Ethernet スタータキット RX65N

取扱説明書(2)

ルネサス エレクトロニクス社　RX マイコン搭載
HSB シリーズマイコンボード　評価キット

-本書を必ずよく読み、ご理解された上でご利用ください

株式会社北斗電子
REV.1.0.0.0

ー目　次ー

注意事項

本書を必ずよく読み、ご理解された上でご利用ください

【ご利用にあたって】

1. 本製品をご利用になる前には必ず取扱説明書をよく読んで下さい。また、本書は必ず保管し、使用上不明な点がある場合は再読し、よく理解して使用して下さい。
2. 本書は株式会社北斗電子製マイコンボードの使用方法について説明するものであり、ユーザシステムは対象ではありません。
3. 本書及び製品は著作権及び工業所有権によって保護されており、全ての権利は弊社に帰属します。本書の無断複写・複製・転載はできません。
4. 弊社のマイコンボードの仕様は全て使用しているマイコンの仕様に準じております。マイコンの仕様に関しましては製造元にお問い合わせ下さい。弊社製品のデザイン・機能・仕様は性能や安全性の向上を目的に、予告無しに変更することがあります。また価格を変更する場合や本書の図は実物と異なる場合もありますので、御了承下さい。
5. 本製品のご使用にあたっては、十分に評価の上ご使用下さい。
6. 未実装の部品に関してはサポート対象外です。お客様の責任においてご使用下さい。

【限定保証】

1. 弊社は本製品が頒布されているご利用条件に従って製造されたもので、本書に記載された動作を保証致します。
2. 本製品の保証期間は購入戴いた日から1年間です。

【保証規定】

保証期間内でも次のような場合は保証対象外となり有料修理となります

1. 火災・地震・第三者による行為その他の事故により本製品に不具合が生じた場合
2. お客様の故意・過失・誤用・異常な条件でのご利用で本製品に不具合が生じた場合
3. 本製品及び付属品のご利用方法に起因した損害が発生した場合
4. お客様によって本製品及び付属品へ改造・修理がなされた場合

【免責事項】

弊社は特定の目的･用途に関する保証や特許権侵害に対する保証等、本保証条件以外のものは明示・黙示に拘わらず一切の保証は致し兼ねます。また、直接的・間接的損害金もしくは欠陥製品や製品の使用方法に起因する損失金・費用には一切責任を負いません。損害の発生についてあらかじめ知らされていた場合でも保証は致し兼ねます。
ただし、明示的に保証責任または担保責任を負う場合でも、その理由のいかんを問わず、累積的な損害賠償責任は、弊社が受領した対価を上限とします。本製品は「現状」で販売されているものであり、使用に際してはお客様がその結果に一切の責任を負うものとします。弊社は使用または使用不能から生ずる損害に関して一切責任を負いません。
保証は最初の購入者であるお客様ご本人にのみ適用され、お客様が転売された第三者には適用されません。よって転売による第三者またはその為になすお客様からのいかなる請求についても責任を負いません。
本製品を使った二次製品の保証は致し兼ねます。

安全上のご注意

製品を安全にお使いいただくための項目を次のように記載しています。絵表示の意味をよく理解した上でお読み下さい。

表記の意味

警告 取扱を誤った場合、人が死亡または重傷を負う危険が切迫して生じる可能性がある事が想定される

注意 取扱を誤った場合、人が軽傷を負う可能性又は、物的損害のみを引き起こすが可能性がある事が想定される

絵記号の意味

	**一般指示** 使用者に対して指示に基づく行為を強制するものを示します		**一般禁止** 一般的な禁止事項を示します
	**電源プラグを抜く** 使用者に対して電源プラグをコンセントから抜くように指示します		**一般注意** 一般的な注意を示しています

警告

以下の警告に反する操作をされた場合、本製品及びユーザシステムの破壊・発煙・発火の危険があります。マイコン内蔵プログラムを破壊する場合もあります。

1. 本製品及びユーザシステムに電源が入ったままケーブルの抜き差しを行わないでください。
2. 本製品及びユーザシステムに電源が入ったままで、ユーザシステム上に実装されたマイコンまたはIC等の抜き差しを行わないでください。
3. 本製品及びユーザシステムは規定の電圧範囲でご利用ください。
4. 本製品及びユーザシステムは、コネクタのピン番号及びユーザシステム上のマイコンとの接続を確認の上正しく扱ってください。

発煙・異音・異臭にお気付きの際はすぐに使用を中止してください。

電源がある場合は電源を切って、コンセントから電源プラグを抜いてください。そのままご使用すると火災や感電の原因になります。

⚠注意

以下のことをされると故障の原因となる場合があります。

1. 静電気が流れ、部品が破壊される恐れがありますので、ボード製品のコネクタ部分や部品面には直接手を触れないでください。
2. 次の様な場所での使用、保管をしないでください。

   ホコリが多い場所、長時間直射日光があたる場所、不安定な場所、衝撃や振動が加わる場所、落下の可能性がある場所、水分や湿気の多い場所、磁気を発するものの近く
3. 落としたり、衝撃を与えたり、重いものを乗せないでください。
4. 製品の上に水などの液体や、クリップなどの金属を置かないでください。
5. 製品の傍で飲食や喫煙をしないでください。

ボード製品では、裏面にハンダ付けの跡があり、尖っている場合があります。

取り付け、取り外しの際は製品の両端を持ってください。裏面のハンダ付け跡で、誤って手など怪我をする場合があります。

CD メディア、フロッピーディスク付属の製品では、故障に備えてバックアップ(複製)をお取りください。

製品をご使用中にデータなどが消失した場合、データなどの保証は一切致しかねます。

アクセスランプがある製品では、アクセスランプの点灯中に電源を切ったり、パソコンをリセットをしないでください。

製品の故障や、データ消失の原因となります。

本製品は、医療、航空宇宙、原子力、輸送などの人命に関わる機器やシステム及び高度な信頼性を必要とする設備や機器などに用いられる事を目的として、設計及び製造されておりません。

医療、航空宇宙、原子力、輸送などの設備や機器、システムなどに本製品を使用され、本製品の故障により、人身や火災事故、社会的な損害などが生じても、弊社では責任を負いかねます。お客様ご自身にて対策を期されるようご注意ください。

1. マイコン搭載の Ethernet 機能に関して

2. Ethernet 通信の概要

3. ネットワーク接続時の注意点

4. Ethernet フレームを送ってみる

5. ping の応答

上記の内容は、Ethernet スタータキット RX65 取扱説明書(1)で説明しています。

6. UDP 通信に関して

UDP(User Datagram Protocol)は、ネットワークでデータを送るプロトコルですが、送りっぱなしである事が特徴です。

事前に、「これからデータを送るよ」と話しかける事もなく、送った後で「ちゃんと届きましたか？」の確認も行いません。なんか頼りない、きちんと確認を行わないプロトコルに見えますが、まさにその点が UDP の利点であるとも言えます。事前や事後のやり取りがないので、単純かつ高速にデータを送信する事が可能です。

但し、PC に比べリソースが限られる RX65N マイコンで UDP を使って通信を行った場合、「現在の感覚」では高速であるメリットは感じられないと思います。RX65N マイコンで UDP を使うメリットは、「単純である」という事であるかと考えます。

