN_MOTOR_SetSpeed(RONIN_MOTOR_FR, 0);
    RONIN_MOTOR_SetSpeed(RONIN_MOTOR_RR, 0);
}

void RONIN_COMMAND_PROC_WDT_Reload(void)
{
    SYS_TMR_ObjectReload(_t.TmrWDTHandle, RONIN_COMMAND_WDT_PERIOD,
0, &RONIN_COMMAND_PROC_WDT_Callback);
    WDT_TO_LED_OFF();
}

void RONIN_COMMAND_PROC_WDT_Callback( uintptr_t context, uint32_t currTick )
{
    RONIN_COMMAND_PROC_WDT_Timeout();
}

void RONIN_COMMAND_PROC_200ms_Callback( uintptr_t context, uint32_t currTick )
{
    RONIN_COMMAND *comPtr;
    // comPtr = (RONIN_COMMAND*)OSAL_Malloc(sizeof(RONIN_COMMAND));
    // if (comPtr)
    // {
    //     comPtr->Type = RONIN_RequestSpeed;
    //     comPtr->VarCount = 2;
    //     comPtr->Data[0] = _t.SpeedL;
    //     comPtr->Data[1] = _t.SpeedR;
    //     if (SimpleQueue_IsFull(&_t.IncomeCommandQueue))
    //         RONIN_COMMAND_PROC_DisposeCommandFromQueue(&_t.IncomeCommandQueue);
    //     SimpleQueue_Insert(&_t.IncomeCommandQueue, (int)comPtr);
    // }
}

void RONIN_COMMAND_PROC_1s_Callback( uintptr_t context, uint32_t currTick )
{

```

```

RONIN_COMMAND *comPtr;
int i;
if (_t.BattVolatge++ > 24) _t.BattVolatge = 0;
if (_t.BumperStatus++ > 1) _t.BumperStatus = 0;
if (_t.LaserStatus++ > 1) _t.LaserStatus = 0;
for (i = 0; i < 4; i++)
{
    _t.MotorCurrent[i]++;
    if (_t.MotorCurrent[i]++ > 100) _t.MotorCurrent[i] = 0;
}
// comPtr = (RONIN_COMMAND*)OSAL_Malloc(sizeof(RONIN_COMMAND));
// if (comPtr)
// {
//     comPtr->Type = RONIN_RequestSystemStatus;
//     comPtr->VarCount = 2;
//     comPtr->Data[0] = _t.BumperStatus;
//     comPtr->Data[1] = _t.LaserStatus;
//     if (SimpleQueue_IsFull(&_t.IncomeCommandQueue))
//         RONIN_COMMAND_PROC_DisposeCommandFromQueue(&_t.IncomeCommandQueue);
//     SimpleQueue_Insert(&_t.IncomeCommandQueue, (int)comPtr);
// }
// comPtr = (RONIN_COMMAND*)OSAL_Malloc(sizeof(RONIN_COMMAND));
// if (comPtr)
// {
//     comPtr->Type = RONIN_RequestBattVoltage;
//     comPtr->VarCount = 1;
//     comPtr->Data[0] = _t.BattVolatge;
//     if (SimpleQueue_IsFull(&_t.IncomeCommandQueue))
//         RONIN_COMMAND_PROC_DisposeCommandFromQueue(&_t.IncomeCommandQueue);
//     SimpleQueue_Insert(&_t.IncomeCommandQueue, (int)comPtr);
// }
// comPtr = (RONIN_COMMAND*)OSAL_Malloc(sizeof(RONIN_COMMAND));
// if (comPtr)
// {
//     comPtr->Type = RONIN_RequestMotorCurrents;
//     comPtr->VarCount = 4;
//     for (i = 0; i < 4; i++)
//     {
//         comPtr->Data[i] = _t.MotorCurrent[i];
//     }
//     if (SimpleQueue_IsFull(&_t.IncomeCommandQueue))
//         RONIN_COMMAND_PROC_DisposeCommandFromQueue(&_t.IncomeCommandQueue);
//     SimpleQueue_Insert(&_t.IncomeCommandQueue, (int)comPtr);
// }
}

```

```

void RONIN_SDO_ClientOpCallback( CANOPENAPP_SDO_ClientOpeResult_t opResult)
{
    if (opResult == CANOPENAPP_SDO_ClientOpeResultOK)
    {
        _t.CurrMotorID++;
        if (_t.CurrMotorID < 4)
        {
            _t.State = RONIN_INITIALIZE;
        }
        else
            _t.State = RONIN_TASKS;
    }
    else
    {
        _t.State = RONIN_ERROR;
    }
}

```

