ModusToolbox 2.3 XMC4700 Relax Kitによる IEEE1588 PTP プロジェクト作成手順

このドキュメントについて

- このドキュメントでは、IEEE1588規格概要及び統合開発環境ModusToolboxを使用して XMC™ 4700 Relax KitでIEEE1588プロジェクトを作成する手順について説明するものです。

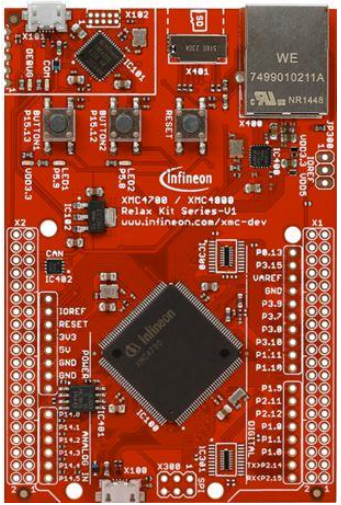
- このドキュメントではPCにModusToolbox 2.3がインストールされている事を前提としています。

Copyright © Infineon Technologies AG 2022. All rights reserved.

使用するハードウェア(Kit)について(1/2)

› このドキュメントで作成するプロジェクトを動作させるターゲットハードウェアとして KIT_XMC47_RELAX_V1を使用する事を前提としています。

› 他のKitを使用する場合、プロジェクト生成時のKit又はマイコンの型格を合わせて設定してください。



– KIT_XMC47_RELAX_V1

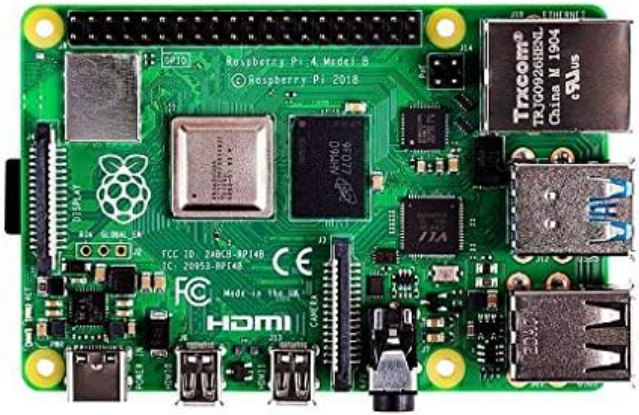
– https://www.infineon.com/cms/jp/product/evaluation-boards/kit_xmc47_relax_v1/

Copyright © Infineon Technologies AG 2022. All rights reserved.

使用するハードウェア(Kit)について(1/2)

› PTP Master側のデバイスとしてRaspberry PI 4を使用しています。
Note:有線LANが接続可能なモデルであれば他のモデルでも代替可能です。

› また、KIT_XMC47_RELAX_V1 と接続するためのLANケーブルが必要です。
Note:クロスケーブルではなく、ストレートケーブルで接続可能です。



- Raspberry PI 4
- https://www.raspberrypi.org/

Copyright © Infineon Technologies AG 2022. All rights reserved.

IEEE1588規格概要

Copyright © Infineon Technologies AG 2022. All rights reserved.

IEEE1588規格とは

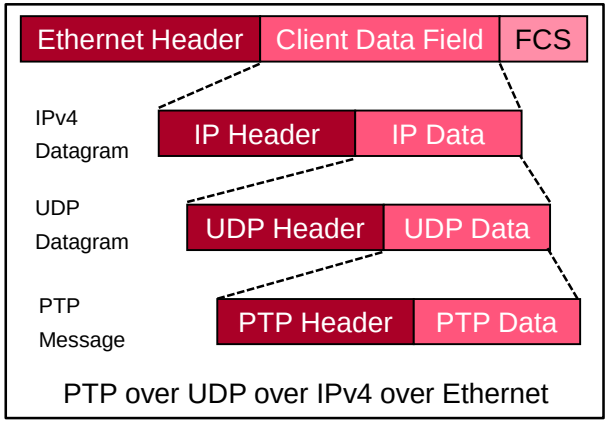
- イーサーネット経由で精度の高い時刻同期を行うためのプロトコル(PTP: Precision Time Protocol)です。
- 2002年にVersion1が、2008年にVersion2(IEEE1588v2)が策定されました。
- NTP (Network Time Protocol) やSNTP（Simple Network Time Protocol）がミリ秒オーダーの精度であるのに対し、PTPはマイクロ秒オーダーの精度であり、ネットワーク構成を限定すればナノ秒オーダーまで精度の向上が可能です。

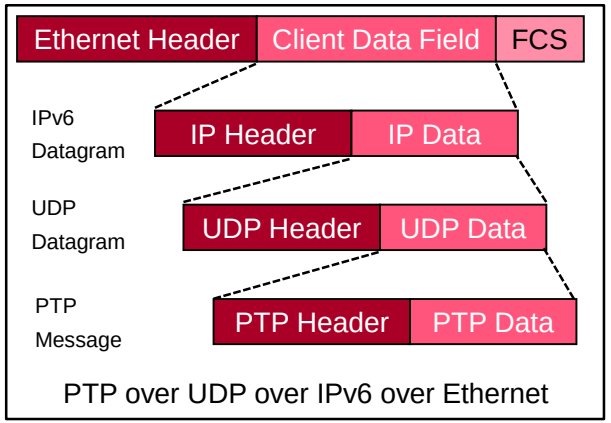
- 採用事例
  - 音響・映像システム
  - 携帯基地局間同期
  - ロボット制御等

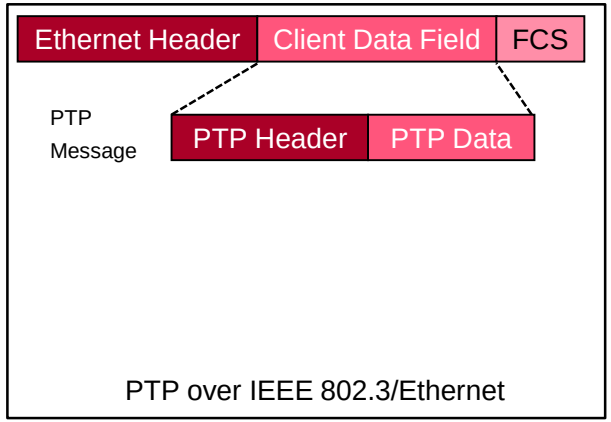
Copyright © Infineon Technologies AG 2022. All rights reserved.

PTP時刻同期概要とネットワークプロトコル

- PTPではメッセージを送受信することで時刻の同期を行います。
- Master(Grand Master Clock)は時刻の配信を行い、SlaveはMasterとの時刻差(Offset From Master)、遅延時間(Mean Path Delay) を計算してMasterと同期します。
- PTPメッセージは複数のネットワークプロトコルを介して転送されます。
  - PTP over UDP over IPv4 over Ethernet
  - PTP over UDP over IPv6 over Ethernet
  - PTP over IEEE 802.3/Ethernet







Copyright © Infineon Technologies AG 2022. All rights reserved.

PTP Header(1/2)

› PTPメッセージのHeaderは34byteで構成されています。

PTP Message Header									
Bits								Octets	Offset
7	6	5	4	3	2	1	0		
transportSpecific				messageType				1	0
Reserved				versionPTP				1	1
messageLength								2	2
domainNumber								1	4
Reserved								1	5
flags								2	6
connectionField								8	8
Reserved								4	16
sourcePortIdentify								10	20
sequenceID								2	30
controlField								1	32
logMessageInterval								1	33

› **transportSpecific**
Ethertypeのサブタイプと定義されていますが、0設定で問題ありません。

› **messageType**
PTPメッセージタイプ(後述)を設定します。

› **Reserved**
全てのReservedフィールドは0設定となります。

› **versionPTP**
対応するPTPのVersionに応じて以下のように設定します。
PTPv1 : 0x01
PTPv2 : 0x02

› **messageLength**
PTPメッセージ(PTP Header + PTP data)
のサイズをbyteで設定します。

Copyright © Infineon Technologies AG 2022. All rights reserved.

PTP Header(2/2)

- **domainNumber**
  複数ドメインでクロック同期を行う際にdomain番号を設定します。単一クロック同期を行う場合0設定となります。

- **flags**
  Master側がSyncメッセージ送信時にマルチキャスト、2Step等の設定を行います(Slave側は0設定となります)
  octet0 bit1 = twoStepFlag　　0:1Step　1:2Step
  octet0 bit2 = unicastFlag　　0:multicast 1:unicast

- **connectionField**
  ナノ秒単位で計測した補正値を2^16倍した値を設定します
  例：2.5ns = 0x0000000000028000
  使用しない場合は0設定となります。

- **sourcePortIdentify**
  送信元のMACアドレスに0xfffeを挿入して作成したIEEE EUI-64拡張固有識別子を含む8バイトのフィールドと2バイトのポート番号(Default:1)を組み合わせた値を設定します。
  例：　送信元のMACアドレス:00-11-22-33-44-55
  　　sourcePortIdentity :00-11-22-ff-fe-33-44-55-00-01

- **sequenceID**
  PTPメッセージタイプに応じて以下のように指定します。
  Follow_Up：　Syncと同じ
  Delay_Resp：　Delay_Reqと同じ
  その他：　同じ宛先アドレスに送信された同じタイプの前のメッセージよりも1つ大きい値

- **controlField**
  PTPメッセージタイプに応じて以下のように指定します。
  Sync:　0x00
  Delay_Req:　0x01
  Follow_Up:　0x02
  Delay_Resp:　0x03
  Management:　0x04
  その他:　0x05

- **logMessageInterval**
  PTPメッセージタイプに応じて以下のように指定します。
  Sync, Follow_Up(mulitcast):　メッセージの平均間隔
  Sync, Follow_Up(unicast):　0x7F
  Delay_Resp(multicast):　最小のDelay_Req間隔
  Delay_Resp(unicast):　0x7F
  その他:　0x7F

Copyright © Infineon Technologies AG 2022. All rights reserved.

PTPメッセージタイプ

- PTPメッセージタイプとして以下が定義されています。
- 本ドキュメントではSync, Delay_Req, Follow_Up, Delay_Respの4種類について説明します。

ID	Type	Description
0x00	**Sync**	MasterからSlaveに送信される同期シーケンス開始メッセージです。Slaveは受信時刻(t2)を記録します。
0x01	**Delay_Req**	Follow_Upを受信したSlaveがMasterに送信します。Slaveは送信時刻(t3)を記録します。
0x02	Pdelay_Req	Peer-to-Peerの遅延メカニズムが使用される場合にのみ使用される遅延要求メッセージです。
0x03	Pdelay_Resp	Peer-to-Peerの遅延メカニズムが使用される場合にのみ使用される遅延応答メッセージです。
0x08	**Follow_Up**	MasterがSyncを送信した時刻(t1)をメッセージに乗せてSlaveに送信します(2Step時)
0x09	**Delay_Resp**	Delay_Reqを受信したMasterが受信時の時刻(t4)メッセージに乗せてSlaveに送信します。
0x0a	Pdelay_Follow_Up	Peer-to-Peerの遅延メカニズムが使用される場合にのみ使用される遅延測定用メッセージです。
0x0b	Announce	複数のMasterの中から1つのMasterを選択するために使用されます。
0x0c	Signaling	メッセージを送信する頻度や、ユニキャストをサポートすること等を伝えるために使用されます。
0x0d	Management	管理装置によって使用されます。

Copyright © Infineon Technologies AG 2022. All rights reserved.

PTPメッセージフォーマット (1/4)

- Syncメッセージ

	Octets	Offset
(PTP Header)	34	0
originTimestamp	10	34

  - **originTimestamp**
    MasterがSlaveに対しSyncメッセージを送出した時刻(t1)を設定します(1Step動作時)
    従ってハードウェアはメッセージ送出中(Ethernet Header送出時)に時刻を取得し、設定する必要があります。
    ハードウェアで対応できない場合は2Step動作として、MasterからSyncメッセージに続いて送出されるFollow_Upメッセージに対し、Syncメッセージを送出した時刻(t1)を設定します。
    その場合、SyncメッセージのoriginTimestampの設定値は0となります。

    Note:
    タイムスタンプはUNIXと同じエポック(1970/01/01 00:00:00)を使用する10byteのデータとなっており、6byteのsecondsと4byteのnanosecondsから構成されます。
    nanosecondsは1秒(=1000000000=0x3B9ACA00)以上の値をとることはできません。
    (その場合secondsを+1して、nanosecondsから0x3B9ACA00を減ずる必要があります)

    Note:
    Slave側ではSyncメッセージを受信した時刻(t2)を取得します。

Copyright © Infineon Technologies AG 2022. All rights reserved.

PTPメッセージフォーマット (2/4)

› Follow_Upメッセージ

	Octets	Offset
(PTP Header)	34	0
preciseOriginTimestamp	10	34

– **preciseOriginTimestamp**
2Step動作時にMasterがSlaveに対しSyncメッセージを送出した時刻(t1)を設定します。

Copyright © Infineon Technologies AG 2022. All rights reserved.

PTPメッセージフォーマット (3/4)

› Delay_Reqメッセージ

	Octets	Offset
(PTP Header)	34	0
originTimestamp	10	34

– **originTimestamp**
SlaveがMasterに対しDelay_Reqメッセージを送出した時刻(t3)を設定します。
但し、時刻差等の計算はSlave側で行うため、仕様では0設定が認められています。

Copyright © Infineon Technologies AG 2022. All rights reserved.

PTPメッセージフォーマット (4/4)

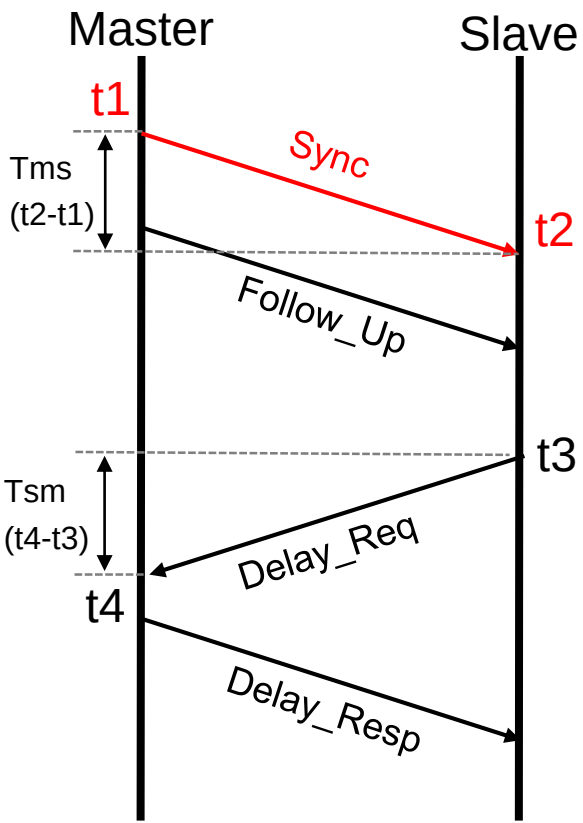
› Delay_Respメッセージ

	Octets	Offset
(PTP Header)	34	0
receiveTimestamp	10	34
requestingPortIdentity	10	44

- **receiveTimestamp**
  MasterがSlaveからDelay_Reqメッセージを受信した時刻(t4)を設定します。
- **requestingPortIdentity**
  MasterがSlaveから受信したDelay_ReqメッセージのsourcePortIdentity(PTP Header)を設定します。

Copyright © Infineon Technologies AG 2022. All rights reserved.