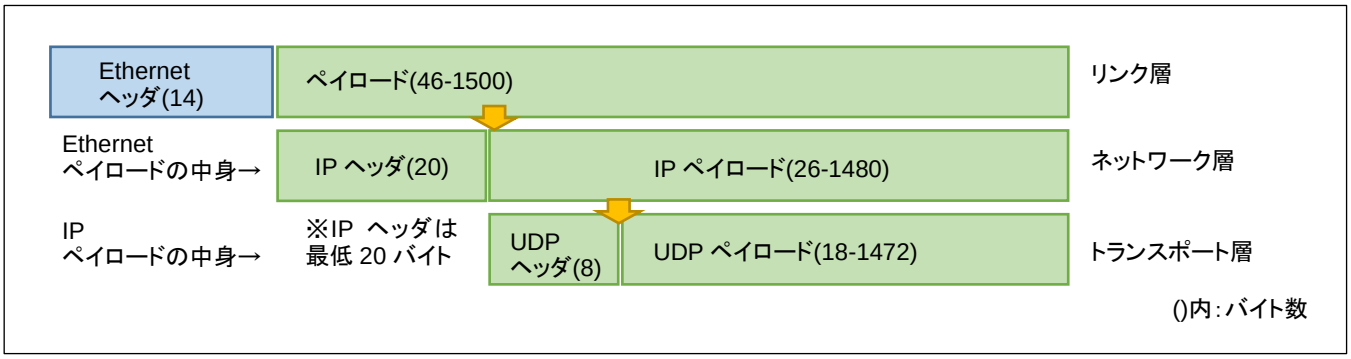
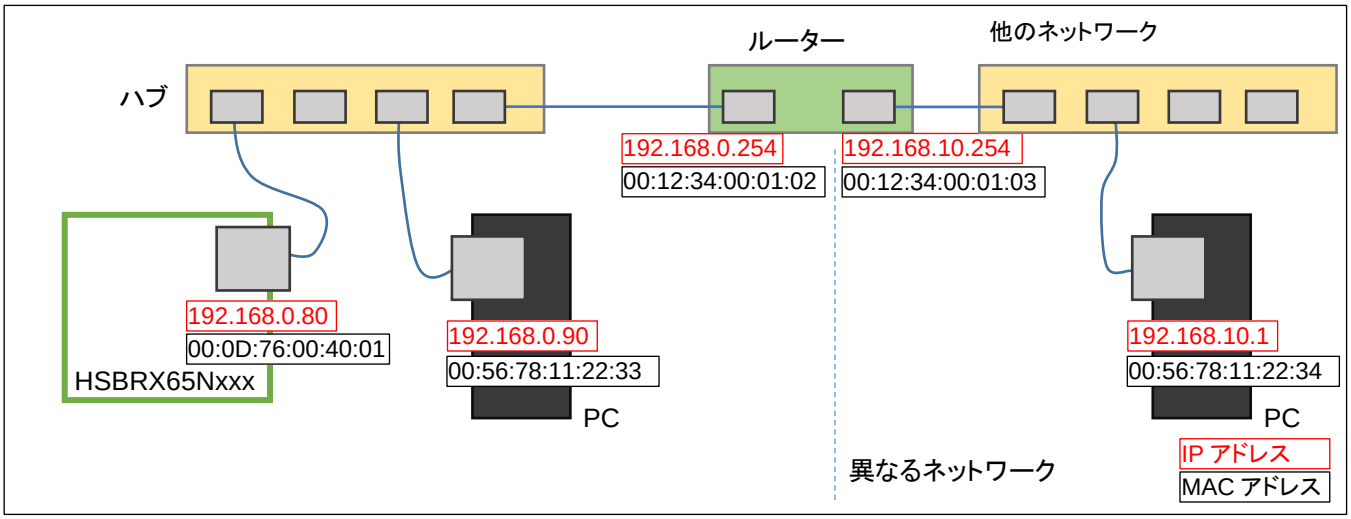


図 6-1 UDP の構造

IP ネットワークを使用した UDP 通信では、Ethernet ヘッダ、IP ヘッダにさらに UDP ヘッダが付加された形で、Ethernet 上にデータが流れます。

Ethernet は階層構造になっており、物理的な通信を担うリンク層、IP アドレスを使って目的値までデータを届けるネットワーク層、アプリケーションと連携してデータを識別するトランスポート層から構成されています。



例えば、192.168.0.80→192.168.0.90 にデータを送る場合は、リンク層 00-0D-76-00-40-01 と 00-56-79-11-22-33 間の通信で完結します。

それに対し、192.168.0.91→192.168.10.1 にデータを送る場合は、リンク層レベルの動作としては、00-56-79-11-22-33 から 00-12-34-00-01-02 にデータを送り、00-12-34-00-01-03 から 00-56-76-11-22-34 にデータが送られます。この場合も、ネットワーク層のレベルで考えると複数のリンク層を経由してデータが流れている事を気にする必要がなく、2 者間の IP アドレスで通信が行われています。

階層化した場合、下位の階層での動作を気にしなくても良い。アプリケーションのレベルでは、同一ネットワーク内か否か（2 者間のリンク層通信で完結するのか、複数のリンク層を経由して通信を行うか）でプログラムを変える必要がないというメリットがあります。

（インターネットの世界では、地球の裏側にデータを送る場合、複数のルータを経由して色々な物理層（メタルケーブルや光ケーブル）を通って通信が行われますが、通信相手の IP アドレスさえ判っていれば、隣にある PC と同じ仕組みで通信ができます。その点が IP ネットワークの強みであると言えます。）

今回使用する UDP 通信では、IP のネットワークを使って、UDP のデータを取り扱います。IP アドレスを使った通信としては、Web の閲覧やファイル共有、リモートデスクトップや、時間の同期など、様々なデータをやり取りしていますが、これらの交通整理を行うのがトランスポート層になるかと思います。

Ethernet ヘッダや IP ヘッダの中身に関しては既出ですので、本章では UDP ヘッダ以下のデータに関して詳しく説明します。

まずはマイコンボードと PC 間で実際にデータのやり取りを行わせてみましょう。

6.1. PC に対してのデータ送信

マイコンボードには、RX65N_UDP プロジェクトの mot ファイルを書き込んでください。
PC 用のアプリケーションとして、PC_APPLI¥UDPTextServer.exe を実行してください。

・マイコンボード側

```
Copyright (C) 2026 HokutoDenshi. All Rights Reserved.
RX65N Ether UDP text client/server sample program.

COMMAND:
   t : send text message set
   p : send text message print
   s : send text message

-setting address-
---
communication target IP address -> 192.168.0.81
this board IP address                -> 192.168.0.80 (MAC address -> 00-0D-76-00-40-01)
---
-setting port-
---
UDP server listen port -> 30000
UDP client dest      port -> 30000
UDP client source port -> 50000
---

address/port setting change -> Please input 'A' within 10 seconds.
>
```

起動すると上記のメッセージが出ます。

項目	規定値	備考
通信相手の IP アドレス	192.168.0.81	A コマンドで変更
マイコンボードの IP アドレス	192.168.0.80	A コマンドで変更
マイコンボードの MAC アドレス	00-0D-76-00-40-01	A コマンドで変更

項目	規定値	備考
データを受信するポート番号	30000	A コマンドで変更
データを送信する送信先ポート番号	30000	A コマンドで変更
データを送信する際に使用するポート番号	50000	A コマンドで変更

デフォルトでは上記設定になっています。通信先の PC の IP アドレスが 192.168.0.97 の場合は、起動後 10 秒以内に A コマンドを入力してください。(10 秒経過すると、自動的に動作開始となります)
(10 秒経過してしまった際は、ボードのリセットボタンを押してください)

・A コマンド入力時

```
>
-- IP/MAC address settings change --

COMMAND:
   1 : communication target IP address
   2 : this board IP address
   3 : this board MAC address
   4 : UDP server listen port (default = 30000)
   5 : UDP client dest port     (default = 30000)
   6 : UDP client source port (default = 50000)
   p : print setting
   e : exit setting
```

上記表示が出ますので、通信先の PC の IP アドレスを設定するために、1 を入力してください。

・1 コマンド入力時

```
communication target IP set(please input 3 letters(0-9) [or 1,2 letters(0-9) + Enter] number x 4)

IP[0](b31-24) >192
IP[1](b23-16) >168
IP[2](b15-8)   >0
IP[3](b7-0)    >97
IP address set to -> 192.168.0.97
```

ping のプログラム同様 IP アドレスの入力画面となりますので、3 桁の数字、もしくは数字＋Enter で値を入力してください。マイコンボード側の IP アドレスを変更する場合は、数字の 2 で変更です。マイコンボードの MAC アドレスや使用するポート番号も変更は可能です。（MAC アドレスは基本的には変更不要かと思います。ポート番号は、詳しくは後述しますが、通信する 2 者間で同じ番号を指定する必要があります。）