```

void RONIN_COMMAND_PROC_DisposeCommandFromQueue(SIMPLE_QUEUE *q)
{
    RONIN_COMMAND *comPtr;//TODO: pareizi izma?ana neapstr?d?tajiem?
    comPtr = (RONIN_COMMAND*)SimpleQueue_Remove(q);
    RONIN_COMMAND_PROC_DisposeCommand(comPtr);
}

void RONIN_COMMAND_PROC_ParseString(int8_t *str)
{
    size_t i = 0, tokenCount = 0;
    int8_t *tokens[MAX_TOKENS];
    int8_t *endPtr;
    int8_t radix;
    int32_t sp;
    RONIN_COMMAND *comPtr, *comPtr2;
    int32_t sign = 1;
    tokenCount = 0;
    tokens[0] = strtok(str, seps);
while(tokens[tokenCount] != NULL)
    {
        tokens[++tokenCount] = strtok(NULL, seps);
    }
if (tokenCount > 1)
    {
        for (i = 0; i < RONIN_COMMAND_PROC_CMD_STR_COUNT; i++)
        {
            if (!strcmp(tokens[1], _RONIN_COMMAND_PROC_CommandStrings[i])) //Check command
type                break;          //Stop checking, if valid command recognized
        }
    }
else
    i = RONIN_COMMAND_PROC_CMD_STR_COUNT;

//parse strings...
//If valid command found, handle command by inserting it into command queue
if (i < RONIN_COMMAND_PROC_CMD_STR_COUNT)
{
    comPtr = (RONIN_COMMAND*)OSAL_Malloc(sizeof(RONIN_COMMAND));
    if (comPtr)
    {
        errno = 0;
        comPtr->ID = strtol(tokens[0], (char**)&endPtr, 10);
        if (!errno && (tokens[0] != endPtr))
        {
            comPtr->Type = i + 1;
            for (i = 2; i < tokenCount; i++)
            {
                errno = 0;
                if (strchr(tokens[i], 'x'))
                    comPtr->Data[i - 2].Integer32 = strtol(tokens[i], (char**)&endPtr,
16);
                else if (strchr(tokens[i], '.'))
                    comPtr->Data[i - 2].Floating32 = strtod(tokens[i],
(char**)&endPtr);
                else
                    comPtr->Data[i - 2].Integer32 = strtol(tokens[i], (char**)&endPtr,
10);
                if (errno || (tokens[i] == endPtr))
                    break;
            }
            if (i == tokenCount)
            {

```

```

        if (SimpleQueue_IsFull(&t.IncomeCommandQueue))
RONIN_COMMAND_PROC_DisposeCommandFromQueue(&t.IncomeCommandQueue);
        comPtr->VarCount = tokenCount - 2;
        SimpleQueue_Insert(&t.IncomeCommandQueue, (int)comPtr);
    }
    else
    {
        RONIN_COMMAND_PROC_DisposeCommand(comPtr);
    }
}
else
{
    RONIN_COMMAND_PROC_DisposeCommand(comPtr);
}
}
//If no valid command found, possibly command form UDP joystick
//check if it is of type XXXXYYYZZZZZZQQQQ
//by checking length and terminating character
//XXXX = [1000...2000], center = 1500
else
{
    if (strlen(str) == 16)
    {
        comPtr = (RONIN_COMMAND*)OSAL_Malloc(sizeof(RONIN_COMMAND));
        if (comPtr)
        {
            comPtr->ID = 0;
            comPtr->Type = RONIN_SetSpeed;
            //values ----YYYY----QQQQ- are used for left and right speeds
            str[8] = 0;
            str[16] = 0;
            comPtr->Data[0].Integer32 = strtol(str + 4, NULL, 10);
            if (comPtr->Data[0].Integer32 < 1600 && comPtr->Data[0].Integer32 > 1400)
                comPtr->Data[0].Integer32 = 1500;
            comPtr->Data[0].Integer32 = (comPtr->Data[0].Integer32 - 1500);
            if (comPtr->Data[0].Integer32 < 0)
                sign = -1;
            else
                sign = 1;
            comPtr->Data[0].Integer32 /= -5;    //[-100...100]
            comPtr->Data[0].Integer32 = comPtr->Data[0].Integer32 * comPtr-
>Data[0].Integer32;
            comPtr->Data[0].Integer32 /= -100 * sign;

            comPtr->Data[1].Integer32 = strtol(str + 12, NULL, 10);
            if (comPtr->Data[1].Integer32 < 1600 && comPtr->Data[1].Integer32 > 1400)
                comPtr->Data[1].Integer32 = 1500;
            comPtr->Data[1].Integer32 = (comPtr->Data[1].Integer32 - 1500);
            if (comPtr->Data[1].Integer32 < 0)
                sign = -1;
            else
                sign = 1;
            comPtr->Data[1].Integer32 /= -5;    //[-100...100]

            comPtr->Data[1].Integer32 = comPtr->Data[1].Integer32 * comPtr-
>Data[1].Integer32;
            comPtr->Data[1].Integer32 /= -100 * sign;

            comPtr->VarCount = 2;
            SimpleQueue_Insert(&t.IncomeCommandQueue, (int)comPtr);
        }
    }
}

```