PTP同期シーケンス (1/4)



› 基本的なPTPの同期シーケンスを説明します。

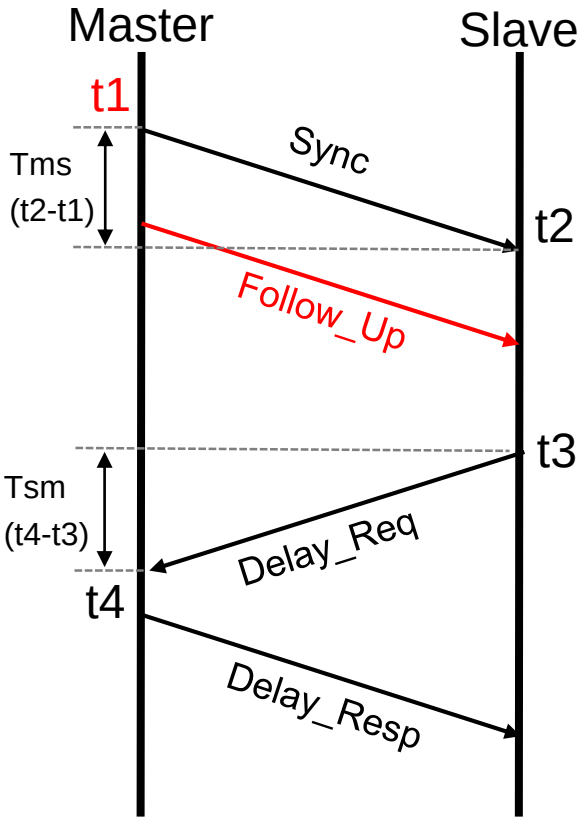
1. Syncメッセージの送出
MasterはSlaveに対しSyncメッセージをマルチキャストします。
1Step動作の場合、ハードウェアはSyncメッセージ送出中(Ethernet Header送出時)に(Master機器の)時刻(**t1**)を取得し、SyncメッセージのoriginTimestampに取得した時刻を設定します。
(ハードウェアが対応できない場合は2Step動作として、Follow_Upメッセージで時刻(**t1**)をSlaveに送信します)

SlaveはSyncメッセージ受信時に(Slave機器の)時刻(**t2**)を取得し、1Step動作の場合SyncメッセージのoriginTimestampを(**t1**)として記録します。

Note: 1Stepと2Stepのどちらかを判断するにはPTP Headerのflagsを参照します。

Copyright © Infineon Technologies AG 2022. All rights reserved.

PTP同期シーケンス (2/4)

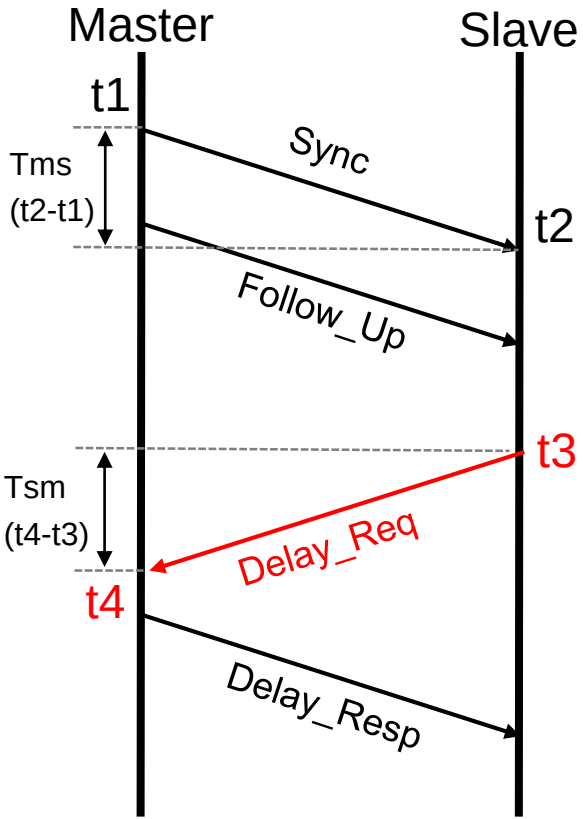


2. Follow_Upメッセージの送出
2Step動作の場合、Syncメッセージに続いてMasterはSlaveに対しFollow_Upメッセージをマルチキャストします。
Follow_UpメッセージのpreciseOriginTimestampには、Syncメッセージ送出中に取得した時刻(**t1**)を設定します。

Slaveは、2Step動作の場合Follow_Upメッセージの
preciseOriginTimestampを(**t1**)として記録します。

Copyright © Infineon Technologies AG 2022. All rights reserved.

PTP同期シーケンス (3/4)

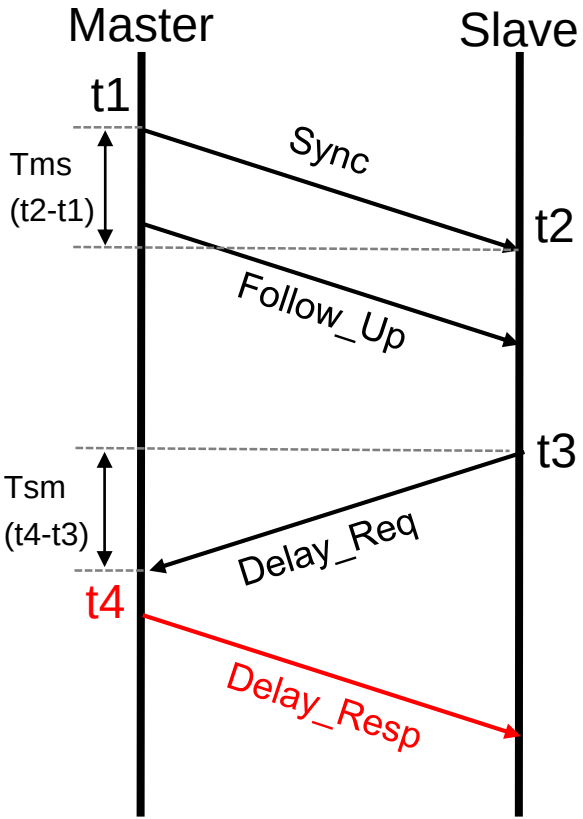


3. Delay_Reqメッセージの送出
1Step動作の場合はSyncメッセージを受信、2Step動作の場合はFollow_Upメッセージを受信した事を契機に、SlaveはMasterに対しDelay_Reqメッセージをマルチキャストします。
SlaveはDelay_Reqメッセージ送出時に(Slave機器の)時刻(**t3**)を取得します。

MasterはDelay_Reqメッセージ受信時に(Master機器の)時刻(**t4**)を取得します。

Copyright © Infineon Technologies AG 2022. All rights reserved.

PTP同期シーケンス (4/4)

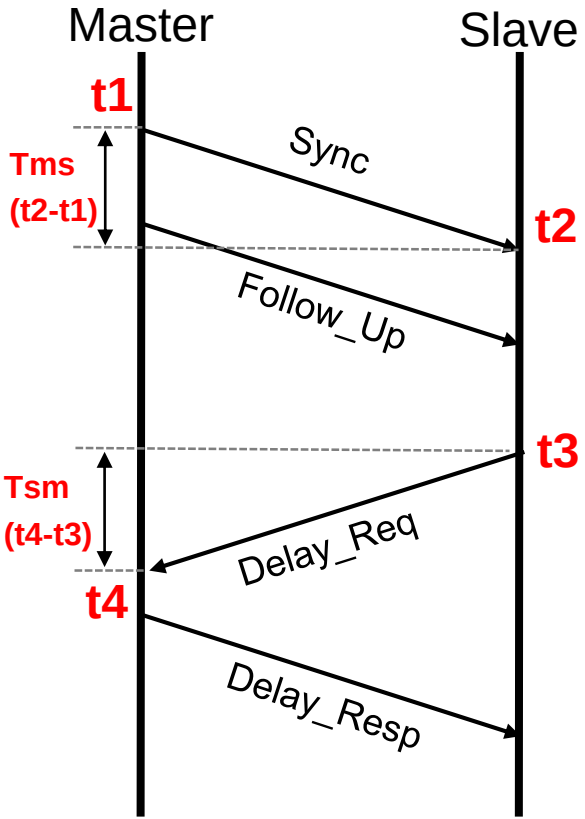


4. Delay_Respメッセージの送出
Delay_Reqメッセージを受信した事を契機に、MasterはSlaveに対しDelay_Respメッセージをマルチキャストします。
Delay_RespメッセージのreceiveTimestampにはDelay_Reqメッセージ受信時に取得した時刻(**t4**)を設定し、requestingPortIdentityにはSlaveから受信したDelay_ReqメッセージのsourcePortIdentity(PTP Header)を設定します。

SlaveはDelay_RespメッセージのreceiveTimestampを(**t4**)として記録します。

Copyright © Infineon Technologies AG 2022. All rights reserved.

Slaveによる時刻差、遅延時間の計算



PTP同期シーケンスにより、Slaveは **t1**, **t2**, **t3**, **t4**の4つのタイムスタンプを取得します。
このタイムスタンプを使用してSlaveは、Masterとの時刻差(Offset From Master)と、通信経路により生じる遅延時間(Mean Path Delay)を計算します。

遅延時間(Mean Path Delay):

```
meanPathDelay = (Tms + Tsm) / 2
              = ((t2 - t1) + (t4 - t3)) / 2
```

時刻差(Offset From Master):

```
offsetFromMaster = (t2 - t1) - meanPathDelay
```

Note:後述のコードptp.cではレジスタ仕様に対応する為、符号を反転させています。

```
offsetFromMaster = meanPathDelay - (t2 - t1)
```

Copyright © Infineon Technologies AG 2022. All rights reserved.

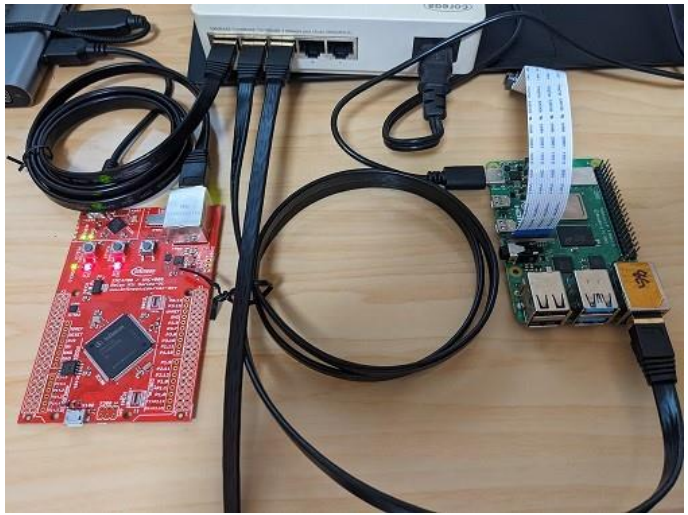
プロジェクト概要

Copyright © Infineon Technologies AG 2022. All rights reserved.

プロジェクト概要

- Raspberry PIをPTP Master, KIT_XMC47_RELAX_V1をPTP Slaveとして構成します。
  Note:写真ではHUBを経由していますが、ストレートケーブルで直接接続しても問題ありません。

- Slaveとして構成するKIT_XMC47_RELAX_V1の仕様は以下となります。
  - TCP/IP スタックとしてフリーライセンスでオープンソースの LwIPを使用
  - PTP over IEEE 802.3/Ethernetのみ対応
  - 2Step動作のみ対応
  - 動作状況はUSB UART出力



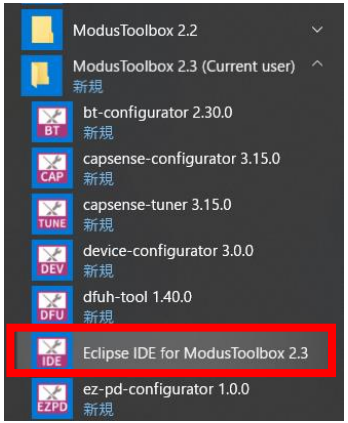
Copyright © Infineon Technologies AG 2022. All rights reserved.

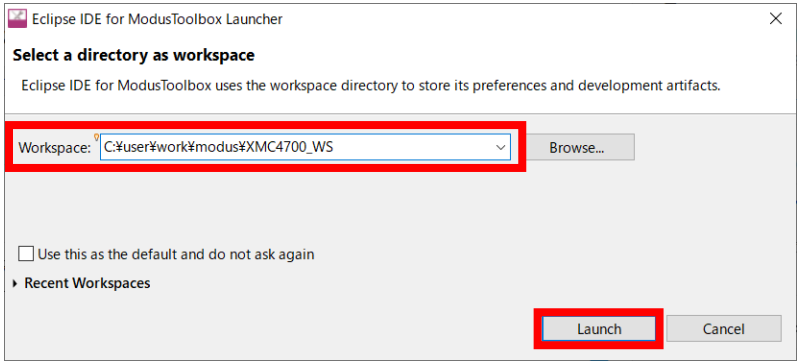
PTP Slaveプロジェクト作成手順

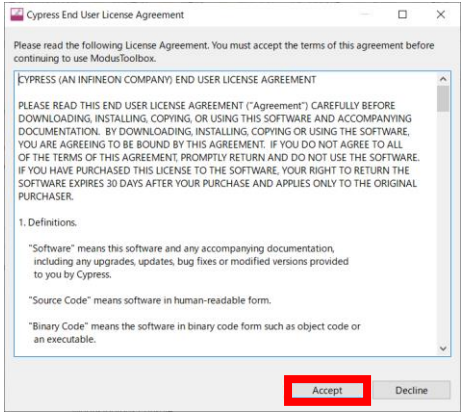
Copyright © Infineon Technologies AG 2022. All rights reserved.