IP アドレスの設定が終われば、p コマンドで確認できます。

・p コマンド

```
>-setting address-
---
communication target IP address -> 192.168.0.97
this board IP address                -> 192.168.0.80 (MAC address -> 00-0D-76-00-40-01)
---
-setting port-
---
UDP server listen port -> 30000
UDP client dest     port -> 30000
UDP client source port -> 50000
---
```

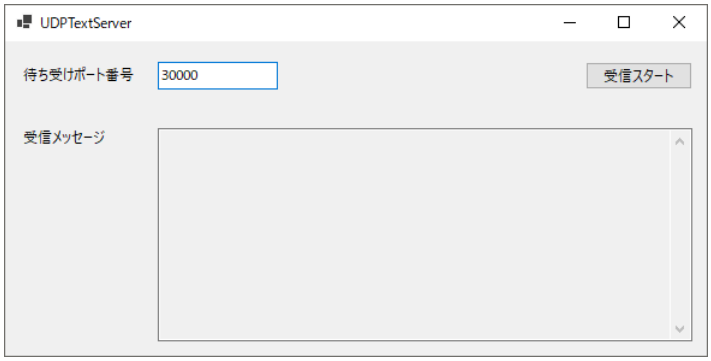
設定項目に間違いがあれば、1~6 の入力で訂正が可能です。設定に問題が無ければ、e コマンド(exit)で入力を終了してください。

・e コマンドの入力

```
network operation start!
Ether0: LINK-UP
```

LINK-UP 表示が出ない場合は、LAN ケーブルを確認してください。

次に、PC 側で UDPTextServer.exe を実行します。



受信スタートを押してください。

マイコンボード側では、s コマンドを入力してください。

・s コマンドの入力

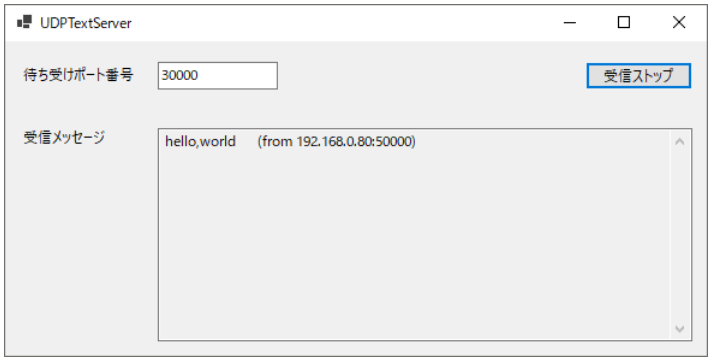
```
>command=s
send text message to 192.168.0.97 port 30000

send target 192.168.0.97 -> MAC address not resolved.
ARP request (192.168.0.97-> MAC address?) send
ARP reply received from 192.168.0.97, MAC address = 70-B5-E8-20-BD-73
UDP text data send to 192.168.0.97:30000
```

初期状態では、192.168.0.97 の MAC アドアレスが判らないので、ARP リクエストで MAC アドレスの問い合わせを行います。この辺りは、ping の動作と同じです。

続いて、UDP でデータを送信します。

（ping の場合は SW 押下 2 回目に ping リクエストを送る仕様でしたが、本プログラムでは s コマンド入力時 ARP リクエストに続いて UDP データ送信を行います。）

PC 側では、



hello, world (from XXX.XXX.XXX.XXX:50000)　XXX は IP アドレス
と表示されれば、マイコンボードから送った UDP データを PC で受信しています。

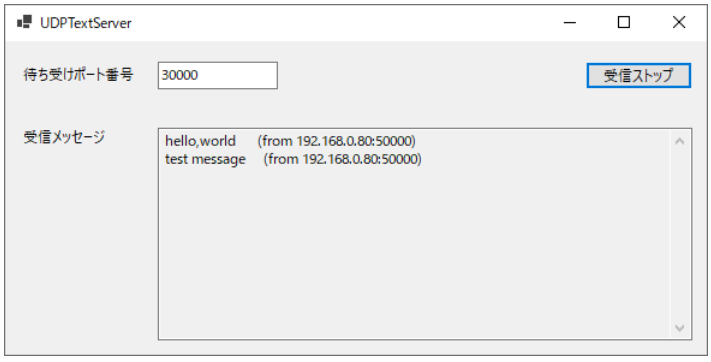
送信するメッセージは、

・t コマンドの入力

```
>command=t
send text message set (please input 16 letters) > "test message"--end
```

t コマンドを入力し、メッセージ（この場合は test message）を入力。入力の終了は、ESC か CNTL-C です。メッセージは最大 16 文字までです。（16 文字というのは、本プログラムで設定しているだけで、プログラムを変更すれば最大 1472 バイトまで送れます。（なお、複数の Ethernet フレームに分割して送信する場合は、1472 バイトの制約はありません。）

メッセージ設定後、s コマンドを入力すると、PC 側では

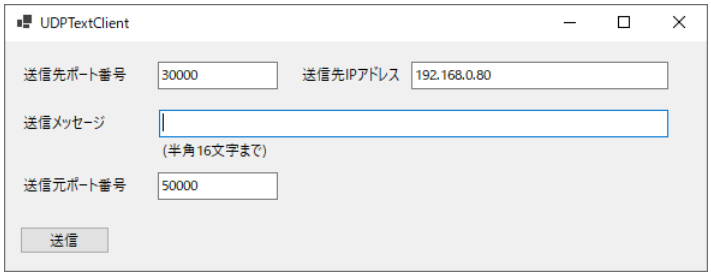


設定後のメッセージを受信します。アプリケーションは「受信ストップ」で停止してください。

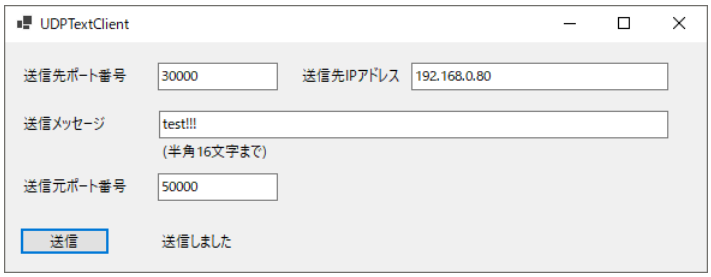
マイコンボード側のプログラムですが、SW3 を押すと通信相手（ここでは、192.168.0.97）に対して、ping リクエストを送信します。SW3 は ping、s コマンドで UDP データ送信です。

6.2. PC からのデータ受信

PC_APPLI¥UDPTextClient.exe を起動してください。



上記の様な画面となりますので、送信メッセージに適当な文字列を設定して、送信ボタンを押してください。



（マイコンボードの IP アドレスを、192.168.0.80 から変えている場合は、送信先 IP アドレスの設定を変えてください。）

・マイコンボード側の表示

```
UDP text data received -> "test!!!              " from 192.168.0.97:50000
ARP reply send (this board MAC address = 00-0D-76-00-40-01) to 70-B5-E8-20-BD-73
```

上の UDP text data received の行が受信した UDP データです。
（PC のアプリで日本語のメッセージを入力した場合、送受信自体は通りますが表示は文字化けします）

ARP reply send のメッセージは、PC から MAC アドレス問い合わせのために、ARP リクエストが来た事に対する応答した旨のメッセージです。
（最初にマイコンボード→PC にメッセージを送っているので、PC からマイコンボードに送信する時点では MAC アドレスの解決が済んでいるので、ARP reply の前に UDP データを受信しています。）
（PC 側でマイコンボードの MAC アドレス解決が済んでいない場合は、ARP reply の後で UDP 受信となります。）