```

        comPtr->Data[1].Unsigned32,
        comPtr->Data[2].Unsigned32,
        comPtr->Data[3].Unsigned32,
        comPtr->Data[4].Unsigned32,
        comPtr->Data[5].Unsigned32);
    }
    break;
case 2:
    RONINCNC_DrawLaserPattern2(
        comPtr->Data[1].Unsigned32,
        comPtr->Data[2].Unsigned32,
        comPtr->Data[3].Unsigned32,
        comPtr->Data[4].Unsigned32,
        comPtr->Data[5].Unsigned32);
    }
    break;
default:
    break;
}
break;
case RONIN_RequestRevStats:
    break;
case RONIN_RequestEnergyStats:
    break;
case RONIN_RequestLaserStats:
    break;
case RONIN_SetLaserPow:
    _t.LaserPower = comPtr->Data[0].Unsigned32;
    RONIN_LASER_SetPower((uint16_t) comPtr->Data[0].Unsigned32);
    break;
case RONIN_EnableLaser:
    if (comPtr->Data[0].Unsigned32)
        RONIN_LASER_Enable();
    else
        RONIN_LASER_Disable();
    break;
case RONIN_EnableMill:
    if (comPtr->Data[0].Unsigned32)
        RONIN_MILL_Enable(comPtr->Data[0].Unsigned32);
    else
        RONIN_MILL_Disable();
    break;
case RONIN_RequestLaserPow:
    break;
case RONIN_RequestSpeed:
    break;
case RONIN_RequestSystemStatus:
    break;
case RONIN_RequestBattVoltage:
    break;
case RONIN_RequestMotorCurrents:
    break;
case RONIN_SetMotorControlMode:
    if (comPtr->Data[0].Unsigned32 == 0)
    {
        //feedback mode
        RONIN_MOTOR_SetControlMode(0);
    }
    else
    {
        //raw PWM mode
        RONIN_MOTOR_SetControlMode(1);
    }
    break;
case RONIN_CNC_Ax1:
    //IGUSD1_SetPos(comPtr->Data[0]);

```



```

        RONINCNC_SetLaserPos(comPtr->Data[0].Integer32, comPtr-
>Data[1].Integer32, 0);
        break;
    case RONIN_ReadCANObj:
        coObject.Idx = (int16_t)comPtr->Data[1].Unsigned32;
        coObject.SubIdx = (int16_t)comPtr->Data[2].Unsigned32;
        coObject.Value = 0; //Cler data buffer, because byte array of
shorter than 4 bytes can be read
        CANOPENAPP_TRANSFER_Initiate( (uint8_t)comPtr-
>Data[0].Unsigned32,
                                     &coObject,
                                     1,
                                     CANOPENAPP_SDO_TRANSFER_UPLOAD,
false);
        break;
    case RONIN_WriteCANObj:
        //Transfer one object to given CANopen node
        coObject.Idx = (int16_t)comPtr->Data[1].Unsigned32;
        coObject.SubIdx = (int16_t)comPtr->Data[2].Unsigned32;
        coObject.Value = comPtr->Data[3].Unsigned32;
        coObject.Size = (int16_t)comPtr->Data[4].Unsigned32;
        CANOPENAPP_TRANSFER_Initiate( (uint8_t)comPtr-
>Data[0].Integer32,
                                     &coObject,
                                     1,
                                     CANOPENAPP_SDO_TRANSFER_DOWNLOAD,
false);

        break;
    case RONIN_StartHoming:
        IGUSD1_StartHoming(NULL);

        break;
    }
    if (SimpleQueue_IsFull(&t.ProcessedCommandQueue))
        RONIN_COMMAND_PROC_DisposeCommandFromQueue(&t.ProcessedCommandQueue);
    SimpleQueue_Insert(&t.ProcessedCommandQueue, (int)comPtr);
}
break;
case RONIN_ERROR:
    _t.State = RONIN_INITIALIZE;
    break;
}

}

void RONIN_COMMAND_PROC_ReplyPatternOk(RONIN_COMMAND_TYPE commandType)
{
    RONIN_COMMAND *comPtr = (RONIN_COMMAND*)OSAL_Malloc(sizeof(RONIN_COMMAND));
    comPtr->ID = 0;
    comPtr->VarCount = 1;
    comPtr->Type = commandType;
    comPtr->InterfaceType = RONIN_COMM_INTERFACE_FAST; //FIXME: ignored in
RONIN_COMMAND_PROC_GetStringToSend
    comPtr->Data[0].Integer32 = 0;

    if (SimpleQueue_IsFull(&t.ProcessedCommandQueue))
        RONIN_COMMAND_PROC_DisposeCommandFromQueue(&t.ProcessedCommandQueue);
    SimpleQueue_Insert(&t.ProcessedCommandQueue, (int)comPtr);
}

```