ModusToolboxの起動

1. スタートメニューよりEclipse IDE for ModusToolbox 2.3をクリックしてModusToolboxを起動します。
2. プロジェクトを格納するワークスペース・フォルダを指定します(例：XMC4700_WS)
   Note:*フォルダ名、Pathには、マルチバイト文字を使用しないでください。*
3. Launchをクリックします。
   Note:*End User License Agreementのダイアログが表示された場合はAcceptをクリックしてください。*



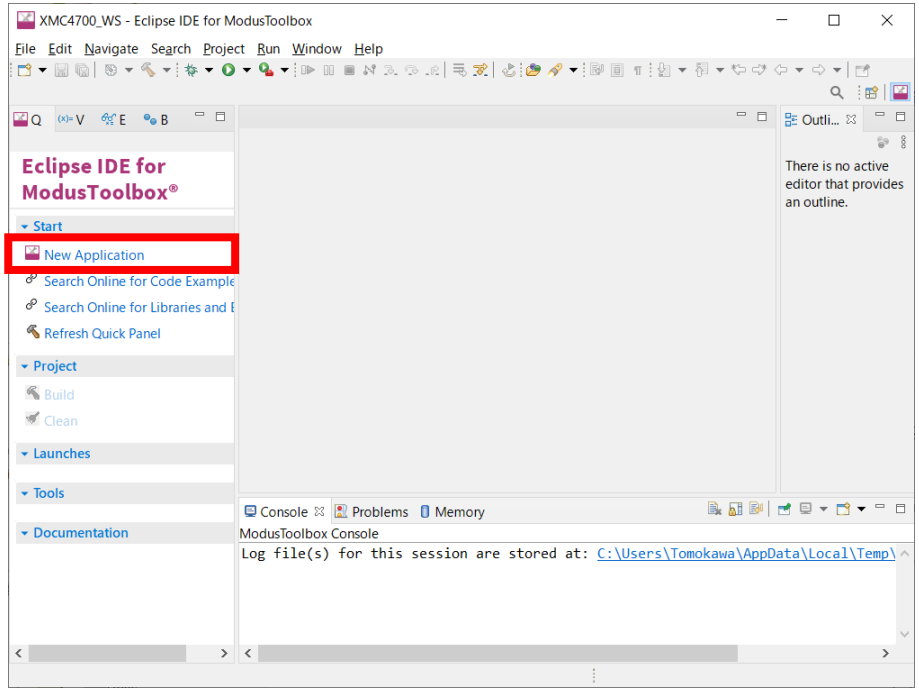


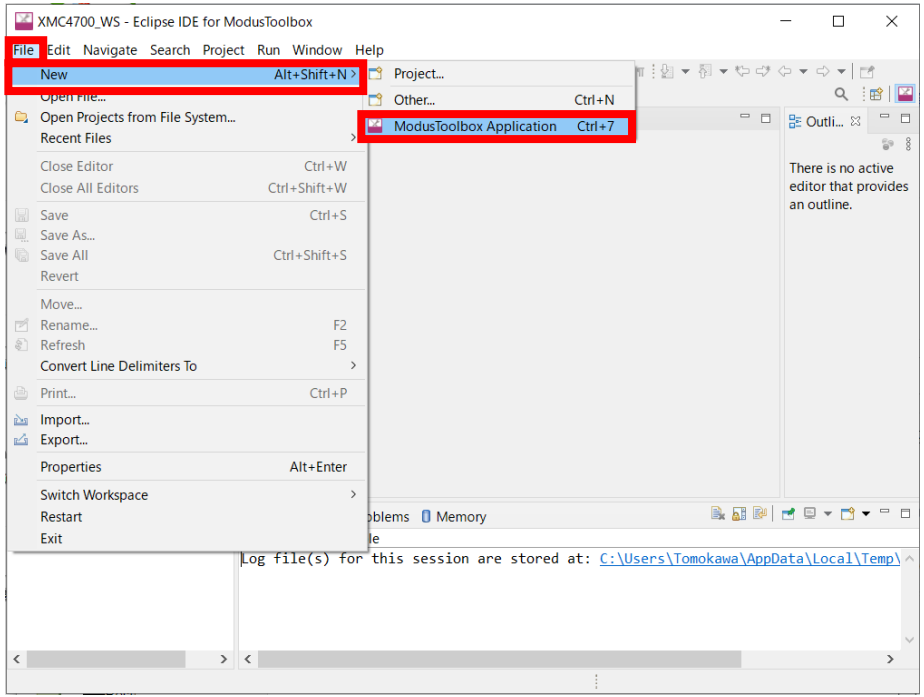


Copyright © Infineon Technologies AG 2022. All rights reserved.

プロジェクトの新規作成(1/3)

1. Quick panelのNew Appricationまたは、File → New → ModusToolbox Applicationを選択します。

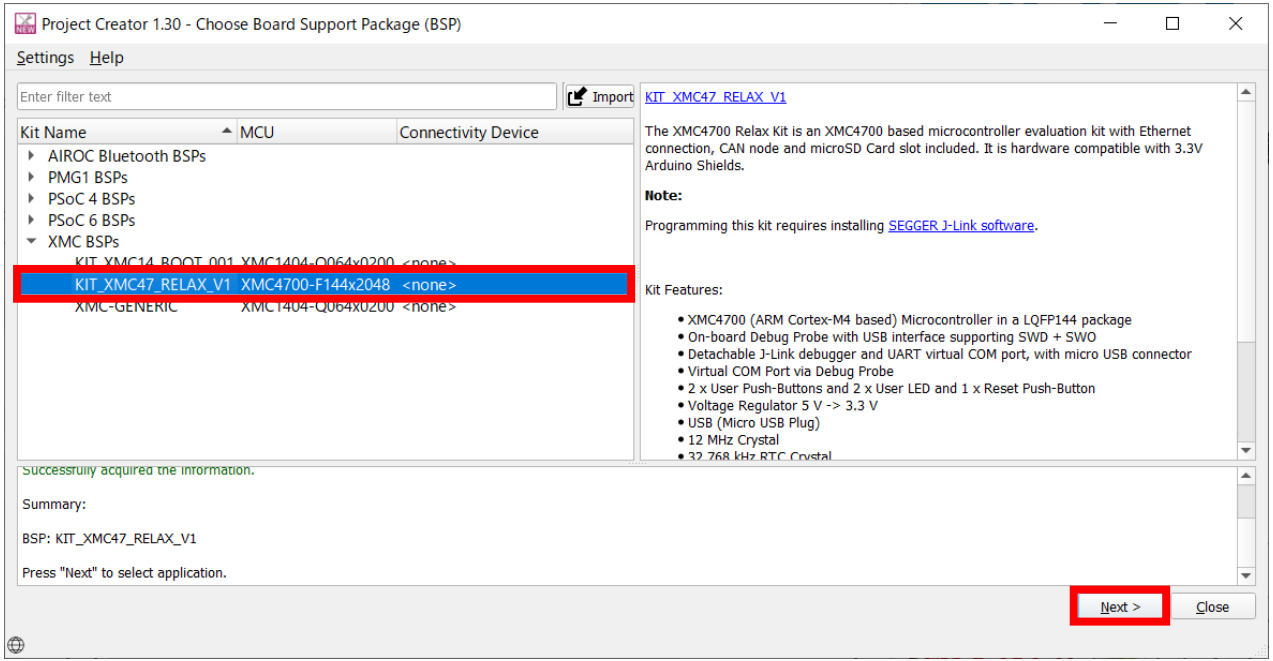




Copyright © Infineon Technologies AG 2022. All rights reserved.

プロジェクトの新規作成(2/3)

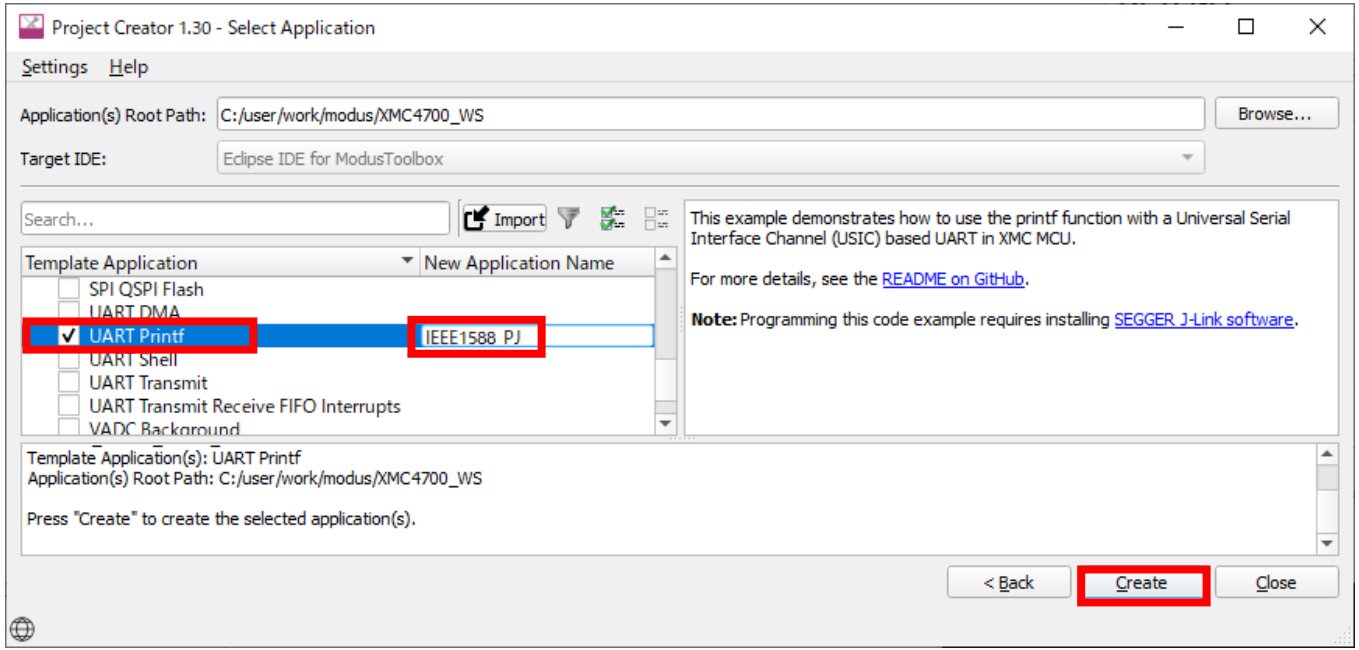
2. XMC　BSPs(Board Support Package)のKIT_XMC47_RELAX_V1を選択しNextをクリックします。



Copyright © Infineon Technologies AG 2022. All rights reserved.

プロジェクトの新規作成(3/3)

3. Template ApplicationでUART Printfをチェックします。
4. New Application NameにIEEE1588_PJを入力します。
5. Createをクリックします。



Copyright © Infineon Technologies AG 2022. All rights reserved.

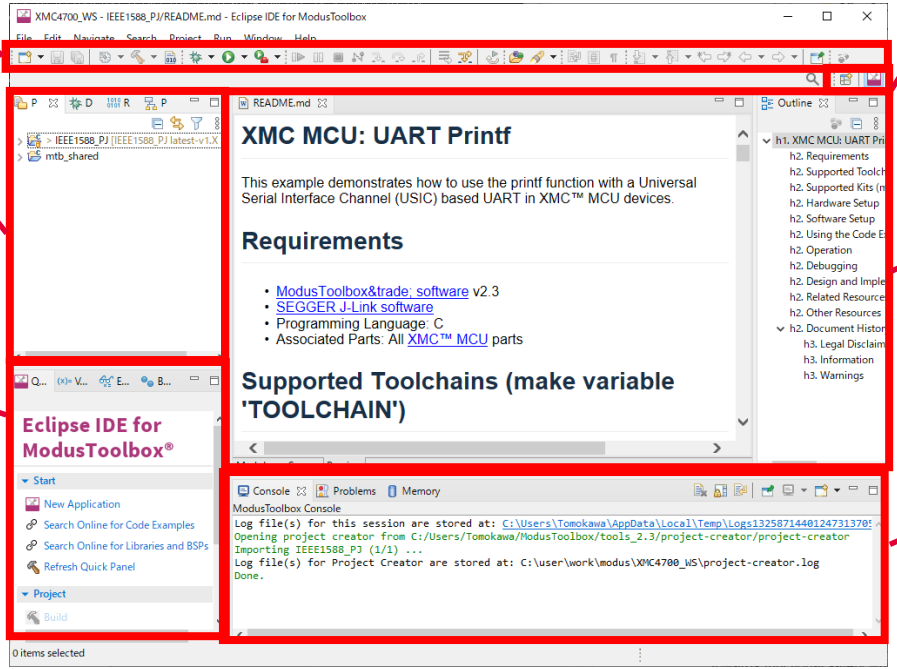
ModusToolBoxパースペクティブ

- プロジェクトを生成するとModusToolboxパースペクティブ(画面レイアウト)が表示されます。

Tool Panel

Project Explorer
プロジェクト構成
ファイルView

Quick Panel
Tool Panel同様のメニューへのクイックアクセスView



Perspective
(画面レイアウト)の切替

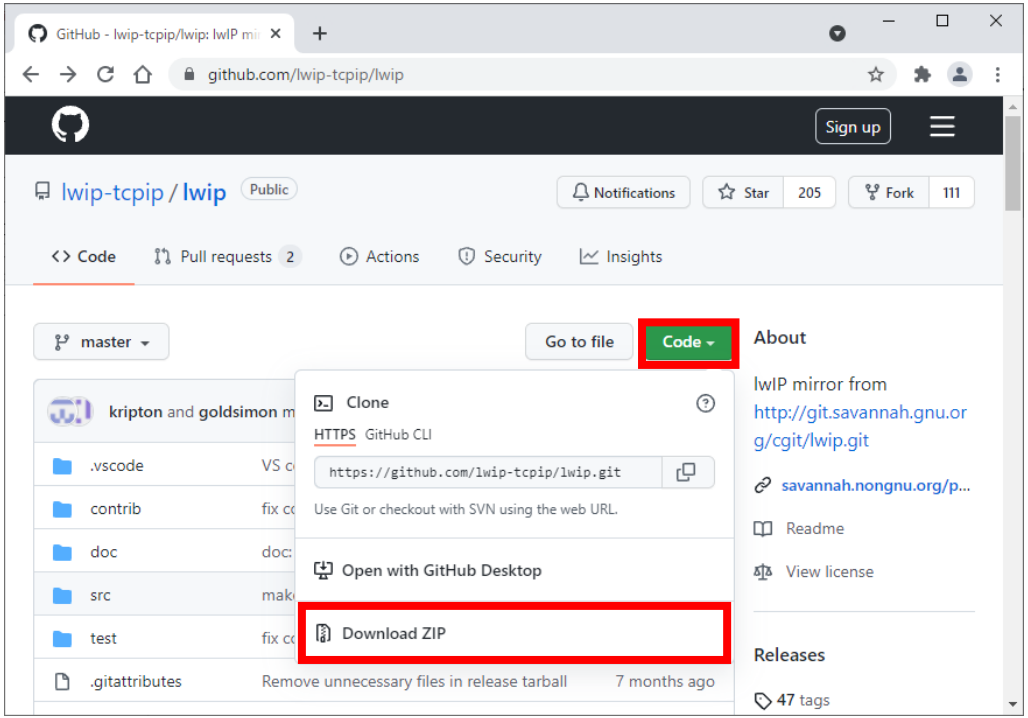
Code editor
コードエディタView

Console / Memory
コンソール / メモリView

Copyright © Infineon Technologies AG 2022. All rights reserved.