6.3. 2 台のマイコンボード間のデータ送信

2 台のマイコンボード間でデータ送信を行う場合は、書き込むプログラムは同じもので問題ありません。一方のボードの DIP-SW(3)を ON にして起動してください。(もう一方は OFF)

〇1 台目

起動後 10 秒待つ。

```
Copyright (C) 2026 HokutoDenshi. All Rights Reserved.
RX65N Ether UDP text client/server sample program.

COMMAND:
   t : send text message set
   p : send text message print
   s : send text message

-setting address-                                          送信先は
---                                                        192.168.0.81
communication target IP address -> 192.168.0.81
this board IP address           -> 192.168.0.80 (MAC address -> 00-0D-76-00-40-01)
---
-setting port-
---
UDP server listen port -> 30000
UDP client dest      port -> 30000
UDP client source port -> 50000
---

address/port setting change -> Please input 'A' within 10 seconds.
>.....
network operation start!
Ether0: LINK-UP
```

〇2 台目

```
(前略)
-setting address-                                          送信先は
---                                                        192.168.0.80
communication target IP address -> 192.168.0.80
this board IP address           -> 192.168.0.81 (MAC address -> 00-0D-76-00-40-02)
---
(後略)
```

1 台目と 2 台目で、相互に通信が可能な様に設定されます。なお、3 台以上のマイコンボードを接続する場合は、起動後 10 秒以内に A コマンドを入力して重複しない様に適宜 IP アドレス等を設定してください。

・1 台目から s コマンドを入力

```
>command=s
send text message to 192.168.0.81 port 30000

send target 192.168.0.81 -> MAC address not resolved.
ARP request (192.168.0.81-> MAC address?) send
ARP reply received from 192.168.0.81, MAC address = 00-0D-76-00-40-02
UDP text data send to 192.168.0.81:30000
```

・2 台目の画面表示

```
ARP reply send (this board MAC address = 00-0D-76-00-40-02) to 00-0D-76-00-40-01
UDP text data received -> "hello,world          " from 192.168.0.80:50000
```

・2 台目でメッセージ変更及び s コマンドで送信

```
>command=t
send text message set (please input 16 letters) > "2nd board"--end

>command=s
send text message to 192.168.0.80 port 30000

send target 192.168.0.80 -> MAC address not resolved.
ARP request (192.168.0.80-> MAC address?) send
ARP reply received from 192.168.0.80, MAC address = 00-0D-76-00-40-01
UDP text data send to 192.168.0.80:30000
```

・1 台目で受信

```
ARP reply send (this board MAC address = 00-0D-76-00-40-01) to 00-0D-76-00-40-02
UDP text data received -> "2nd board            " from 192.168.0.81:50000
```

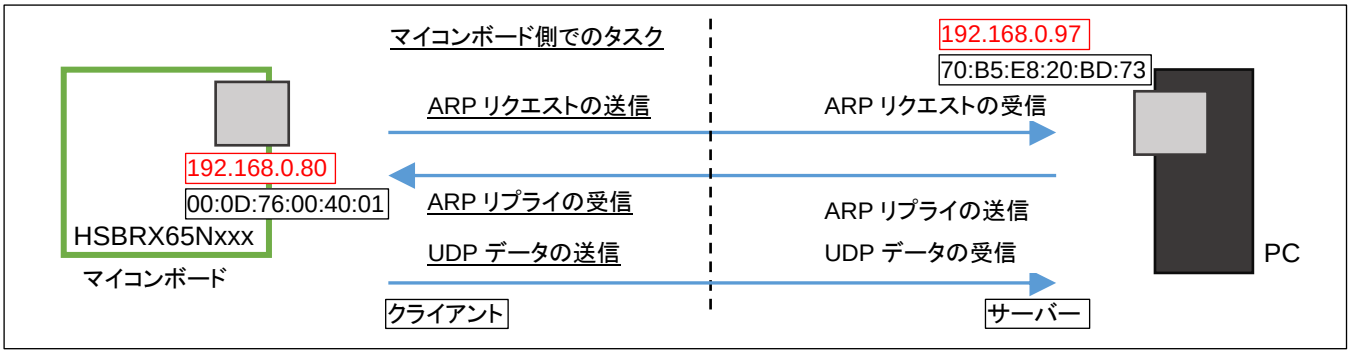
上記の様に相互で、メッセージのやり取りができます。

6.4. UDP パケットの中身

6.4.1. マイコンボードから PC に対してデータ送信

192.168.0.80 のマイコンボードから、192.168.0.97 の PC にデータ送信した場合のパケットは以下の様になります。

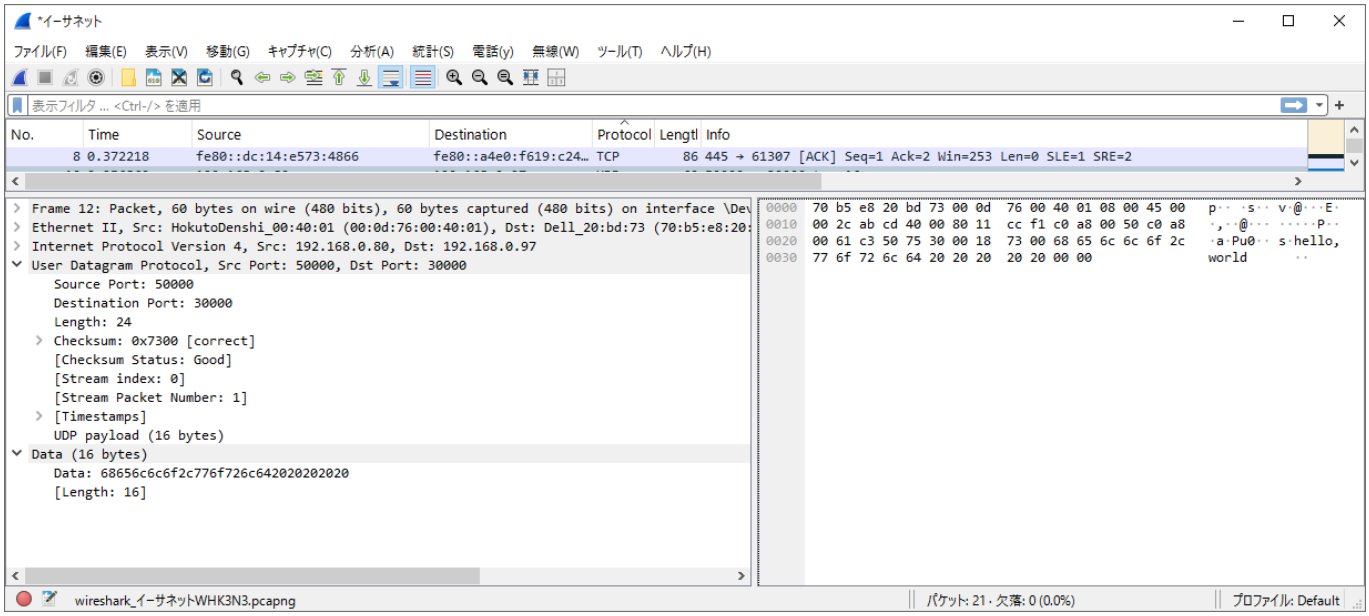
・マイコンボードから PC に対してのデータ送信



マイコンボードは、起動後は 192.168.0.97 の MAC アドレスは判りませんので、ARP リクエストで MAC アドレスの情報を取得します。この部分は、5 章で説明したマイコンボードから PC に対して ping を送る場合と同様です。（ARP のやり取りは、5 章で説明しているのでここでは説明しません。）