```

void RONIN_COMMAND_PROC_ReplyHomingOk (void)
{
    RONIN_COMMAND *comPtr = (RONIN_COMMAND*)OSAL_Malloc(sizeof(RONIN_COMMAND));
    comPtr->ID = 0;
    comPtr->VarCount = 1;
    comPtr->Type = RONIN_RequestHomingComplete;
    comPtr->InterfaceType = RONIN_COMM_INTERFACE_FAST;          //FIXME: ignored in
RONIN_COMMAND_PROC_GetStringToSend
    comPtr->Data[0].Integer32 = 0;

    if (SimpleQueue_IsFull(&_t.ProcessedCommandQueue))
        RONIN_COMMAND_PROC_DisposeCommandFromQueue (&_t.ProcessedCommandQueue);
    SimpleQueue_Insert (&_t.ProcessedCommandQueue, (int)comPtr);
}

/* POTENCI?LA K??DA!
 *
 *
 * RONIN_COMMAND_PROC_DisposeCommand tiek izsaukts tikai seit. Ja neviens negribes
atbildes stringu, komanda nedispososies!
 *
 *
 */
RONIN_COMM_INTERFACE_TYPE RONIN_COMMAND_PROC_GetStringToSend(int8_t *s)
{
    RONIN_COMMAND *comPtr = NULL;
    RONIN_COMM_INTERFACE_TYPE ret = RONIN_COMM_NO_DATA;
    if (!SimpleQueue_IsEmpty(&_t.ProcessedCommandQueue))
    {
        comPtr = (RONIN_COMMAND*)SimpleQueue_Remove (&_t.ProcessedCommandQueue);
        switch (comPtr->Type)
        {
            case RONIN_SetSpeed:
                break;
            case RONIN_SetLaserCoord:
                //sprintf(s, "WF");
                //ret = RONIN_COMM_INTERFACE_FAST;
                break;
            case RONIN_RequestRevStats:
                sprintf(s, "R;%d", _t.RPMtotal);
                ret = RONIN_COMM_INTERFACE_FAST;
                break;
            case RONIN_RequestEnergyStats:
                sprintf(s, "E;%lu", _t.Etotal);
                ret = RONIN_COMM_INTERFACE_FAST;
                break;
            case RONIN_RequestLaserStats:
                sprintf(s, "L;%lu", _t.LaserTtotal);
                ret = RONIN_COMM_INTERFACE_FAST;
                break;
            case RONIN_RequestPatternComplete:
                sprintf(s, "patternOk");
                ret = RONIN_COMM_INTERFACE_FAST;
                break;
            case RONIN_RequestPatternCompleteLaser:
                sprintf(s, "patternOkLaser");
                ret = RONIN_COMM_INTERFACE_FAST;
                break;
            case RONIN_RequestPatternCompleteMill:
                sprintf(s, "patternOkMill");
                ret = RONIN_COMM_INTERFACE_FAST;
                break;
        }
    }
}

```

```

    case RONIN_RequestHomingComplete:
        sprintf(s, "homingOk");
        ret = RONIN_COMM_INTERFACE_FAST;
        break;
    case RONIN_SetLaserPow:
        sprintf(s, "LP;%d", _t.LaserPower);
        ret = RONIN_COMM_INTERFACE_FAST;
        break;
    case RONIN_EnableMill:
        sprintf(s, "EM");
        ret = RONIN_COMM_INTERFACE_FAST;
        break;
    case RONIN_RequestLaserPow:
        sprintf(s, "LP;%d", _t.LaserPower);
        ret = RONIN_COMM_INTERFACE_FAST;
        break;
    case RONIN_RequestSpeed:
        sprintf(s, "S;%d;%d", comPtr->Data[0], comPtr->Data[1]); //TODO: var
p?rtais?t uz pa?reiz?jiem (caur_t.(...))
        ret = RONIN_COMM_INTERFACE_FAST;
        break;
    case RONIN_RequestSystemStatus:
        sprintf(s, "SS;%d;%d", comPtr->Data[0], comPtr->Data[1]);
        ret = RONIN_COMM_INTERFACE_FAST;
        break;
    case RONIN_RequestBattVoltage:
        sprintf(s, "V;%d", comPtr->Data[0]);
        ret = RONIN_COMM_INTERFACE_FAST;
        break;
    case RONIN_RequestMotorCurrents:
        sprintf(s, "I;%d;%d;%d;%d",
                comPtr->Data[0], comPtr->Data[1],
                comPtr->Data[2], comPtr->Data[3]);
        ret = RONIN_COMM_INTERFACE_FAST;
        break;
    case RONIN_SetMotorControlMode:

        break;
}
RONIN_COMMAND_PROC_DisposeCommand(comPtr);
}
return ret;
}

```

## Fails "RoninCommandProc.h"