LwIP(lightweight IP)のプロジェクトへの追加(1/7)

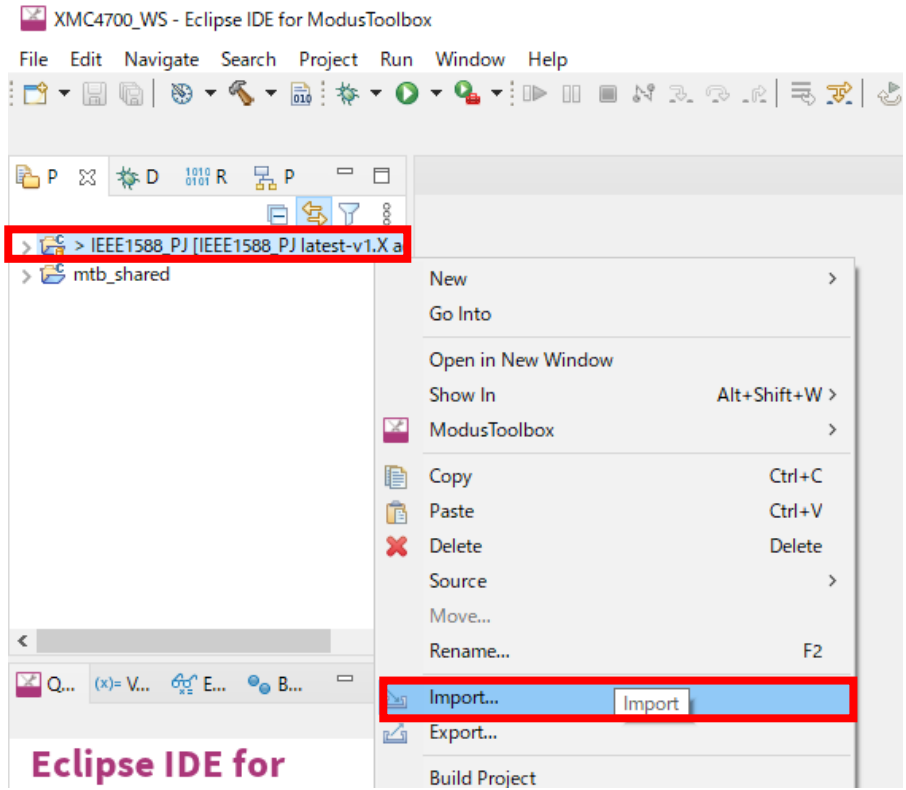
- LwIPとは、フリーライセンスでオープンソースの TCP/IP スタックです。
- このLwIPをダウンロードし、プロジェクトに追加します。



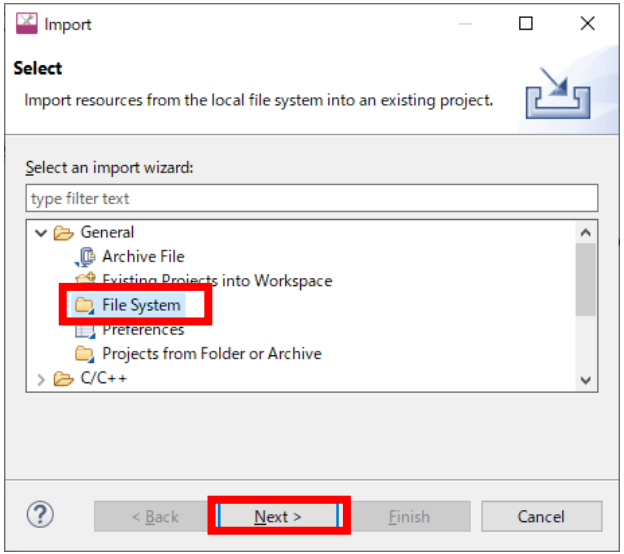
1. ブラウザで https://github.com/lwip-tcpip/lwip にアクセスし、Codeをクリックします。
2. Download ZIPをクリックしてダウンロードし、lwip-master.zipファイルを展開します。

Copyright © Infineon Technologies AG 2022. All rights reserved.

LwIP(lightweight IP)のプロジェクトへの追加(2/7)

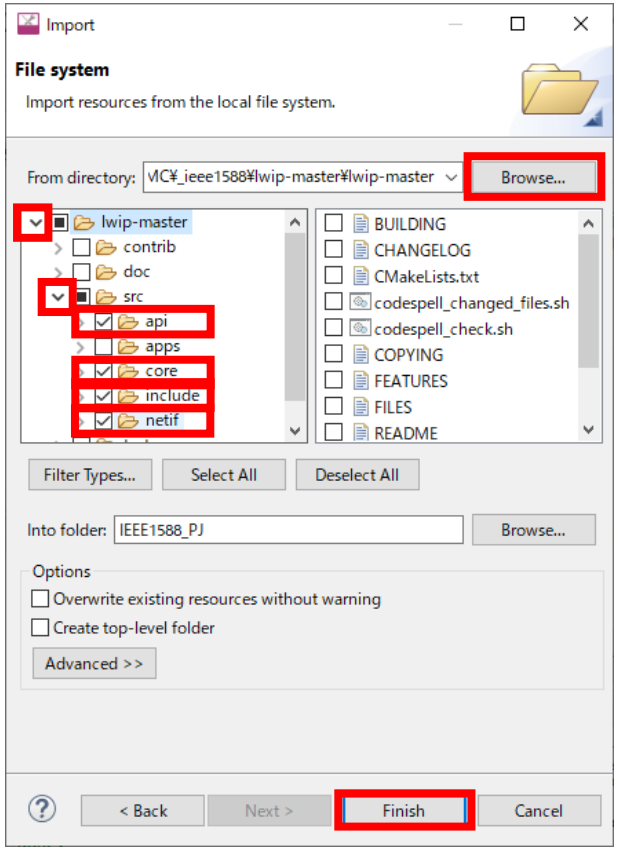


3. Project ExplorerのIEEE1588_PJ上で右クリックしてImportを選択します。
4. General→File Systemを選択してNextをクリックします。



Copyright © Infineon Technologies AG 2022. All rights reserved.

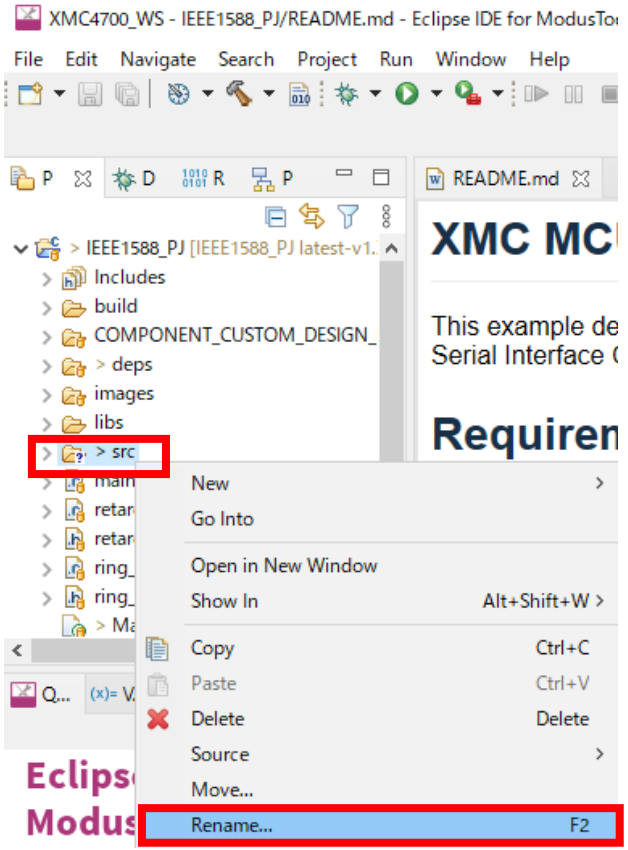
LwIP(lightweight IP)のプロジェクトへの追加(3/7)



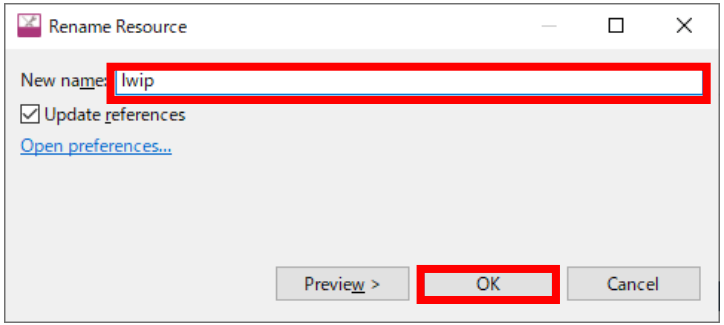
5. BrowseをクリックしてFrom directoryにlwip-master.zipを展開したPathを入力します。
6. lwip-masterとsrcフォルダの左端をクリックして展開し、appsを除く全てのフォルダ(api, core, include, netif)をチェックします。
7. Finishをクリックします。

Copyright © Infineon Technologies AG 2022. All rights reserved.

LwIP(lightweight IP)のプロジェクトへの追加(4/7)



8. Project ExplorerのIEEE1588_PJ以下にImportされたsrcフォルダ上で右クリックしてRenameを選択します。
9. New nameにlwipと入力し、OKをクリックします。



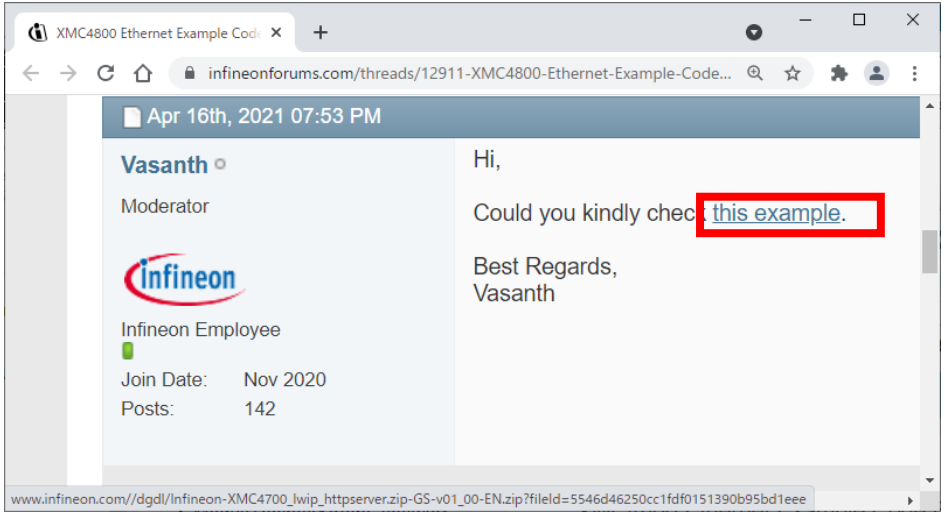
Copyright © Infineon Technologies AG 2022. All rights reserved.

LwIP(lightweight IP)のプロジェクトへの追加(5/7)

10. Infineonのフォーラムhttps://www.infineonforums.com/threads/12911-XMC4800-Ethernet-Example-Code-For-TCP-IP-DHCP-using-XMClib-without-RTOS?から Infineon-XMC4700_lwip_httpserver.zip-GS-v01_00-EN.zip をダウンロードして展開します。

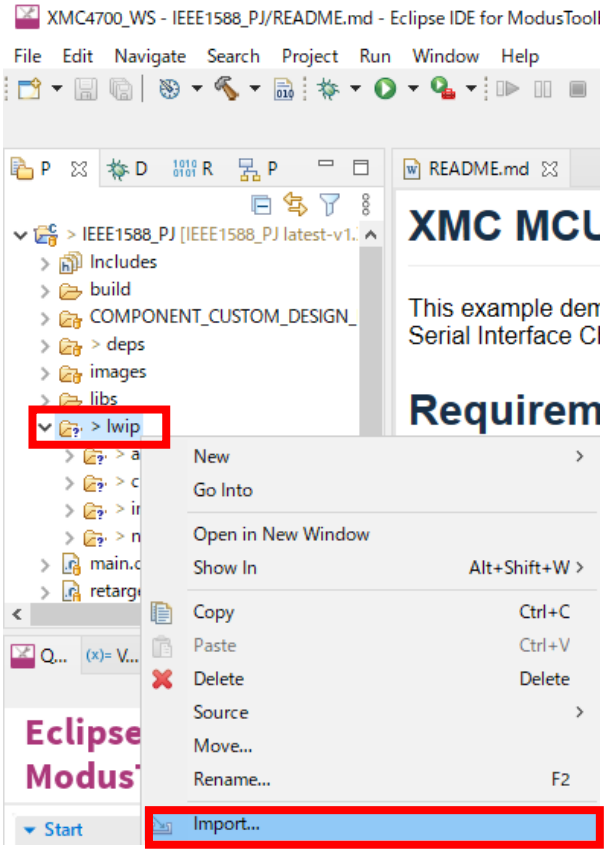
    Note: *フォーラムのアカウントを作成する必要があります。また、chromeブラウザでは正しくダウンロードできない場合があります。その場合は、Edge等他のブラウザを使用してください。*

    http://www.infineon.com//dgdl/Infineon-XMC4700_lwip_httpserver.zip-GS-v01_00-EN.zip?fileId=5546d46250cc1fdf0151390b95bd1eee

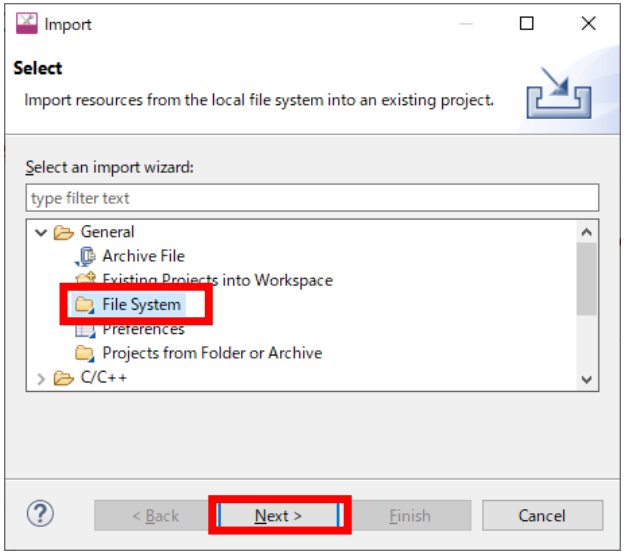


Copyright © Infineon Technologies AG 2022. All rights reserved.