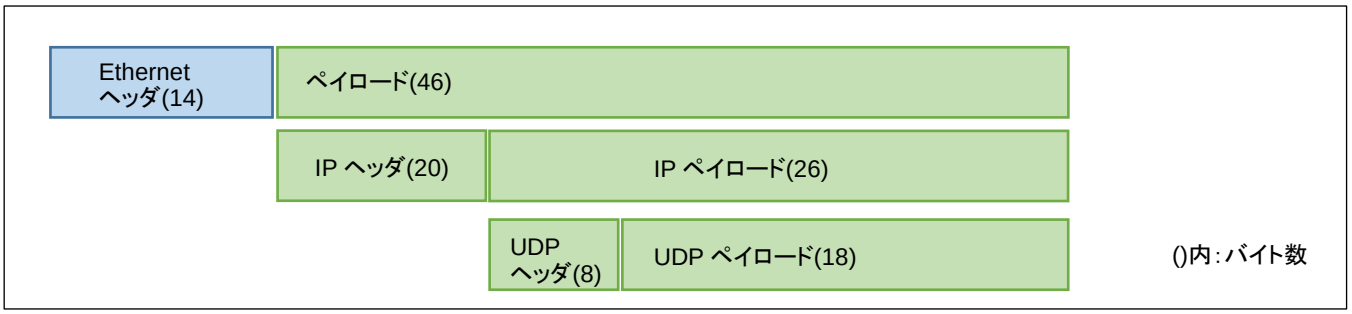
本章で説明するのは、UDP データの送信の部分です。

ここで、データを送る側を「クライアント」。データを受け取る側を「サーバー」と呼ぶ事とします。



Wireshark でパケットダンプすると上記の様になります。

－UDP データの構造－



Ethernet フレームとしては 60 バイトのデータ。Ethernet フレームのペイロードは、IP ヘッダ＋IP ペイロードで構成。IP のペイロードは、さらに UDP のヘッダ＋UDP ペイロードで構成される形です。

－通信パケット例－

・Ethernet ヘッダ(14)

70	B5	E8	20	BD	73	00	0D	76	00	40	01	08	00
宛先 MAC アドレス						送信元 MAC アドレス						Ethernet type (IPv4)	

・IP ヘッダ(20)

45	00	00	2C	AB	CD	40	00	80	11	CC	F1	C0	A8	00	50
IPv4 ヘッダ 20B	Service type	Total Length 44 バイト		ID (*1)		Fragment (*2)		TTL	Protocol UDP (*3)	Checksum		送信元 IP アドレス (192.168.0.80)			

C0	A8	00	61
送信先 IP アドレス (192.168.0.97)			

・UDP ヘッダ(8)

C3	50	75	30	00	18	73	00
Source port (50000)		Destination Port (30000)		length (24 バイト)		Checksum	

・UDP データ(16)

68	65	6C	6C	6F	2C	77	6F	72	6C	64	20	20	20	20	20
'h'	'e'	'l'	'l'	'o'	','	'w'	'o'	'r'	'l'	'd'	' '	' '	' '	' '	' '

・パディング(2)　　…Ethernet フレーム最低 60 バイトのためパディングデータ追加

00	00

(*1)IP ヘッダ内の ID は、「本プログラムでは 0xABCD に固定」しています

※ID はパケットが分割された場合、どのパケットを 1 まとまりとするかという情報なので、本来は毎回変えるべきものです（このプラグラムでは固定値としています）

(*2)IP ヘッダの Flagment フィールドは

0b010 : 3bit, Flags=0x2（分割禁止）

0b0 0000 0000 0000: 13bit, Flagment Offset = 0 （分割送信された場合の分割位置）としています。

(*3)IP ヘッダ内のプロトコルフィールドは、UDP の場合 17(=0x11)となります。

〇UDP ヘッダ（8 バイト）

フィールド	bit 数	内容	データ例
Source Port	16	送信元ポート番号	C3 50
Distination Port	16	送信先ポート番号	75 30
Length	16	UDP セグメントの長さ（ヘッダ＋データ）[バイト]	00 18
Checksum	16	チェックサム	73 00

送信元ポート番号と送信先ポート番号ですが、UDP（や、7 章で説明する TCP）では IP アドレスに加え、ポート番号（後述）でやり取りされるデータの交通整理を行います。

〇UDP データ（16 バイト） ※本プログラムでは 16 バイトとしています（最大 1472 バイト）
→任意のデータ

UDP データの部分は任意のデータを埋め込む事が可能です。本プログラムでは、この部分にメッセージを埋め込んで送信しています。

―UDP のチェックサムの計算方法―

UDP ヘッダ内にはチェックサムがありますが、UDP チェックサムの計算範囲は

・疑似ヘッダ
・UDP ヘッダ
・UDP データ

となります。ここで、UDP ヘッダと UDP データに関しては上記に記載がありますが、「疑似ヘッダ」というのが、新しく登場してきます。

〇疑似ヘッダ（12 バイト）

フィールド	bit 数	内容	データ例
Source IP Address	32	送信元 IP アドレス	C0 A8 00 50
Distination IP Address	32	送信先 IP アドレス	C0 A8 00 61
Zero padding	8	パディング(0x00)	00
Protocol	8	プロトコル（UDP の場合は 0x11）	11
Length	16	UDP セグメントの長さ（ヘッダ＋データ）[バイト]	00 18

疑似ヘッダは、IP アドレスとプロトコルと UDP セグメント（UDP のヘッダと UDP のデータを合わせた長さ）の情報から構成されます。これら、疑似ヘッダ＋UDP ヘッダ＋UDP データで、5 章で説明した 1 の補数和で計算する形です。

・疑似ヘッダ

C0A8	+	0050	=	C0F8
C0F8	+	C0A8	=	181A0
81A1	+	0061	=	8202
8202	+	0011	=	8213
8213	+	0018	=	822B

→81A1（bit16 を 0 にして、1 を加算する）

・UDP ヘッダ

822B	+	C350	=	1457B	→475C
457C	+	7530	=	BAAC	
BAAC	+	0018	=	BAC4	
BAC4	+	0000	=	BAC4	チェックサムは 0x0000 で計算

・UDP データ

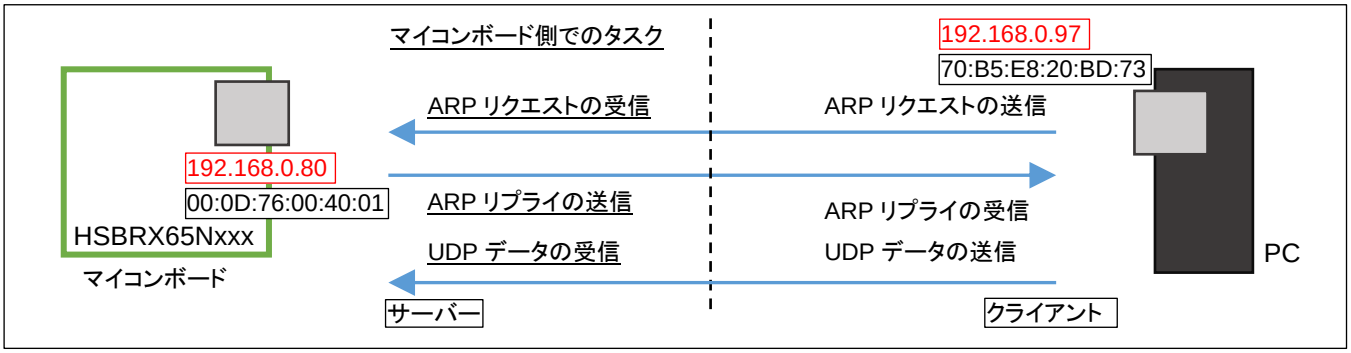
BAC4	+	6865	=	12329	→232A
232A	+	6C6C	=	8F96	
8F96	+	6F2C	=	FEC2	
FEC2	+	776F	=	17631	→7632
7632	+	726C	=	E89E	
E89E	+	6420	=	14CBE	→4CBF
4CBF	+	2020	=	6CDF	
6CDF	+	2020	=	8CFF	
~8CFF			=	7300	