```

/*
 * File:   RONINCommandProc.h
 * Author: Vitalijs
 *
 * Created on otrdiena, 2019, 11 j?nijs, 17:01
 */

#ifndef RONINCOMMANDPROC_H
#define RONINCOMMANDPROC_H

#ifdef __cplusplus
extern "C" {
#endif

#include <stdint.h>
#include <string.h>
#include <stddef.h>
#include <errno.h>
#include "simple_queue.h"
#include "system_definitions.h"
#include "system_config.h"
#include "osal/osal.h"
#include "CANopenApp.h"

//Max command data size in words
#define RONIN_COMMAND_DATA_VAR_COUNT_MAX 8

/*
 * General command structure
 * <command id>;<command ascii name>;[parameter1;parameter2...]
 *
 * -----RONIN_SetSpeed-----
 * <command id>;SSP;<left speed>;<right speed>
 * 1;SSP;123;123          set left and right speed in RPM [-250...250]
 *                          negative values - backward direction
 *
 * -----RONIN_SetLaserCoord-----
 * <command id>;SLC;<x>;<y>;<z>
 * 1;SLC;100;100;100      set laser CNC coordinates in mm [0...axis Max]
 *
 * reply: "patternOkLaser"
 *
 * -----RONIN_SetMillCoord-----
 * <command id>;SMC;<x>;<y>;<z>
 * 1;SMC;100;100;100      set mill CNC coordinates in mm [0...axis Max]
 *
 * reply: "patternOkMill"
 *
 * -----RONIN_SetLaserPow-----
 * <command id>;SL;<power>
 * 1;SL;50                set laser power in % [0...100]
 *
 * -----RONIN_EnableLaser-----
 * <command id>;EL;<state>
 * 1;EL;1                  enable laser (if 1) [0;1]
 *
 * -----RONIN_EnableMill-----
 * <command id>;EM;<duration>
 * 1;EM;100                enable mill for duration (ms), resolution 100 ms
 *                          if 0, mill is disabled
 */

```

```

*
* -----RONIN_DrawPattern-----
* <command id>;DPA;<type>;<x>;<y>;<z>;<w>;<h>
* 1;DPA;1;100;100;100;20;20
*
* Draw pattern type with laser ON at point x, y, with size w, h and at constant height z
* x,y -> x+w,y
* x+w -> x+w,y+h
* x+w,y+h -> x,y+h
* x,y+h -> x,y
* type:
*   - 1 - rectangle
*   - 2 - horizontal lines (along x axis) with step 3 mm
*
* reply: "patternOkLaser"
*
* -----RONIN_SetMotorControlMode-----
* <command id>;SMM;<modeType>
* 1;SMM;0
* modeType:
*   - 0 - feedback with current and speed sensors
*   - 1 - raw PWM
*/

```

```

typedef enum
{
    RONIN_SetSpeed = 1,
    RONIN_SetLaserCoord,
    RONIN_SetMillCoord,
    RONIN_DrawPattern,
    RONIN_RequestRevStats,
    RONIN_RequestEnergyStats,
    RONIN_RequestLaserStats,
    RONIN_RequestPatternComplete,
    RONIN_RequestHomingComplete,
    RONIN_SetLaserPow,
    RONIN_EnableLaser,
    RONIN_EnableMill,
    RONIN_RequestLaserPow,
    RONIN_RequestSpeed,
    RONIN_RequestSystemStatus,
    RONIN_RequestBattVoltage,
    RONIN_RequestMotorCurrents,
    RONIN_SetMotorControlMode,
    RONIN_CNC_Ax1,
    RONIN_ReadCANObj,
    RONIN_WriteCANObj,
    RONIN_StartHoming,
    RONIN_RequestPatternCompleteLaser,
    RONIN_RequestPatternCompleteMill
} RONIN_COMMAND_TYPE;

```

```

#define RONIN_COMMAND_ASCII_STRINGS
    "SSP", \
    "SLC", \
    "SMC", \
    "DPA", \
    "RRS", \
    "RES", \
    "RLS", \
    "RPC", \
    "RHC", \
    "SL", \
    "EL", \
    "EM", \

```

```
"LPR", \
"RSP", \
"RST", \
"RBV", \
"RMC", \
"SMM", \
"Ax1", \
"CO", \
"WCO", \
"HO"
```

```
typedef enum
{
    RONIN_COMM_NO_DATA = 0,
    RONIN_COMM_INTERFACE_FAST = 1,
    RONIN_COMM_INTERFACE_RELIABLE = 2
} RONIN_COMM_INTERFACE_TYPE;
```

```
typedef enum
{
    RONIN_INITIALIZE = 0,
    RONIN_INIT_BUSY,
    RONIN_TASKS,
    RONIN_ERROR,
} RONIN_STATE;
```

```
typedef union
{
    int32_t Integer32;
    uint32_t Unsigned32;
    float Floating32;
} genericData32_t;
```