LwIP(lightweight IP)のプロジェクトへの追加(6/7)

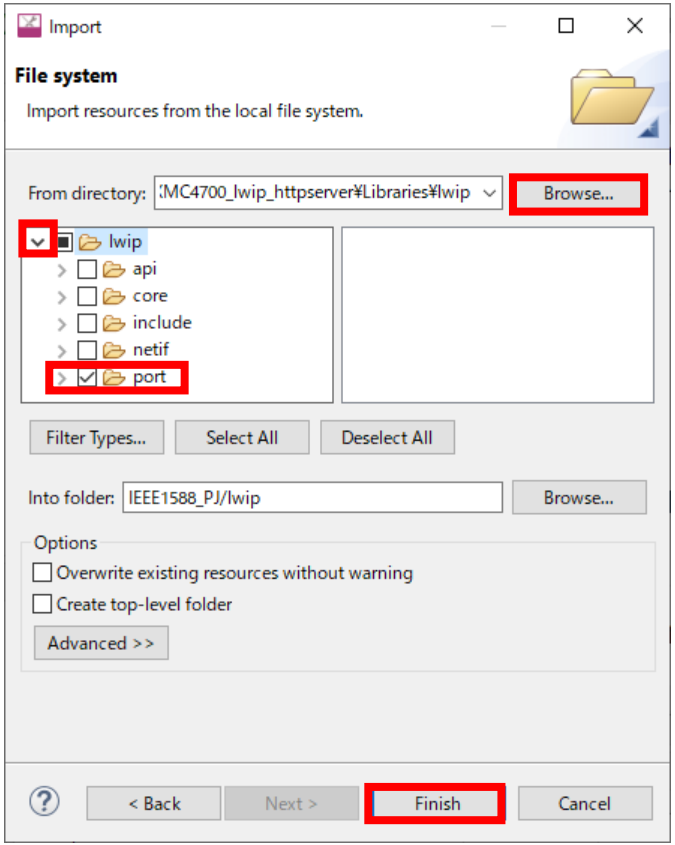


11. Project ExplorerのIEEE1588_PJ以下のlwipフォルダ上で右クリックしてImportを選択します。
12. General→File Systemを選択してNextをクリックします。



Copyright © Infineon Technologies AG 2022. All rights reserved.

LwIP(lightweight IP)のプロジェクトへの追加(7/7)



5. Browseをクリックして From directoryにInfineon-XMC4700_lwip_httpserver.zip-GS-v01_00-EN.zipを展開したフォルダ内のLibraries¥lwipのPathを入力します。
   (例：C:¥Infineon-XMC4700_lwip_httpserver.zip-GS-v01_00-EN¥XMC4700_lwip_httpserver¥Libraries¥lwip)
6. lwipフォルダの左端をクリックして展開し、portフォルダをチェックします。
7. Finishをクリックします。

Note: *portフォルダ以下のファイルには低レベルI/O(XMClibへのアクセス関数)が記述されています。*

Copyright © Infineon Technologies AG 2022. All rights reserved.

LwIP(lightweight IP) : cc.hの修正

cc.h

```
54  //typedef unsigned   long    mem_ptr_t;
55
56  /* Define (sn)printf formatters for these lwIP types */
57  #define X8_F  "02x"
58  #define U16_F "u"
59  #define S16_F "d"
60  #define X16_F "x"
61  #define U32_F "u"
62  #define S32_F "d"
63  #define X32_F "x"
64  #define SZT_F "u"
```

1. XMC4700_WS¥IEEE1588_PJ¥lwip¥port¥nosys¥include¥arch¥cc.hで定義されているmem_ptr_tをコメントアウトし、ファイルを保存します。

***Note**: 行番号を表示するには、コードエディタViewの左端でマウスを右クリックしてメニューを表示し、Show Line Numbersを選択します。*

Copyright © Infineon Technologies AG 2022. All rights reserved.

LwIP(lightweight IP) : opt.hの修正

opt.h

```
205 /**
206  * SYS_LIGHTWEIGHT_PROT==1: enable inter-task protection (and task-vs-interrupt
207  * protection) for certain critical regions during buffer allocation, deallocation
208  * and memory allocation and deallocation.
209  * ATTENTION: This is required when using lwIP from more than one context! If
210  * you disable this, you must be sure what you are doing!
211  */
212 #if !defined SYS_LIGHTWEIGHT_PROT || defined __DOXYGEN__
213 #define SYS_LIGHTWEIGHT_PROT            0
214 #endif
215
```

1. XMC4700_WS¥IEEE1588_PJ¥lwip¥include¥lwip¥opt.hで定義されている SYS_LIGHTWEIGHT_PROTの定義値を1から0に変更し、ファイルを保存します。

Copyright © Infineon Technologies AG 2022. All rights reserved.

LwIP(lightweight IP) : ethernetif.cの修正(1/4)

ethernetif.c

```
41  #include "netif/etharp.h"
42  //#include "netif/ppp_oe.h"
43  #include "ptp.h"
44
45  #include "xmc_gpio.h"
```

ethernetif.c

```
96  XMC_ETH_MAC_t eth_mac =
97  {
```

ethernetif.c

```
110 extern struct netif xnetif;
111
112 void getMacAddr( struct eth_addr *mac ){
113     mac->addr[0] = MAC_ADDR5;
114     mac->addr[1] = MAC_ADDR4;
115     mac->addr[2] = MAC_ADDR3;
116     mac->addr[3] = MAC_ADDR2;
117     mac->addr[4] = MAC_ADDR1;
118     mac->addr[5] = MAC_ADDR0;
119 }
120
```

1. XMC4700_WS¥IEEE1588_PJ¥lwip¥port¥nosys¥netif¥ethernetif.cでincludeされているppp_oe.hをコメントアウトし、ptp.hを追加します。

2. eth_mac変数を外部参照する為static修飾子を削除します。

3. 外部よりMAC Addressを参照する為、下記の関数を定義します。

```
void getMacAddr( struct eth_addr *mac ){
    mac->addr[0] = MAC_ADDR5;
    mac->addr[1] = MAC_ADDR4;
    mac->addr[2] = MAC_ADDR3;
    mac->addr[3] = MAC_ADDR2;
    mac->addr[4] = MAC_ADDR1;
    mac->addr[5] = MAC_ADDR0;
}
```

Copyright © Infineon Technologies AG 2022. All rights reserved.

LwIP(lightweight IP) : ethernetif.cの修正(2/4)

ethernetif.c

```
170   XMC_ETH_MAC_Init(&eth_mac);
171   XMC_ETH_MAC_InitPTP(&eth_mac, XMC_ETH_MAC_TIMESTAMP_CONFIG_FINE_UPDATE
172                               | XMC_ETH_MAC_TIMESTAMP_CONFIG_ENABLE_PTPV2
173                               | XMC_ETH_MAC_TIMESTAMP_CONFIG_ENABLE_ALL_FRAMES
174                               | XMC_ETH_MAC_TIMESTAMP_CONFIG_ENABLE_PTP_OVER_ETHERNET);
175   XMC_ETH_MAC_EnableReceptionMulticastFrames(&eth_mac);
176
177   XMC_ETH_MAC_DisableJumboFrame(&eth_mac);
```

4. low_level_init()関数内のXMC_ETH_MAC_Init()関数呼び出しの後にXMC_ETH_MAC_InitPTP()関数とXMC_ETH_MAC_EnableReceptionMulticastFrames()関数の呼び出しを追加します。

```
XMC_ETH_MAC_InitPTP(&eth_mac, XMC_ETH_MAC_TIMESTAMP_CONFIG_FINE_UPDATE
                            | XMC_ETH_MAC_TIMESTAMP_CONFIG_ENABLE_PTPV2
                            | XMC_ETH_MAC_TIMESTAMP_CONFIG_ENABLE_ALL_FRAMES
                            | XMC_ETH_MAC_TIMESTAMP_CONFIG_ENABLE_PTP_OVER_ETHERNET);
XMC_ETH_MAC_EnableReceptionMulticastFrames(&eth_mac);
```

Copyright © Infineon Technologies AG 2022. All rights reserved.

LwIP(lightweight IP) : ethernetif.cの修正(3/4)

ethernetif.c

```
248
249 #if ETH_PAD_SIZE
250   pbuf_header(p, ETH_PAD_SIZE);     /* Reclaim the padding word */
251 #endif
252
253   status = XMC_ETH_MAC_SendFrame(&eth_mac, buffer, framelen, XMC_ETH_MAC_TX_FRAME_TIMESTAMP);
254   if (status != XMC_ETH_MAC_STATUS_OK)
255   {
```

5. low_level_output()関数内のXMC_ETH_MAC_SendFrame()関数の最後の引数を0からXMC_ETH_MAC_TX_FRAME_TIMESTAMPに変更します。

Copyright © Infineon Technologies AG 2022. All rights reserved.

LwIP(lightweight IP) : ethernetif.cの修正(4/4)

ethernetif.c

```
345       switch (htons(ethhdr->type))
346       {
347         case ETHTYPE_IP:
348         case ETHTYPE_ARP:
349           /* full packet send to tcpip_thread to process */
350         if (netif->input( p, netif) != ERR_OK)
351         {
352           pbuf_free(p);
353         }
354
355         break;
356         case ETHTYPE_PTP:
357         if (ptp_input( p, netif) != ERR_OK){
358             pbuf_free(p);
359         }
360         break;
361
362         default:
```

ethernetif.c

```
392 #if LWIP_NETIF_HOSTNAME
393   /* Initialize interface hostname */
394 //  netif->hostname = "lwip";
395 #endif /* LWIP_NETIF_HOSTNAME */
396
```

6. ethernetif_input()関数内のswitch文内にETHTYPE_PTPを追加します。

```
case ETHTYPE_PTP:
if (ptp_input( p, netif) != ERR_OK){
    pbuf_free(p);
}
break;
```

7. ethernetif_init()関数内のnetif->hostnameへの代入文をコメントアウトし、ファイルを保存します。

Copyright © Infineon Technologies AG 2022. All rights reserved.

XMCLib: xmc_eth_phy_ksz8081rnb.cの修正

xmc_eth_phy_ksz8081rnb.c

```
59
60 /*******************************************************************************
61  * HEADER FILES
62  *******************************************************************************/
63 #define XMC_ETH_PHY_KSZ8081RNB
64 #if defined(XMC_ETH_PHY_KSZ8081RNB)
65 #include "xmc_eth_phy.h"
66
67 /*******************************************************************************
68  * MACROS
69  *******************************************************************************/
```

1. XMC4700_WS¥mtb_shared¥mtb-xmclib-cat3¥release-v3.0.0¥XMCLib¥src¥xmc_eth_phy_ksz8081rnb.c内の #if defined(XMC_ETH_PHY_KSZ8081RNB) の前でXMC_ETH_PHY_KSZ8081RNB を定義し、以降に記載されているコードを有効化して、ファイルを保存します。

Copyright © Infineon Technologies AG 2022. All rights reserved.

NewLib: syscall.cの修正(1/2)

syscalls.c

```
111 /* Init */
112 //void _init(void)
113 //{}
114
115 __attribute__((weak)) int _open(const char *name, int flags, int mode){
116     (void)flags;
117     (void)mode;
118     return -1;
119 }
```

:

syscalls.c

```
187 __attribute__((weak)) int _isatty(int file){
188     (void)file;
189     return -1;
190 }
191
192 #endif /* __GNUC__ */
```

1. XMC4700_WS¥mtb_shared¥mtb-xmclib-cat3¥release-v3.0.0¥Newlib¥syscall.c内の _init()関数をコメントアウトし、以降に16個の関数を追加し、ファイルを保存します。
   Note:追加関数は次項に示します。

Copyright © Infineon Technologies AG 2022. All rights reserved.

NewLib: syscall.cの修正(2/2)

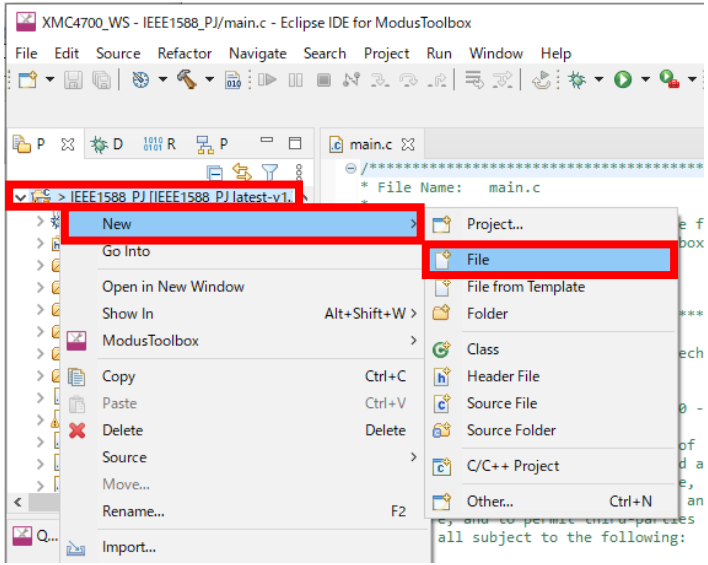
```
__attribute__((weak)) int _open(const char *name, int flags, int mode){
    (void)flags;
    (void)mode;
    return -1;
}

__attribute__((weak)) int _lseek(int file, int offset, int whence){
    (void)file;
    (void)offset;
    (void)whence;
    return -1;
}

__attribute__((weak)) int _read(int file, char *ptr, int len){
    (void)file;
    (void)len;
    return 0;
}

__attribute__((weak)) int _write(int file, char *buf, int nbytes){
    return -1;
}

__attribute__((weak)) int _close(void){
    return -1;
}

__attribute__((weak)) int _fstat(int file, struct stat *st){
    (void)file;
    if(st) return -1;
    else   return -2;
}

__attribute__((weak)) int _link(char *oldname, char *newname){
    if (oldname == newname) return -1;
    else                    return -2;
}
```

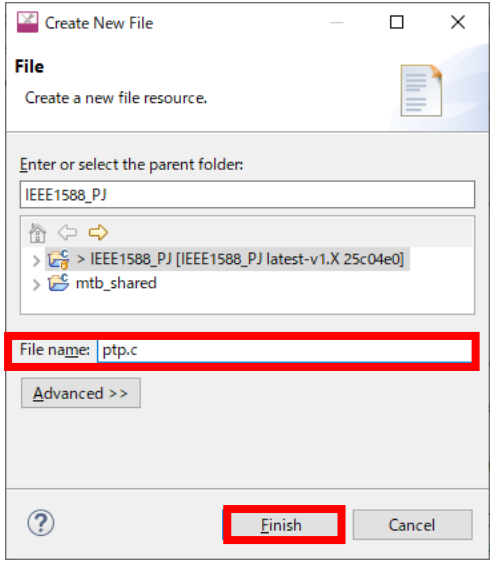
```
__attribute__((weak)) int _unlink(char *name){
    return -1;
}

__attribute__((weak)) int _times(struct tms *buf){
    return -1;
}

__attribute__((weak)) int _wait(int *status){
    return -1;
}

__attribute__((weak)) int _kill(int pid,int sig){
    (void)pid;
    (void)sig;
    return -1;
}

__attribute__((weak)) int _fork(void){
    return -1;
}

__attribute__((weak)) int _getpid(void){
    return -1;
}

__attribute__((weak)) void _exit(int rc){
    (void)rc;
    while(1){}
}

__attribute__((weak)) void _init(void){
}

__attribute__((weak)) int _isatty(int file){
    (void)file;
    return -1;
}
```