上記の様な計算で、UDP ヘッダ内の Checksum を計算可能です。

6.4.2. PC からマイコンボードに対してデータ送信

192.168.0.97 の PC から、192.168.0.80 のマイコンボードにデータ送信した場合のパケットは以下の様になります。

・マイコンボードから PC に対してのデータ送信



ー通信パケット例ー

・Ethernet ヘッダ(14)

00	0D	76	00	40	01	70	B5	E8	20	BD	73	08	00
宛先 MAC アドレス						送信元 MAC アドレス						Ethernet type (IPv4)	

・IP ヘッダ(20)

45	00	00	2C	E7	20	00	00	80	11	00	00	C0	A8	00	61
IPv4 ヘッダ 20B	Service type	Total Length 44 バイト		ID (*1)		Fragment (*2)		TTL	Protocol UDP	Checksum (*3)		送信元 IP アドレス (192.168.0.97)			

C0	A8	00	50
送信先 IP アドレス (192.168.0.80)			

・UDP ヘッダ(8)

C3	50	75	30	00	18	82	2B
Source port (50000)		Destination Port (30000)		length (24 バイト)		Checksum	

・UDP データ(16)

74	65	73	74	21	21	21	20	20	20	20	20	20	20	20	20
't'	'e'	's'	't'	'!'	'!'	'!'	' '	' '	' '	' '	' '	' '	' '	' '	' '

・Wireshark でモニタした場合パディングデータなし(*4)

(*1)IP ヘッダ内の ID は、PC の場合はランダムに決定されます

(*2)IP ヘッダの Flagment フィールドは

0b000 : 3bit, Flags=0x0（分割 OK）

0b0 0000 0000 0000: 13bit, Flagment Offset = 0 （分割送信された場合の分割位置）

となる様です。（特にプログラムでは指定していないので、デフォルトがそうなっているものと思われます。）

(*3)(*4)チェックサムとパディングデータに関しては、ネットワークインタフェース側で付加（詳しくは 5.5 節で説明しています）

〇UDP ヘッダ（8 バイト）

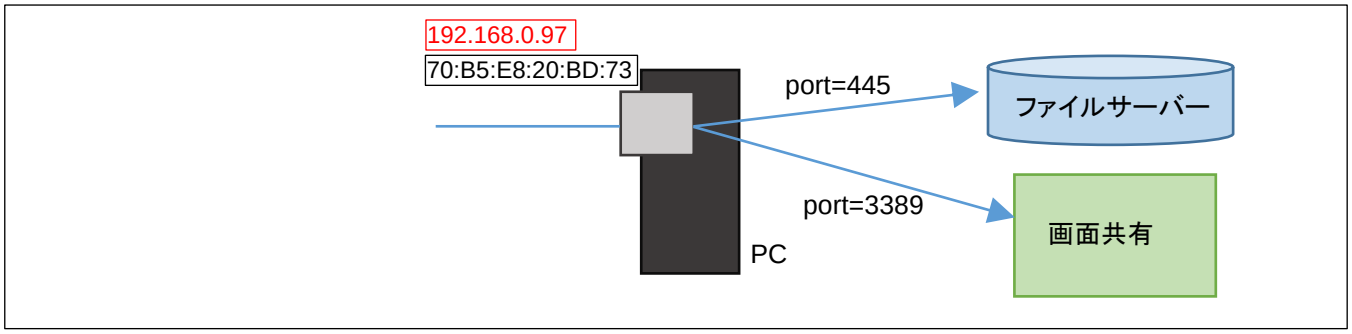
フィールド	bit 数	内容	データ例
Source Port	16	送信元ポート番号	C3 50
Distination Port	16	送信先ポート番号	75 30
Length	16	UDP セグメントの長さ（ヘッダ＋データ）[バイト]	00 18
Checksum	16	チェックサム	82 2B

〇UDP データ（16 バイト）

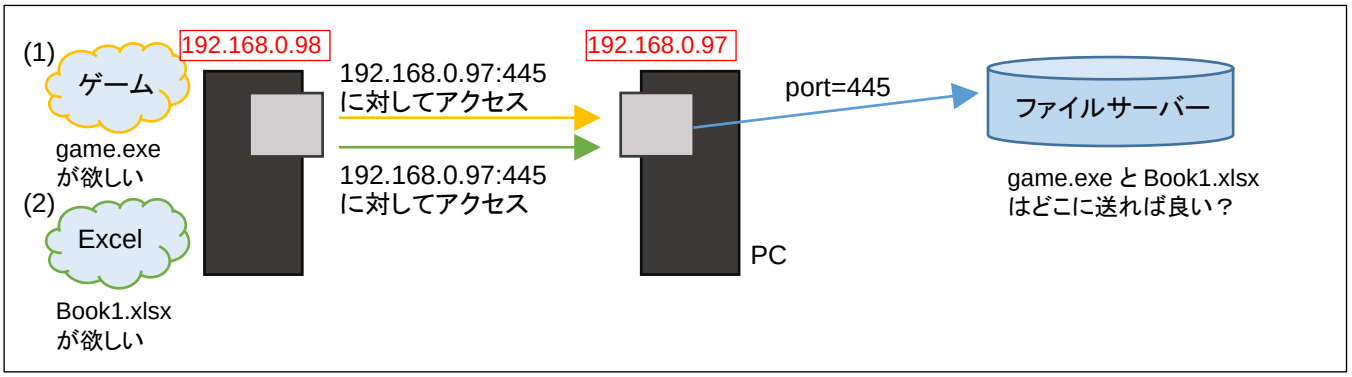
→任意のデータ

6.4.3. ポート番号に関して

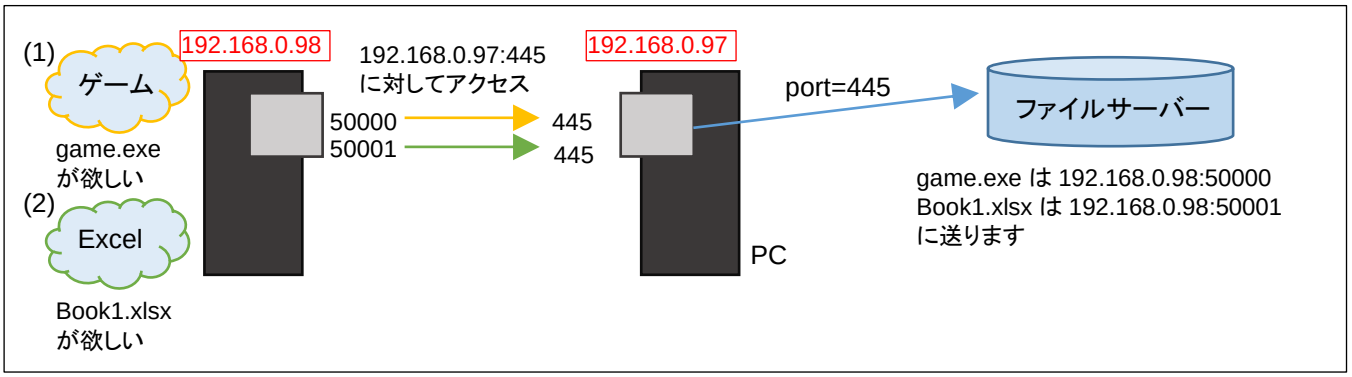
UDP の通信では、IP アドレスに加え、ポート番号を使って送受信するデータの交通整理を行います。



例えば、外部から 445 番のポートにアクセスが来た場合はファイルの共有を行うプログラムに通信データを渡す。3389 番のポートにアクセスが来た場合は画面共有の情報をやり取りするプログラムにデータを渡す、といった具合です。ポート番号によってどのプログラムに取り次ぐかを決めます。



また、ある IP アドレス(192.168.0.98)の機器から、とあるポート(445)に複数のアクセスが来た場合、返送先が判らなくならない様に、送り元にもポート番号が付くようになっています。



ポート番号に関しては、0~1023 の番号をウェルノウン・ポート番号(well-known port number)といい、一般的にはサーバプログラムが使用します。