```
typedef struct
{
    int32_t ID;
    RONIN_COMMAND_TYPE Type;
    genericData32_t Data[RONIN_COMMAND_DATA_VAR_COUNT_MAX];
    ///Token count
    int32_t VarCount;
    RONIN_COMM_INTERFACE_TYPE InterfaceType;
} _RONIN_COMMAND;
```

```
typedef _RONIN_COMMAND RONIN_COMMAND;
```

```
void RONIN_COMMAND_PROC_Initialize(void);
void RONIN_COMMAND_PROC_ParseString(int8_t *str);
bool RONIN_COMMAND_PROC_GetCommand(RONIN_COMMAND *ptr);
void RONIN_COMMAND_PROC_DisposeCommand(RONIN_COMMAND *comPtr);
void RONIN_COMMAND_PROC_WDT_Timeout(void);
```

```
void RONIN_COMMAND_PROC_ReplyPatternOk(RONIN_COMMAND_TYPE commandType);
void RONIN_COMMAND_PROC_Tasks(void);
RONIN_COMM_INTERFACE_TYPE RONIN_COMMAND_PROC_GetStringToSend(int8_t *s);
void RONIN_COMMAND_PROC_DisposeCommandFromQueue(SIMPLE_QUEUE *q);
```

```
#ifdef    __cplusplus
}
#endif
```

```
#endif    /* RONINCOMMANDPROC_H */
```



## Fails "Ronin\_laser.c"

```

#include "RONIN_laser.h"

typedef enum
{
    LASER_STATE_UNINITIALIZED,
    LASER_STATE_OPERATING
} RONIN_LASER_STATE;

typedef struct
{
    RONIN_LASER_STATE State;
    DRV_HANDLE TimerHadle;
    DRV_HANDLE OCHadle;
    int16_t Power;
    struct
    {
        int32_t LaserEnabled:1;
        int32_t LaserEnabledSoft:1;
    } Flags;
} RONIN_LASER_DATA;

RONIN_LASER_DATA _tl;

void RONIN_LASER_Initialize(void)
{
    _tl.State = LASER_STATE_UNINITIALIZED;
    _tl.TimerHadle = DRV_HANDLE_INVALID;
    _tl.OCHadle = DRV_HANDLE_INVALID;
    _tl.Flags.LaserEnabled = 0;
    _tl.Flags.LaserEnabledSoft = 0;
}

void RONIN_LASER_AddPointPair(RONIN_LASER_POINTPAIR *pp)
{
}

void RONIN_LASER_AddPointPairRange(RONIN_LASER_POINTPAIR *pp, uint32_t count)
{
}

void RONIN_LASER_StartSequence(void)
{
}

bool RONIN_LASER_IsSequenceComplete(void)
{
    return true;
}

/*
 * Power - 0...100%
 */
void RONIN_LASER_SetPower(uint16_t power)
{
    if (power <= 100)
        _tl.Power = power * LAER_POWER_MULT;
}

void RONIN_LASER_Enable(void)

```



```

{
    _tl.Flags.LaserEnabled = 1;
}

void RONIN_LASER_Disable(void)
{
    _tl.Flags.LaserEnabled = 0;
}

void RONIN_LASER_Tasks(void)
{
    switch(_tl.State)
    {
        case LASER_STATE_UNINITIALIZED:
            //Opening timer driver
            if (_tl.TimerHadle == DRV_HANDLE_INVALID)
                _tl.TimerHadle = DRV_TMR_Open(DRV_TMR_INDEX_1,
                    DRV_IO_INTENT_READWRITE | DRV_IO_INTENT_EXCLUSIVE);
            if (_tl.TimerHadle == DRV_HANDLE_INVALID)
                break;
            //Opening OC driver
            if (_tl.OCHadle == DRV_HANDLE_INVALID)
                _tl.OCHadle = DRV_OC_Open(DRV_OC_INDEX_0,
                    DRV_IO_INTENT_READWRITE | DRV_IO_INTENT_EXCLUSIVE);
            if (_tl.OCHadle == DRV_HANDLE_INVALID)
                break;

            DRV_TMR_AlarmRegister(_tl.TimerHadle, LASER_PWM_PERIOD, true, 0, NULL);
            DRV_TMR_Start(_tl.TimerHadle);
            DRV_OC_PulseWidthSet(_tl.OCHadle, LASER_ZERO_POWER);
            DRV_OC_Start(DRV_OC_INDEX_0,
                DRV_IO_INTENT_READWRITE | DRV_IO_INTENT_EXCLUSIVE);
            _tl.State = LASER_STATE_OPERATING;
            break;
        case LASER_STATE_OPERATING:
            if (_tl.Flags.LaserEnabled || _tl.Flags.LaserEnabledSoft)
            {
                DRV_OC_PulseWidthSet(_tl.OCHadle, _tl.Power);
                BSP_LEDOn(BSP_LED_1);
                BSP_LEDOn(BSP_LED_2);
                BSP_LEDOn(BSP_LED_3);
            }
            else
            {
                DRV_OC_PulseWidthSet(_tl.OCHadle, LASER_ZERO_POWER);
                BSP_LEDOff(BSP_LED_1);
                BSP_LEDOff(BSP_LED_2);
                BSP_LEDOff(BSP_LED_3);
            }
            break;
    }
}

```

## Fails "Ronin\_laser.h"