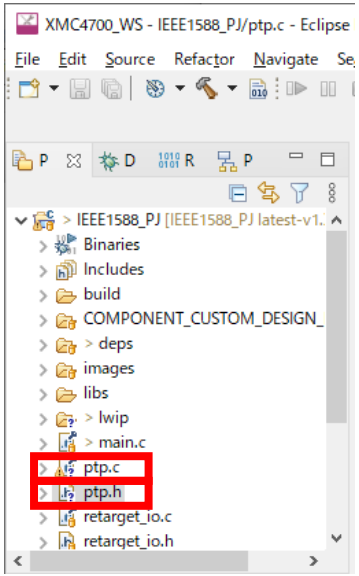
Copyright © Infineon Technologies AG 2022. All rights reserved.

ptp.c及びptp.hの作成

1. Project ExplorerのIEEE1588_PJ上で右クリックしてNew→Fileを選択します。
2. File name:にptp.cを入力してFinishをクリックします。同様にptp.hを作成します。
3. Project Explorerでptp.c及びptp.hをダブルクリックしてファイルを開き、次項以降に記載した内容を入力し、ファイルを保存します。







Copyright © Infineon Technologies AG 2022. All rights reserved.

ptp.h

```
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
// IEEE1588 PTP(Precision Time Protocol) header
// Ver. 1.0 20210930
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
#include "lwip/opt.h"
#include "lwip/def.h"
#include "lwip/pbuf.h"
#include "lwip/ip_addr.h"
#include "lwip/err.h"
#include "lwip/netif.h"
#include "netif/etharp.h"

#ifdef __cplusplus
extern "C" {
#endif

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
//  Define declaration
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
typedef struct {
    uint8_t  transSpec_msgType;
    uint8_t  versionPTP;
    uint16_t messageLength;
    uint8_t  domainNumber;
    uint8_t  reserved0;
    uint16_t flags;
    uint8_t  correctionField[8];
    uint8_t  reserved1[4];
    uint8_t  sourcePortIdentity[10];
    uint16_t sequenceId;
    uint8_t  control;
    uint8_t  logMeanMessageInterval;
}PTP_hdr_t;

#define PTP_HDR_messageType(hdr) ((hdr)->transSpec_msgType & 0x0f)
#define PTP_HDR_versionPTP(hdr) ((hdr)->versionPTP & 0x0f)
```

```
#define PTP_MSG_Sync                0x00
#define PTP_MSG_Delay_Req           0x01
#define PTP_MSG_Pdelay_Req          0x02
#define PTP_MSG_Pdelay_Resp         0x03
#define PTP_MSG_Reserved_4          0x04
#define PTP_MSG_Reserved_5          0x05
#define PTP_MSG_Reserved_6          0x06
#define PTP_MSG_Reserved_7          0x07
#define PTP_MSG_Follow_Up           0x08
#define PTP_MSG_Delay_Resp          0x09
#define PTP_MSG_Pdelay_Follow_Up    0x0a
#define PTP_MSG_Announce            0x0b
#define PTP_MSG_Signaling           0x0c
#define PTP_MSG_Management          0x0d
#define PTP_MSG_Reserved_E          0x0e
#define PTP_MSG_Reserved_F          0x0f

#define PTP_MSGSIZ_Delay_Req        10

#ifndef __PTP_H__
#define __PTP_H__
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
// Variable declaration
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
extern uint32_t _ptp_sequenceId;

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
// Function declaration
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
extern err_t ptp_input(struct pbuf *p, struct netif *netif);

#endif /*#ifndef  PTP_H */

#ifdef __cplusplus
}
#endif
```

Copyright © Infineon Technologies AG 2022. All rights reserved.

ptp.c (1/5)

```
//%%%
```

ptp.c (2/5)

```
//=============================================================================
// Subtraction Of XMC_ETH_MAC_TIME_t Type Variable (pTr = pTa - pTb)
// XMC_ETH_MAC_TIME_t *pTr  : Pointer for difference
// XMC_ETH_MAC_TIME_t *pTa  : Pointer for minuend
// XMC_ETH_MAC_TIME_t *pTb  : Pointer for subtrahend
// return                   : void
//=============================================================================
static void calcTimeSub( XMC_ETH_MAC_TIME_t *pTr, XMC_ETH_MAC_TIME_t *pTa,
XMC_ETH_MAC_TIME_t *pTb ){

    pTr->seconds     = pTa->seconds     - pTb->seconds;
    pTr->nanoseconds = pTa->nanoseconds - pTb->nanoseconds;
    if(pTr->nanoseconds < 0){
        pTr->nanoseconds += 1000000000;
        pTr->seconds     -= 1;
    }
}

//=============================================================================
// Addition Of XMC_ETH_MAC_TIME_t Type Variable (pTr = pTa + pTb)
// XMC_ETH_MAC_TIME_t *pTr  : Pointer for difference
// XMC_ETH_MAC_TIME_t *pTa  : Pointer for minuend
// XMC_ETH_MAC_TIME_t *pTb  : Pointer for subtrahend
// return                   : void
//=============================================================================
static void calcTimeAdd( XMC_ETH_MAC_TIME_t *pTr, XMC_ETH_MAC_TIME_t *pTa,
XMC_ETH_MAC_TIME_t *pTb ){

    pTr->seconds     = pTa->seconds     + pTb->seconds;
    pTr->nanoseconds = pTa->nanoseconds + pTb->nanoseconds;
    if(pTr->nanoseconds >= 1000000000){
        pTr->nanoseconds -= 1000000000;
        pTr->seconds     += 1;
    }
}
```

```
//=============================================================================
// Send Delay_Req packet
// netif *netif : Pointer to the network interfaces
// return       : == ERR_OK:success, != ERR_OK:fail (see also err.h)
//=============================================================================
err_t ptp_send_Delay_Req(struct netif *netif){
uint32_t           i;
struct pbuf        *pSend;
struct eth_hdr     *ethhdr;
PTP_hdr_t          *ptphdr;
//uint8_t            *ptpMsg;

    // allocate a buffer
    pSend = pbuf_alloc(PBUF_LINK, (u16_t)(sizeof(struct eth_hdr) +
sizeof(PTP_hdr_t) + PTP_MSGSIZ_Delay_Req), PBUF_RAM);
    if (!pSend) {
        return ERR_MEM;
    }
    // init buffer
    for( i = 0; i < sizeof(struct eth_hdr) + sizeof(PTP_hdr_t) +
PTP_MSGSIZ_Delay_Req; i ++ ){
        ((u8_t *)pSend->payload)[i] = 0;
    }

    // identify the Ether header
    ethhdr = (struct eth_hdr *)pSend->payload;

    // Create Ether Header
    ethhdr->dest.addr[0] = PTP_MULTICAST_ADDR_0;
    ethhdr->dest.addr[1] = PTP_MULTICAST_ADDR_1;
    ethhdr->dest.addr[2] = PTP_MULTICAST_ADDR_2;
    ethhdr->dest.addr[3] = 0x00;
    ethhdr->dest.addr[4] = 0x00;
    ethhdr->dest.addr[5] = 0x00;

    getMacAddr( (struct eth_addr *)ethhdr->src.addr );
```

Copyright © Infineon Technologies AG 2022. All rights reserved.

ptp.c (3/5)

```
    ethhdr->type = PP_HTONS( ETHTYPE_PTP );

    // identify the PTP header
    ptphdr = (PTP_hdr_t *)(pSend->payload + sizeof (struct eth_hdr));

    // Create PTP header
    ptphdr->transSpec_msgType      = PTP_MSG_Delay_Req;
    ptphdr->versionPTP             = 0x02;
    ptphdr->messageLength          = PP_HTONS(sizeof(PTP_hdr_t) +
PTP_MSGSIZ_Delay_Req);

    // 8byte field containing the IEEE EUI-64 extended unique identifier created by
inserting 0xfffe
    // in hexadecimal into the source MAC address, and a 2byte port number
(Default:1).
    // e.g. Source MAC Address :00-11-22-33-44-55
    //      sourcePortIdentity :00-11-22-ff-fe-33-44-55-00-01
    ptphdr->sourcePortIdentity[0]  = ethhdr->src.addr[0];
    ptphdr->sourcePortIdentity[1]  = ethhdr->src.addr[1];
    ptphdr->sourcePortIdentity[2]  = ethhdr->src.addr[2];
    ptphdr->sourcePortIdentity[3]  = 0xff;
    ptphdr->sourcePortIdentity[4]  = 0xfe;
    ptphdr->sourcePortIdentity[5]  = ethhdr->src.addr[3];
    ptphdr->sourcePortIdentity[6]  = ethhdr->src.addr[4];
    ptphdr->sourcePortIdentity[7]  = ethhdr->src.addr[5];
    ptphdr->sourcePortIdentity[8]  = 0x00;
    ptphdr->sourcePortIdentity[9]  = 0x01;
    ptphdr->sequenceId             = PP_HTONS( ++ _ptp_sequenceId );
    ptphdr->control                = 0x01;//Delay_Req
    ptphdr->logMeanMessageInterval = 0x7f;//Delay_Req

    // Identify the PTP message
//    ptpMsg = (u8_t *)(pSend->payload + sizeof (struct eth_hdr) + sizeof
(PTP_hdr_t));
    // Delay_Req PTP message All zero
```

```
    // Send packet
    err_t ret = netif->linkoutput(netif, pSend);

    pbuf_free(pSend);
    return ret;
}

//============================================================================
// Receive & Decode PTP packet
// pbuf *p      : Pointer to the ether net packet buffer
// netif *netif : Pointer to the network interfaces
// return       : == ERR_OK:success, != ERR_OK:fail (see also err.h)
//============================================================================
err_t ptp_input(struct pbuf *p, struct netif *netif){
struct eth_hdr     *ethhdr;
PTP_hdr_t          *ptphdr;
void               *ptpMsg;
XMC_ETH_MAC_TIME_t meanPathDelay;
XMC_ETH_MAC_TIME_t offsetFromMaster;
XMC_ETH_MAC_TIME_t t2_1, t4_3, t_result;

    if (p->len <= SIZEOF_ETH_HDR) {
        goto free_and_return;
    }

    // points to packet payload, which starts with an Ethernet header
    ethhdr = (struct eth_hdr *)p->payload;

#ifdef _PTP_DEBUG_
    printf("¥nDest:%02x-%02x-%02x-%02x-%02x-%02x src:%02x-%02x-%02x-%02x-%02x-%02x
type:%04x¥n",
    (unsigned)ethhdr->dest.addr[0], (unsigned)ethhdr->dest.addr[1],
(unsigned)ethhdr->dest.addr[2],
    (unsigned)ethhdr->dest.addr[3], (unsigned)ethhdr->dest.addr[4],
(unsigned)ethhdr->dest.addr[5],
```

Copyright © Infineon Technologies AG 2022. All rights reserved.

ptp.c (4/5)

```
    (unsigned)ethhdr->src.addr[0], (unsigned)ethhdr->src.addr[1], (unsigned)ethhdr-
>src.addr[2],
    (unsigned)ethhdr->src.addr[3], (unsigned)ethhdr->src.addr[4], (unsigned)ethhdr-
>src.addr[5],
    (unsigned)htons(ethhdr->type)
    );
#endif//#ifdef _PTP_DEBUG_

    if(pbuf_header(p, -SIZEOF_ETH_HDR)) {
        LWIP_ASSERT("Can't move over header in packet", 0);
        goto free_and_return;
    } else {
        // pass to PTP layer
// identify the PTP header
        ptphdr = (PTP_hdr_t *)p->payload;
        _ptp_sequenceId = htons( ptphdr->sequenceId );

#ifdef _PTP_DEBUG_
printf("messageType           :%s¥n",
PTP_MessageStr[ PTP_HDR_messageType( ptphdr ) ] );
        printf("versionPTP            :%02x¥n", PTP_HDR_versionPTP( ptphdr ) );
        printf("messageLength         :%04x¥n", htons( ptphdr->messageLength ) );
        printf("sequenceId            :%04x¥n", htons( ptphdr->sequenceId ) );
#endif//#ifdef _PTP_DEBUG_

        if(pbuf_header(p, -(s16_t)sizeof(PTP_hdr_t))) {
            LWIP_ASSERT("Can't move over PTP message in packet", 0);
            goto free_and_return;
        } else {
            // identify the PTP message
            ptpMsg = (void *)p->payload;

            if (           ethhdr->dest.addr[0] == PTP_MULTICAST_ADDR_0) {
                if (      (ethhdr->dest.addr[1] == PTP_MULTICAST_ADDR_1)        &&
(ethhdr->dest.addr[2] == PTP_MULTICAST_ADDR_2)) {
                    // For other than PTP multicast peer delay messages 01-1b-19:
```

```
                    switch( PTP_HDR_messageType( ptphdr ) ){
                    case PTP_MSG_Sync:
                        // Get t2
                        while( XMC_ETH_MAC_GetRxTimeStamp( &eth_mac,
(XMC_ETH_MAC_TIME_t *)&_ptp_t2 ) == XMC_ETH_MAC_STATUS_BUSY){
                        }
                        break;
                    case PTP_MSG_Delay_Req:
                        goto free_and_return;
                    case PTP_MSG_Follow_Up:
                        // Recv t1 & Send Delay_Req, Get t3
                        getTimeStampFromPTPMessage( (u8_t *)ptpMsg,
(XMC_ETH_MAC_TIME_t *)&_ptp_t1 );
                        if( ptp_send_Delay_Req( netif ) != ERR_OK){
                        }
                        while( XMC_ETH_MAC_GetTxTimeStamp( &eth_mac,
(XMC_ETH_MAC_TIME_t *)&_ptp_t3 ) == XMC_ETH_MAC_STATUS_BUSY ){
                        }
                        break;
                    case PTP_MSG_Delay_Resp:
                        // Recv t4
                        getTimeStampFromPTPMessage( (u8_t *)ptpMsg,
(XMC_ETH_MAC_TIME_t *)&_ptp_t4 );
                        printf("t1                    : %08x %08x¥n", (unsigned
int)_ptp_t1.seconds, (unsigned int)_ptp_t1.nanoseconds );
                        printf("t2                    : %08x %08x¥n", (unsigned
int)_ptp_t2.seconds, (unsigned int)_ptp_t2.nanoseconds );
                        printf("t3                    : %08x %08x¥n", (unsigned
int)_ptp_t3.seconds, (unsigned int)_ptp_t3.nanoseconds );
                        printf("t4                    : %08x %08x¥n", (unsigned
int)_ptp_t4.seconds, (unsigned int)_ptp_t4.nanoseconds );

                        //  = ((t2 - t1) + (t4 - t3)) / 2
                        calcTimeSub( &t2_1,     &_ptp_t2, &_ptp_t1 );
                        calcTimeSub( &t4_3,     &_ptp_t4, &_ptp_t3 );
                        calcTimeAdd( &t_result, &t2_1,    &t4_3    );
```

Copyright © Infineon Technologies AG 2022. All rights reserved.