なお、IP アドレスとポート番号は、192.168.0.97:445 の様にコロン(:)で区切って表記するのが一般的です。

○一般的なウェルノウン・ポート番号

ポート番号	プロトコル名	アプリケーション
20(TCP) 21(TCP)	FTP (File Transfer Protocol)	ファイル転送
22(TCP)	ssh (secure shell)	セキュアシェル（コンピュータの遠隔操作）
25(TCP)	SMTP (Simple Mail Transfer Protocol)	メール送受信
53(UDP/TCP)	DNS (Domain Name System)	ホスト名と IP アドレスの対応の通信
80(TCP)	HTTP (Hyper Text Ttansfer Protocol)	Web ページ

※TCP は 7 章で説明します（世の中には、UDP よりも TCP を使用するアプリケーションが多く存在します）

ウェルノウン・ポートは、アプリケーションの種類により、基本的には固定されています。

マイコンボードでプログラムを動かす場合は関係ありませんが、PC でプログラムを動かす場合は、1023 番以下のポートを使う場合は管理者権限が必要であったりして、ユーザ作成のサーバーアプリケーションでは 1024 番以上の番号を使用するのが一般的です。

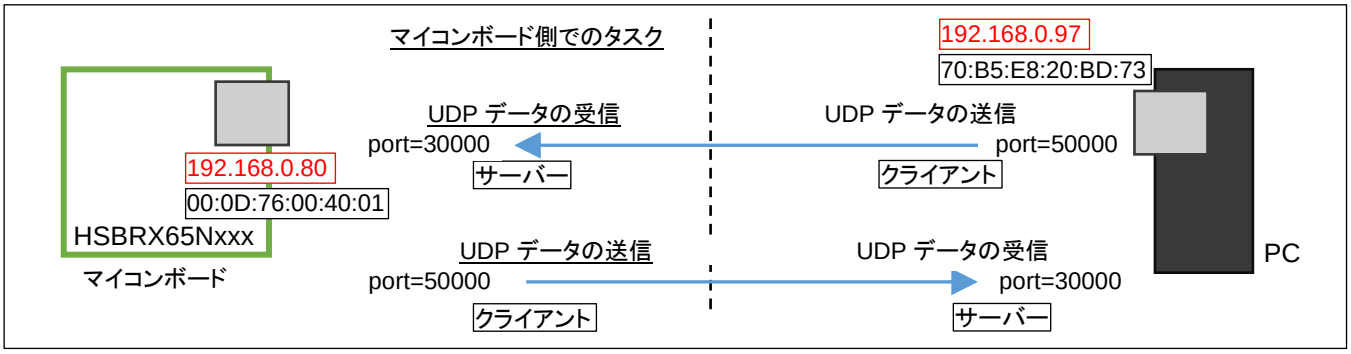
また、サーバと通信するクライアントアプリケーションで使用されるポート番号を、エフェメラル・ポート番号(ephemeral port number)と呼ぶことがあります。エフェメラルは短命という意味で、アプリケーションの起動毎にランダムな番号が使われて、アプリケーションの終了と共に使用されたポート番号は解放されるのが一般的です。

エフェメラル・ポート番号としては、49152~65535 などが使用されます（OS によっても異なります）。

マイコンボード向けのプロジェクト、RX65N_UDP 及び PC アプリケーションの UDPTextServer.exe, UDPTextClient.exe では、デフォルトのポート番号を

ポート番号	用途	備考
30000	サーバの待ち受け	待ち受けポートは通常固定
50000	送信元ポート	送信元ポートは一般的にはランダムに決める （本章のプログラムでは固定としている）

に固定しています。（RX65N_UDP では、起動後 A コマンドで変更可能。PC アプリケーションでは、アプリケーションのテキストボックスに指定する事で変更可能。）



デフォルトの使用ポートは、待ち受け 30000、出力側 50000 から宛先 30000 に対して送信の設定です。（待ち受けと宛先ポートが合っていれば良いですので、出力側のポート番号を変更しても通信は通ります。）

6.5. PC から送信した MagicPacket に関して

4 章では、PC から送信した MagicPacket がマイコンボードから送信した MagicPacket とデータ内容が異なるという話をしましたが、ここでもう一度 PC から送信した MagicPacket のデータを見てみます。

－PC から送信した MagicPacket の例－

・Ethernet ヘッダ(14)

FF	FF	FF	FF	FF	FF	00	B5	E8	20	BD	73	08	00
宛先 MAC アドレス						送信元 MAC アドレス						Ethernet type (IPv4)	

・IP ヘッダ(20)

45	00	00	82	79	29	00	00	80	11	00	00	C0	A8	00	61
IPv4 ヘッダ 20B	Service type	Total Length 130 バイト		ID		Fragment		TTL	Protocol UDP	Checksum		送信元 IP アドレス (192.168.0.97)			

C0	A8	00	FF
送信先 IP アドレス (192.168.0.255)			

・UDP ヘッダ(8)

00	09	00	09	00	6E	1B	64
Source port (9)		Destination Port (9)		length (110 バイト)		Checksum	

・UDP データ(102)

FF	FF	FF	FF	FF	FF	00	0D	76	00	40	01	00	0D	76	00

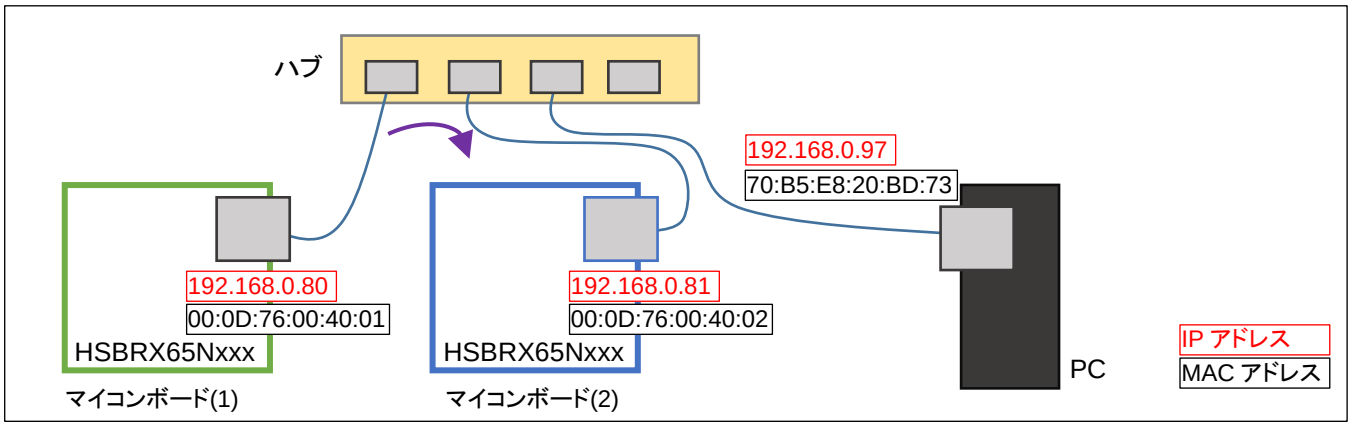
40	01	00	0D	76	00	40	01	00	0D	76	00	40	01	...	...