```
/*
 * File:   RONIN_laser.h
 * Author: Vitalijs
 *
 * Created on tre?diena, 2019, 12 j?nijs, 13:38
 */

#ifndef RONIN_LASER_H
#define RONIN_LASER_H

#ifdef __cplusplus
extern "C" {
#endif

#include <stdint.h>
#include <stdbool.h>
#include "system_config.h"
#include "system_definitions.h"

#define LASER_PWM_PERIOD    200
#define LAER_POWER_MULT 2
#define LASER_ZERO_POWER 0
#define LASER_MAX_POWER LAER_POWER_MULT * 100

typedef struct
{
    int32_t X;
    int32_t Y;
} RONIN_LASER_POINTPAIR;

void RONIN_LASER_Initialize(void);
void RONIN_LASER_AddPointPair(RONIN_LASER_POINTPAIR *pp);
void RONIN_LASER_AddPointPairRange(RONIN_LASER_POINTPAIR *pp, uint32_t count);
void RONIN_LASER_StartSequence(void);
bool RONIN_LASER_IsSequenceComplete(void);
void RONIN_LASER_Tasks(void);

void RONIN_LASER_SetPower(uint16_t power);
void RONIN_LASER_Enable(void);
void RONIN_LASER_Disable(void);

#ifdef __cplusplus
}
#endif

#endif /* RONIN_LASER_H */
```

## Fails "Ronin\_mill.c"

```

#include "RONIN_mill.h"

typedef enum
{
    MILL_STATE_UNINITIALIZED,
    MILL_STATE_OPERATING
} RONIN_MILL_STATE;

typedef struct
{
    RONIN_MILL_STATE State;
    SYS_TMR_HANDLE Tmr100msHandle;
    uint32_t TimeCounter;
    int16_t Power;
    struct
    {
        int32_t InitiateMill:1;
        int32_t MillEnabled:1;
        int32_t MillEnabledSoft:1;
    } Flags;
} RONIN_MILL_DATA;

RONIN_MILL_DATA _tm;

void RONINMill_Tmr100ms_Callback ( uintptr_t context, uint32_t currTick );

void RONINMill_Tmr100ms_Callback ( uintptr_t context, uint32_t currTick )
{
    if (_tm.Flags.InitiateMill)
    {
        _tm.Flags.InitiateMill = 0;
        _tm.Flags.MillEnabled = 1;
    }
    if (_tm.TimeCounter)
        _tm.TimeCounter--;
    else
        _tm.Flags.MillEnabled = 0;
}

void RONIN_MILL_Initialize(void)
{
    _tm.State = MILL_STATE_UNINITIALIZED;
    _tm.Flags.MillEnabled = 0;
    _tm.Flags.MillEnabledSoft = 0;
    _tm.Tmr100msHandle = DRV_HANDLE_INVALID;
    _tm.TimeCounter = 0;
}

void RONIN_MILL_Enable(uint32_t ms)
{
    _tm.Flags.InitiateMill = 1;
    _tm.TimeCounter = ms / 100;
}

void RONIN_MILL_Disable(void)
{
    _tm.Flags.MillEnabled = 0;
}

void RONIN_MILL_Tasks(void)
{

```

```
switch(_tm.State)
{
    case MILL_STATE_UNINITIALIZED:
        if (SYS_TMR_Status(sysObj.sysTmr) == SYS_STATUS_READY)
        {
            if (_tm.Tmr100msHandle == DRV_HANDLE_INVALID)
                _tm.Tmr100msHandle = SYS_TMR_CallbackPeriodic(100, 0,
&RONINMill_Tmr100ms_Callback);
            if (_tm.Tmr100msHandle != DRV_HANDLE_INVALID)
                _tm.State = MILL_STATE_OPERATING;
        }
        break;
    case MILL_STATE_OPERATING:
        if (_tm.Flags.MillEnabled || _tm.Flags.MillEnabledSoft)
        {
            MillControlOn();
        }
        else
        {
            MillControlOff();
        }
        break;
}
}
```

## Fails "Ronin\_mill.h"

```
/*
 * File:   RONIN_laser.h
 * Author: Vitalijs
 *
 * Created on tre?diena, 2019, 12 j?nijs, 13:38
 */