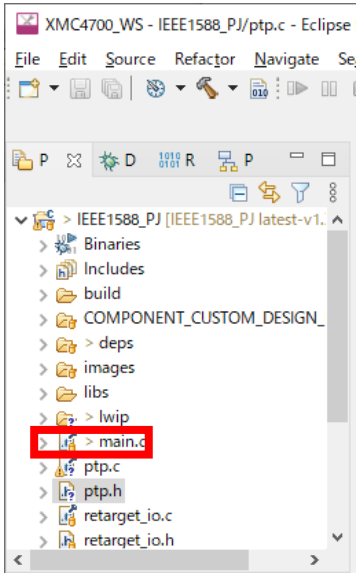
ptp.c (5/5)

```
                        meanPathDelay.seconds        =
(uint32_t)(((int32_t)t_result.seconds) / 2);
                        meanPathDelay.nanoseconds    = t_result.nanoseconds / 2;
                        //  = meanPathDelay - (t2 - t1) (for
XMC_ETH_MAC_UpdatePTPTime)
                        calcTimeSub( &offsetFromMaster, &meanPathDelay, &t2_1    );

                        printf("t2_1                 : %08x%08x¥n", (unsigned
int)t2_1.seconds, (unsigned int)t2_1.nanoseconds );
                        printf("t4_3                 : %08x%08x¥n", (unsigned
int)t4_3.seconds, (unsigned int)t4_3.nanoseconds );
                        printf("meanPathDelay        : %08x%08x¥n", (unsigned
int)meanPathDelay.seconds, (unsigned int)meanPathDelay.nanoseconds );
                        printf("offsetFromMaster     : %08x%08x¥n", (unsigned
int)offsetFromMaster.seconds, (unsigned int)offsetFromMaster.nanoseconds );

                        XMC_ETH_MAC_UpdatePTPTime( &eth_mac, &offsetFromMaster );
                        break;
                    case PTP_MSG_Announce:
                        goto free_and_return;
                    case PTP_MSG_Signaling:
                        goto free_and_return;
                    case PTP_MSG_Management:
                        goto free_and_return;
                    default:
                        goto free_and_return;
                    }
                }else if ((ethhdr->dest.addr[1] == PTP_PDELAY_MULTICAST_ADDR_1) &&
(ethhdr->dest.addr[2] == PTP_PDELAY_MULTICAST_ADDR_2)) {
                    // For PTP multicast peer delay messages 01-80-c2:
                    switch( PTP_HDR_messageType( ptphdr ) ){
                    case PTP_MSG_Pdelay_Req:
                        goto free_and_return;
                    case PTP_MSG_Pdelay_Resp:
                        goto free_and_return;
                    case PTP_MSG_Pdelay_Follow_Up:
                        goto free_and_return;
                    default:
                        goto free_and_return;
                    }
                }
            }
        }
    }

free_and_return:
    pbuf_free(p);
    return ERR_OK;
}
```

Copyright © Infineon Technologies AG 2022. All rights reserved.

main.c修正

1. Project Explorerでmain.cをダブルクリックしてファイルを開き以降に記載した内容に変更し、ファイルを保存します (元のコードは削除してください)



Copyright © Infineon Technologies AG 2022. All rights reserved.

main.c (1/2)

```
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
// IEEE1588 PTP Slave project main
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
#include "cybsp.h"
#include "cy_utils.h"
#include "retarget_io.h"

#include "lwip/netif.h"
#include "lwip/init.h"
#include "netif/etharp.h"
#include "ethernetif.h"
#include "lwip/dhcp.h"
#include "lwip/timeouts.h"

#include "ptp.h"
#include "ip_addr.h"

// Static IP ADDRESS (Please modify according to your environment)
#define IP_ADDR0   192
#define IP_ADDR1   168
#define IP_ADDR2   2
#define IP_ADDR3   10

// NETMASK (Please modify according to your environment)
#define NETMASK_ADDR0   255
#define NETMASK_ADDR1   255
#define NETMASK_ADDR2   255
#define NETMASK_ADDR3   0

// Gateway Address (Please modify according to your environment)
#define GW_ADDR0   192
#define GW_ADDR1   168
#define GW_ADDR2   1
#define GW_ADDR3   1

struct netif xnetif;
```

```
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
// Function declaration
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
//============================================================================
// LWIP initialization
//============================================================================
void LWIP_Init(void){
ip_addr_t  ipaddr;
ip_addr_t  netmask;
ip_addr_t  gw;

    IP4_ADDR(&ipaddr, IP_ADDR0, IP_ADDR1, IP_ADDR2, IP_ADDR3);
    IP4_ADDR(&netmask, NETMASK_ADDR0, NETMASK_ADDR1 , NETMASK_ADDR2,
NETMASK_ADDR3);
    IP4_ADDR(&gw, GW_ADDR0, GW_ADDR1, GW_ADDR2, GW_ADDR3);

    // Initialize lwip
    lwip_init();

    //Add a network interface to the list of lwIP netifs
    netif_add(&xnetif, &ipaddr, &netmask, &gw, NULL, &ethernetif_init,
&ethernet_input);

    //  Registers the default network interface
    netif_set_default(&xnetif);

    // Set Ethernet link flag
    xnetif.flags |= NETIF_FLAG_LINK_UP;

    // When the netif is fully configured this function must be called
    netif_set_up(&xnetif);
}
```

Copyright © Infineon Technologies AG 2022. All rights reserved.

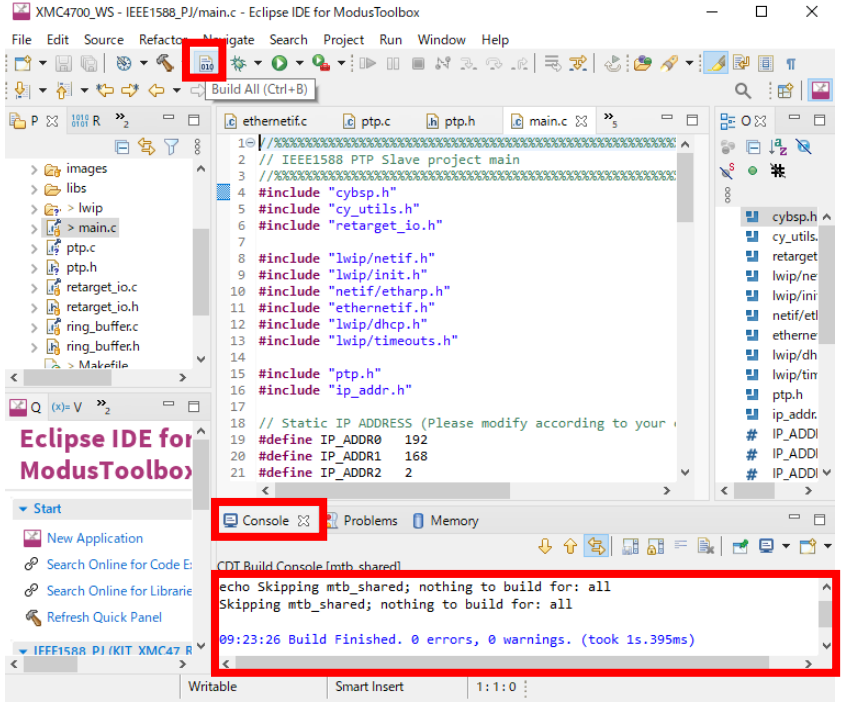
main.c (2/2)

```
//=============================================================================
// main function
//=============================================================================
int main(void){
cy_rslt_t result;

    // Initialize the device and board peripherals
    result = cybsp_init();
    if (result != CY_RSLT_SUCCESS){
        CY_ASSERT(0);
    }

    // Initialize printf retarget
    retarget_io_init();

    SysTick_Config(SystemCoreClock / 1000);
    XMC_SCU_CLOCK_EnableClock(SCU_CLK_CLKCLR_MMCCDI_Msk);//System Control Unit (SCU)

    // Initialize LWIP
    LWIP_Init();

    printf("PTP Slave Ready¥n");

    while(1){
        sys_check_timeouts();
    }
}
```

Copyright © Infineon Technologies AG 2022. All rights reserved.

プロジェクトのビルド

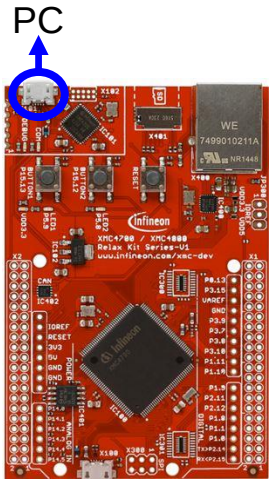
1. Tool PanelのBuild Allアイコンをクリックしてプロジェクトをビルドします。
2. Errorが発生していない事をConsole / MemoryのConsoleタブで確認します。



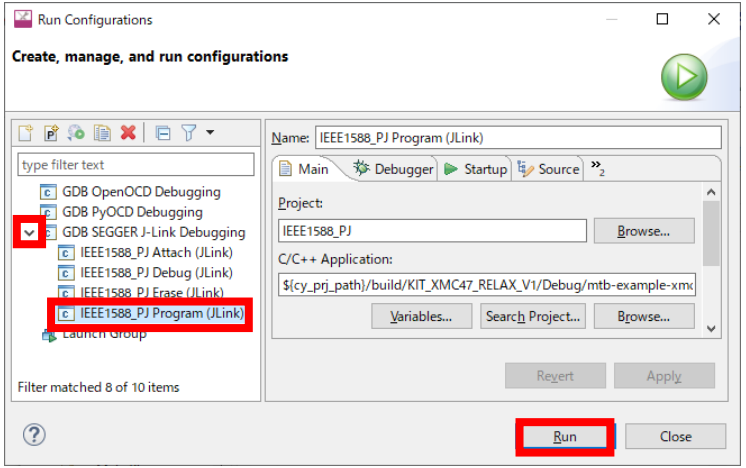
Copyright © Infineon Technologies AG 2022. All rights reserved.

プロジェクトの書き込み及び確認 (1/3)

1. PCとKIT_XMC47_RELAX_V1 をUSBケーブルで接続します。
2. Tool PanelのRunアイコンをクリックします。
3. 初めて書き込みを行う場合、Run Configuration生成画面が表示されるので、GDB SEGGER J-Link Debuggingの左端をクリックし、IEEE1588_PJ Program(JLink)をクリックして選択します。
4. Runをクリックします。



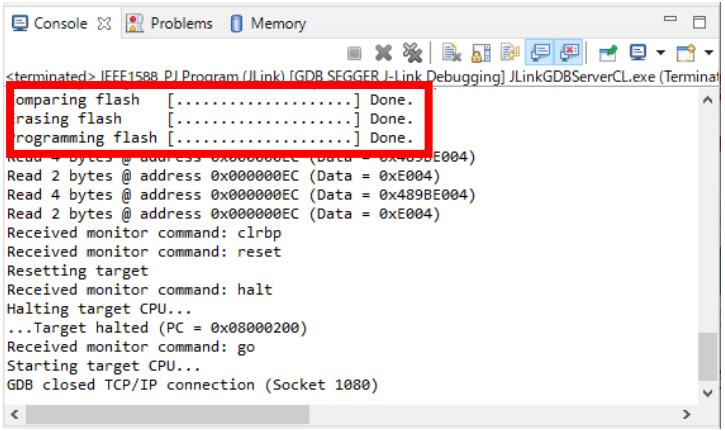




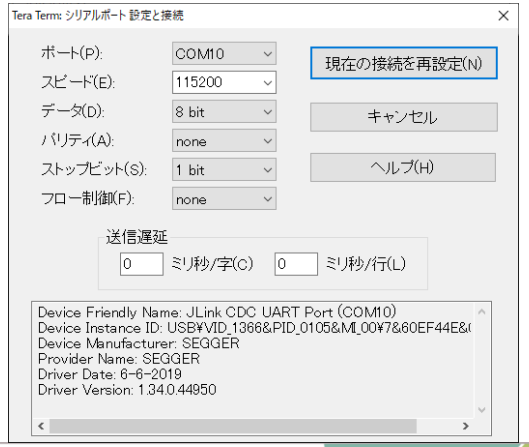
Copyright © Infineon Technologies AG 2022. All rights reserved.

プロジェクトの書き込み及び確認 (2/3)

5. ConsoleタブでProgramming flashが完了したことを確認します。
6. ターミナルソフトでJLink CDC UART Portに接続します。
   Note:環境によりCOMポート番号は変化します。
7. シリアルポートに以下の設定を行います。
   - Speed: 115200
   - Data: 8bit
   - Parity: None
   - Stop bit: 1bit
   - Flow Control: None



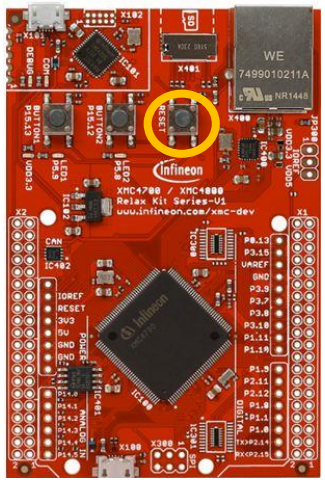


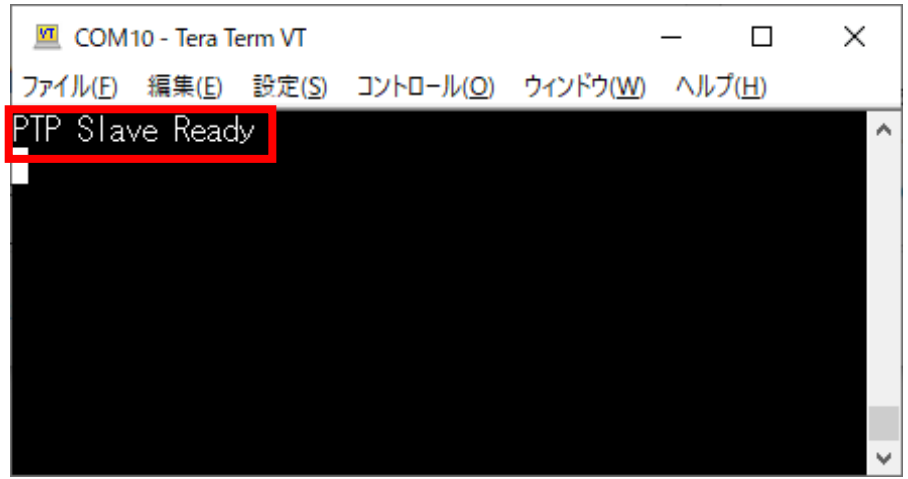


Copyright © Infineon Technologies AG 2022. All rights reserved.

プロジェクトの書き込み及び確認 (3/3)

8. KIT_XMC47_RELAX_V1 のResetボタンを押します。
9. ターミナル画面にPTP Slave Readyの表示がされることを確認します。

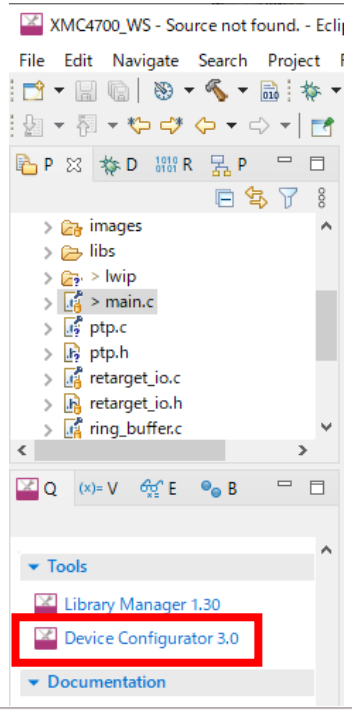


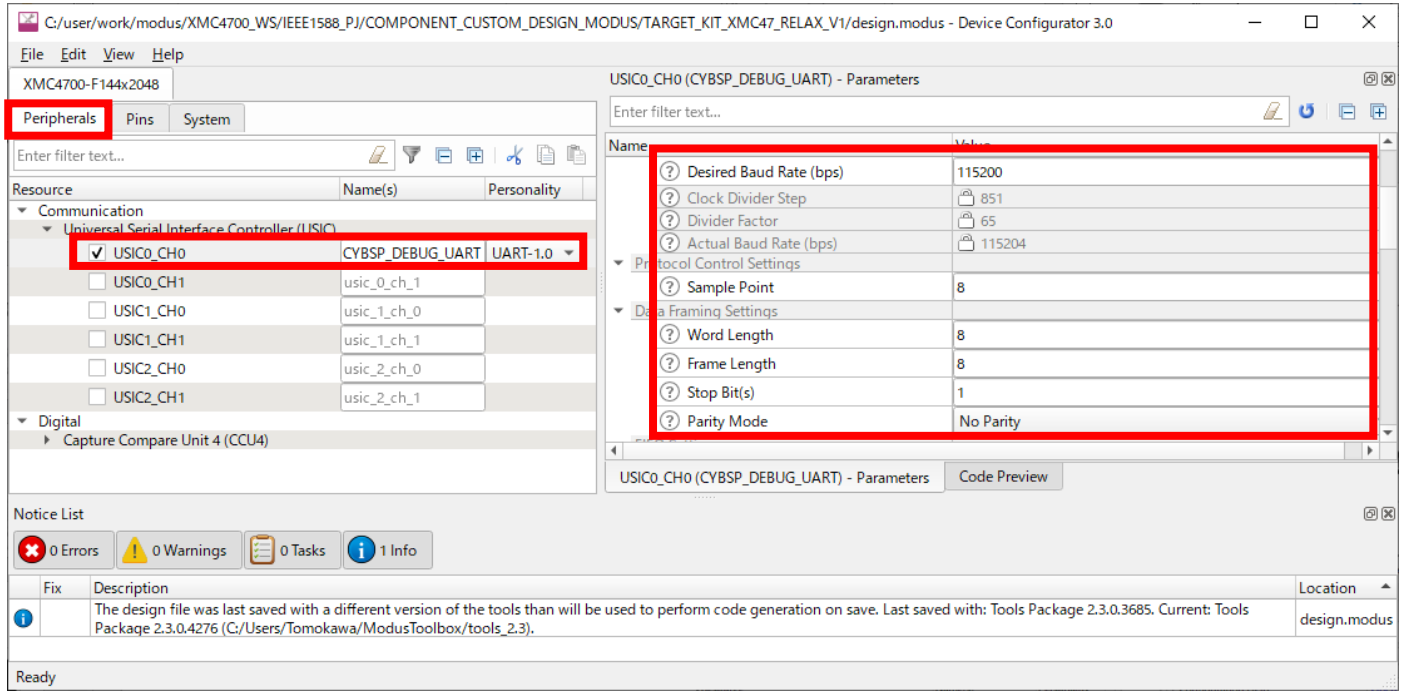


Copyright © Infineon Technologies AG 2022. All rights reserved.

シリアルポート設定について(補足)

› シリアルポートについては、Quick PanelでDevice Configuratorを起動し、USICを選択することで設定を行う事ができます。





Copyright © Infineon Technologies AG 2022. All rights reserved.

PTP Master環境構築

Copyright © Infineon Technologies AG 2022. All rights reserved.

PTP Master環境構築(1/5)

› Raspberry PIをPTP Masterとしてセットアップを行います。
ルーターへの接続は無線LAN(wlan0)を使用し、有線LAN(eth0)をPTP Slaveとの接続に使用します。

1. 公式サイトよりRaspberry PI OSをダウンロードし、SDカードにインストールします。
https://www.raspberrypi.org/software/operating-systems
Note: SDカードフォーマッタ:https://www.sdcard.org/ja/downloads-2/formatter-2/
書き込みユーティリティ:https://www.balena.io/etcher/

2. SDカードのrootディレクトリに下記に示す内容のwpa_supplicant.conf ファイルと、内容は空のsshファイルをコピーします。

```
country=JP
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
Network = {
    ssid="test"
    psk="abcd1234"
}
```

← 接続するアクセスポイントのSSIDとPassword

Note:起動時にシステムにより/etc/wpa_supplicant/wpa_supplicant.confに上書きされます。

Copyright © Infineon Technologies AG 2022. All rights reserved.

PTP Master環境構築(2/5)

3. SDカードをRaspberry PIのスロットに挿入して電源を投入します。

4. 初期アカウント(UserName:pi Password:raspberry)でLoginします。

5. アップデートを実行します。
   - $ sudo apt-get update
   - $ sudo apt-get upgrade
   - $ sudo apt-get dist-upgrade

Copyright © Infineon Technologies AG 2022. All rights reserved.

PTP Master環境構築(3/5)

6. 無線LAN(wlan0)のアドレスを確認します。

   $ ip route show

   表示例:
   default via 192.168.1.1 dev wlan0 proto dhcp src **192.168.1.110** metric 303
   192.168.1.0/24 dev wlan0 proto dhcp scope link src 192.168.1.110 metric 303

   Note:表示例の場合、PC等からアクセスする際は192. 168. 1. 110を指定します。

7. 有線LAN(eth0) のIPアドレスを**192.168.2.1** として、/etc/dhcpcd.conf ファイルを編集して設定します。

   $ sudo vi /etc/dhcpcd.conf

   # A sample configuration for dhcpcd.
   …(omitted)…
   # Example static IP configuration:
   interface eth0
   static ip_address=**192.168.2.1**/24

Copyright © Infineon Technologies AG 2022. All rights reserved.

PTP Master環境構築(4/5)

8. /etc/rc.local ファイルを編集してルーティング設定を行います。
   $ sudo vi /etc/rc.local

```
#!/bin/sh -e
…(omitted)…
if [ "$_IP" ]; then
  printf "My IP address is %s¥n" "$_IP"
fi
echo 1 > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A POSTROUTING -o wlan0 -j MASQUERADE
exit 0
```

9. 以下のコマンドを実行し eth0の起動とNetworkの再起動を行います。
   $ sudo ip link set eth0 up
   $ sudo systemctl restart networking

10. 以下のコマンドを実行し Linuxptp(ptp4l)のインストールを行います。
   $ sudo apt-get install linuxptp

Copyright © Infineon Technologies AG 2022. All rights reserved.

PTP Master環境構築(5/5)

› Linuxptp(ptp4l) をRaspberry PIの起動時に自動起動する場合は以下の設定を行います

11. /usr/lib/systemd/system/ptp4l.serviceファイルを編集します。

$ sudo vi /usr/lib/systemd/system/ptp4l.service

```
[Unit]
Description=Precision Time Protocol (PTP) service
Documentation=man:ptp4l

[Service]
Type=simple
ExecStart=/usr/sbin/ptp4l –f /tec/linuxptp/ptp4l.conf -i eth0 -m -S -2

[Install]
WantedBy=multi-user.target
```

12. 以下のコマンドを実行しLinuxptpの起動を確認します。

$ sudo systemctl daemon-reload

$ sudo systemctl start ptp4l

13. 以下のコマンドを実行し 自動起動を有効化します。

$ sudo systemctl enable ptp4l

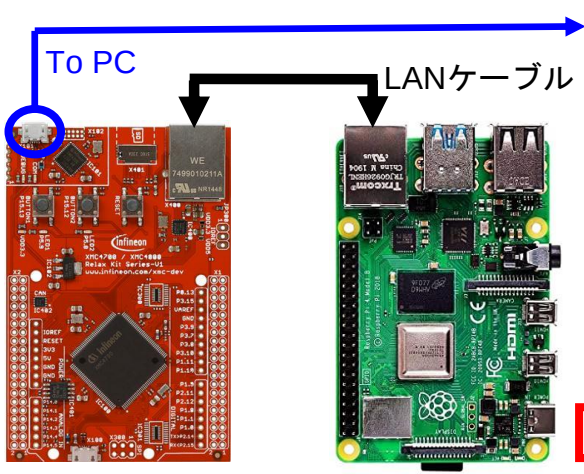
Copyright © Infineon Technologies AG 2022. All rights reserved.

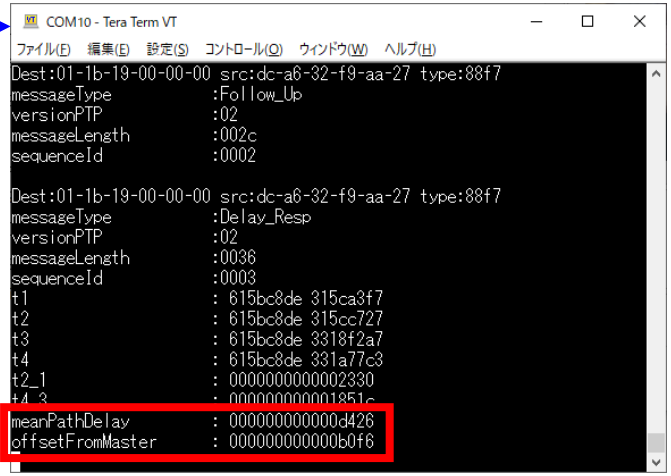
結合テスト

Copyright © Infineon Technologies AG 2022. All rights reserved.

結合テスト

1. Raspberry PIと KIT_XMC47_RELAX_V1 をLANケーブルで接続します。
2. ターミナルソフトでJLink CDC UART Portに接続します。
   Note **:ptp4lを自動起動に設定した場合は3～4は不要です。**
3. Raspberry PIに初期アカウント(UserName:pi Password:raspberry)でLoginします。
4. Raspberry PI側で以下のコマンドを実行します。
   $ sudo ptp4l -2 -A -S -i eth0 -m
5. ターミナルソフトで結果を確認します。





meanPathDelay : 000000000000d426
0xd426 = 54310 [ns] = 54.3 [µs]

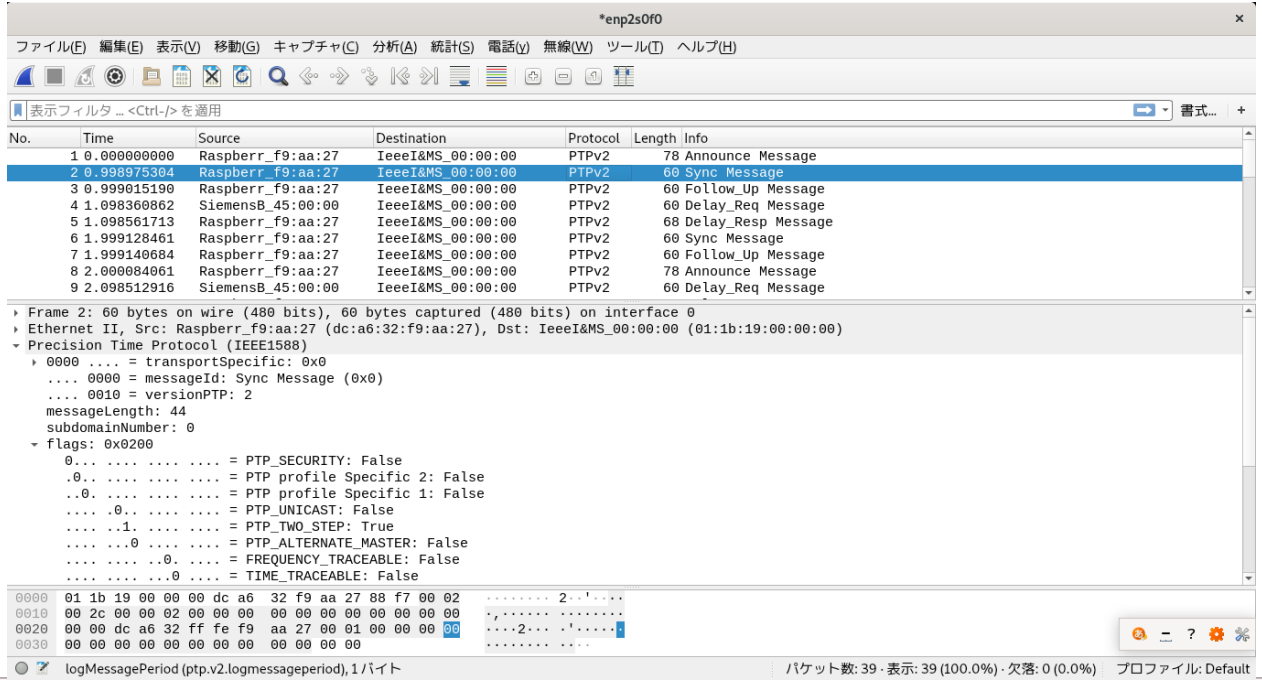
offsetFromMaster : 000000000000b0f6
0xb0f6 = 45302 [ns] = 45.3 [µs]

Note:Raspberry PIのEthernetコントローラにはHardwareTimeStamp機能が無いため誤差が大きくなります

Copyright © Infineon Technologies AG 2022. All rights reserved.

動作時のパケットキャプチャー結果

- Raspberr_f9:aa:27(Raspberry PI)からSync, Follow_Upが送出され、SiemensB_45:00:00(KIT_XMC47_RELAX_V1)がDelay_Reqを送出し、Raspberr_f9:aa:27からDelay_Respが送出されていることが分かります。



Copyright © Infineon Technologies AG 2022. All rights reserved.



Part of your life. Part of tomorrow.

Copyright © Infineon Technologies AG 2022. All rights reserved.