#ifndef RONIN_MILL_H
#define RONIN_MILL_H

#ifdef __cplusplus
extern "C" {
#endif

#include <stdint.h>
#include <stdbool.h>
#include "system_config.h"
#include "system_definitions.h"

void RONIN_MILL_Initialize(void);

void RONIN_MILL_Tasks(void);

void RONIN_MILL_Enable(uint32_t ms);
void RONIN_MILL_Disable(void);

#ifdef __cplusplus
}
#endif

#endif /* RONIN_MILL_H */
```

## Fails "Ronin\_motor.c"

```

#include "RONIN_motor.h"

static RONIN_MOTOR_DATA _tmd;

void MotorTimer_Callback ( uintptr_t context, uint32_t currTick )
{
}

void MotorTimer100_Callback ( uintptr_t context, uint32_t currTick )
{
    //SYS_CONSOLE_PRINT("Motor actual velocity: %d \r\n", CO_OD_RAM.motorActualVelocity1);
}

void RONIN_MOTOR_Initialize(void)
{
    memset(&_tmd, 0, sizeof (_tmd));
    _tmd.MotorControlMode = 0;
    _tmd.State = RONIN_MOTOR_STATE_INIT;
}

void RONIN_MOTOR_Tasks(void)
{
    switch (_tmd.State)
    {
        case RONIN_MOTOR_STATE_INIT:
            if (SYS_TMR_Status(sysObj.sysTmr) == SYS_STATUS_READY)
            {
                _tmd.Tmr10msHandle = SYS_TMR_CallbackPeriodic(10, 0,
&MotorTimer_Callback);
                _tmd.Tmr100msHandle = SYS_TMR_CallbackPeriodic(100, 0,
&MotorTimer100_Callback);
            }
            else
                break;
            _tmd.State = RONIN_MOTOR_STATE_TASKS;
            break;
        case RONIN_MOTOR_STATE_TASKS:
            break;
    }
}

void RONIN_MOTOR_SetSpeed(RONIN_MOTOR_ID motor, int16_t speed)
{
    int32_t spd = speed;
    //if PWM mode, scale up to PWM max
    // if (_tmd.MotorControlMode == 1)
    // {
    //     speed *= 32;          //250 * 32 = 8000;
    // }
    switch (motor)
    {
        case RONIN_MOTOR_FL:
            CO_OD_RAM.motorTargetVelocity1 = speed;
            break;
        case RONIN_MOTOR_RL:
            CO_OD_RAM.motorTargetVelocity2 = speed;
            break;
        case RONIN_MOTOR_RR:
            CO_OD_RAM.motorTargetVelocity3 = speed;
            break;
    }
}

```

```
        case RONIN_MOTOR_FR:
            CO_OD_RAM.motorTargetVelocity4 = speed;
            break;
    }

}

void RONIN_MOTOR_SetControlMode(int16_t mode)
{
    _tmd.MotorControlMode = mode;
    if (mode)
    {
        CO_OD_RAM.motorControlword1 |= 4;
        CO_OD_RAM.motorControlword2 |= 4;
        CO_OD_RAM.motorControlword3 |= 4;
        CO_OD_RAM.motorControlword4 |= 4;
    }
    else
    {
        CO_OD_RAM.motorControlword1 &= ~4;
        CO_OD_RAM.motorControlword2 &= ~4;
        CO_OD_RAM.motorControlword3 &= ~4;
        CO_OD_RAM.motorControlword4 &= ~4;
    }
}
```

## Fails "Ronin\_motor.h"

```
/*
 * File:   RONIN_CNC.h
 * Author: Vitalijs
 *
 * Created on tre?diena, 2020, 12 augusts, 08:45
 */

#ifndef RONIN_CNC_H
#define RONIN_CNC_H

#ifdef __cplusplus
extern "C" {
#endif

#include <stdint.h>
#include <stdbool.h>
#include "system_config.h"
#include "system_definitions.h"

void RONINCNC_Initialize(void);
void RONINCNC_Tasks(void);

void RONINCNC_SetLaserPos(int32_t x, int32_t y, int32_t z);
void RONINCNC_SetMillPos(int32_t x, int32_t y, int32_t z);
void RONINCNC_DrawLaserPattern1(int32_t x, int32_t y, int32_t z, int32_t w, int32_t h);
void RONINCNC_DrawLaserPattern2(int32_t x, int32_t y, int32_t z, int32_t w, int32_t h);

#ifdef __cplusplus
}
#endif

#endif /* RONIN_CNC_H */
```



**Fails "simple\_queue\_config.h"**

```
/*
 * File:   queue_config.h
 * Author: Vitalijs
 *
 * Created on otrdiena, 2019, 4 j?nijs, 16:40
 */

#ifndef SIMPLE_QUEUE_CONFIG_H
#define SIMPLE_QUEUE_CONFIG_H

#ifdef __cplusplus
extern "C" {
#endif

#define SIMPLE_QUEUE_MAX 64

#ifdef __cplusplus
}
#endif

#endif /* QUEUE_CONFIG_H */
```