0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF に続いて 0x00, 0x0D, 0x76, 0x00, 0x40,0x01 を合計 16 回

このデータは、6 章で説明している UDP データそのものです。
（送信先 IP アドレスは、192.168.0.255（ブロードキャストアドレス）になっており、特定のホストではなくネットワーク全体に送る送り先となっています。）

実は本章で登場した、UDP パケットを送るプログラム(UDPTextClient.exe)と、MagicPacket の送信プログラム(MagicPacketSend.exe)。UDP パケットを受信するプログラム(UDPTextServer.exe)と、MagicPacket の受信プログラム(MagicPacketReceive.exe)は、取り扱うデータがテキストなのか MAC アドレスなのかが異なるだけで、ほとんど同じプログラムとなっています。

6.6. 2 台のマイコンボード間のデータ送信を Wireshark でモニタした場合



ここで、192.168.0.80 から 192.168.0.81 に対して UDP データを送信し、そのデータを PC(192.168.0.97)で Wireshark を使って観測してみます。

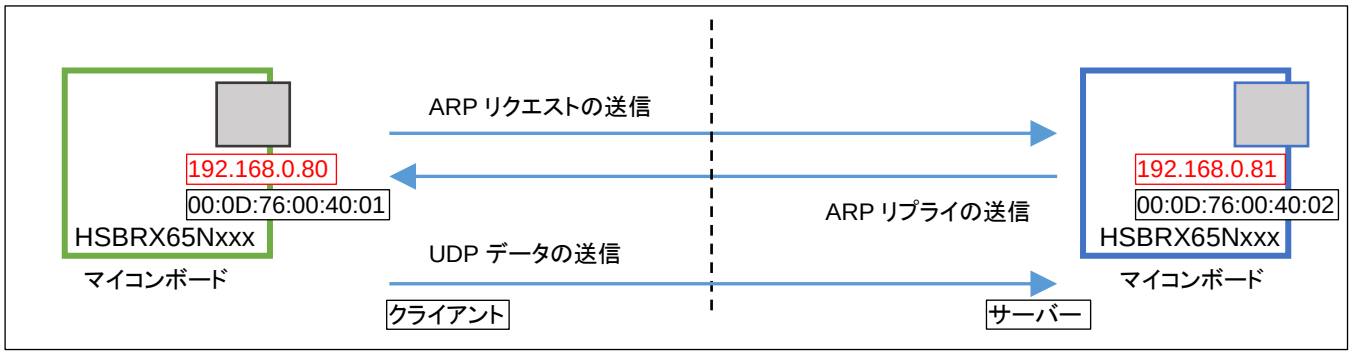
・1 台目から s コマンドを入力

```
>command=s
send text message to 192.168.0.81 port 30000

send target 192.168.0.81 -> MAC address not resolved.
ARP request (192.168.0.81-> MAC address?) send
ARP reply received from 192.168.0.81, MAC address = 00-0D-76-00-40-02
UDP text data send to 192.168.0.81:30000
```

・2 台目の画面表示

```
ARP reply send (this board MAC address = 00-0D-76-00-40-02) to 00-0D-76-00-40-01
UDP text data received -> "hello,world          " from 192.168.0.80:50000
```

マイコンボードに接続した端末には、送受信ができており、



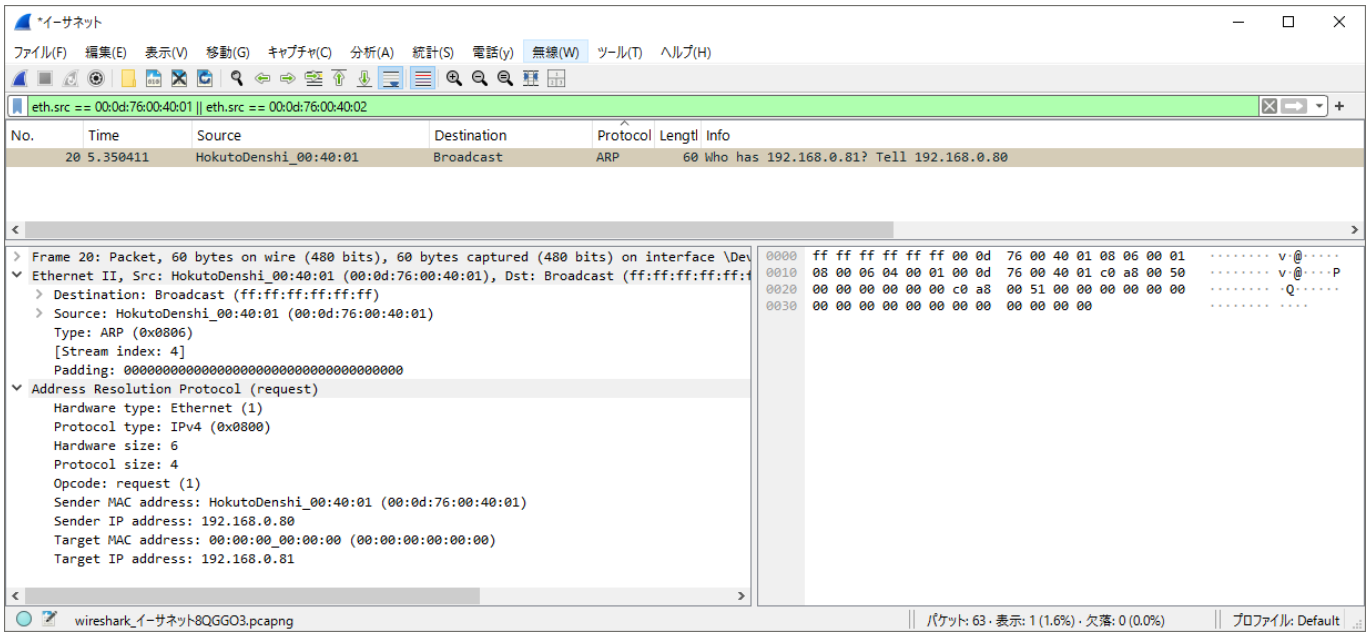
上記の様に合計 3 回のパケットのやり取りが行われているはずです。Wireshark で

・00-0D-76-40-01 が出しているパケット

または

・00-0D-76-40-02 が出しているパケット

でフィルタリングすると、



(1)00-0D-76-00-40-01 が FF-FF-FF-FF-FF-FF に対して送信している、ARP リクエスト

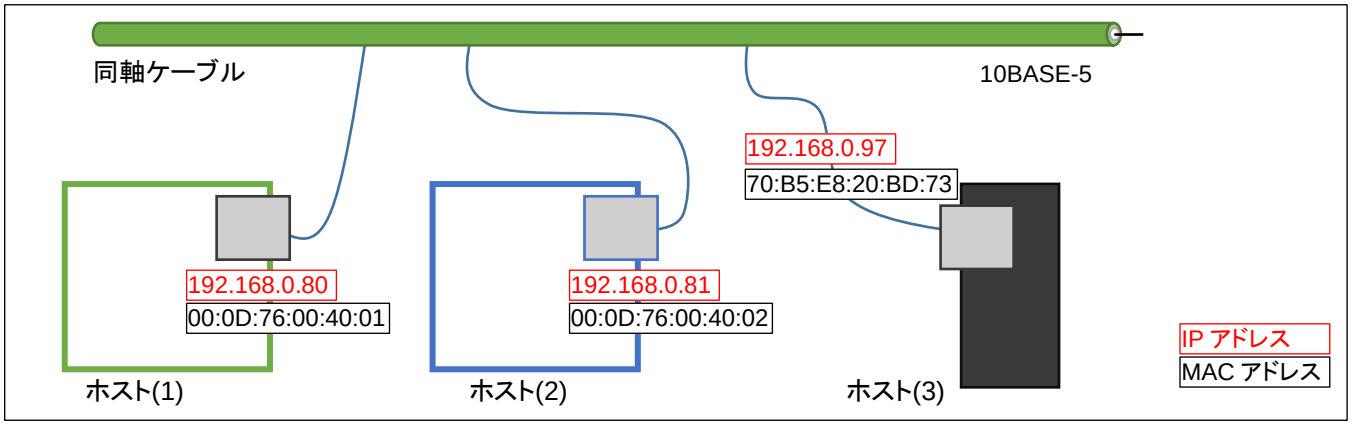
(2)00-0D-76-00-40-02 が 00-0D-76-00-40-01 に送っている ARP リプライ

(3)192.168.0.80(00-0D-76-00-40-01)が 192.168.0.81(00-0D-76-00-40-02)に対して送っている UDP データ

上記の内観測できているのは、(1)のみで、(2)(3)に関しては観測ができていません。

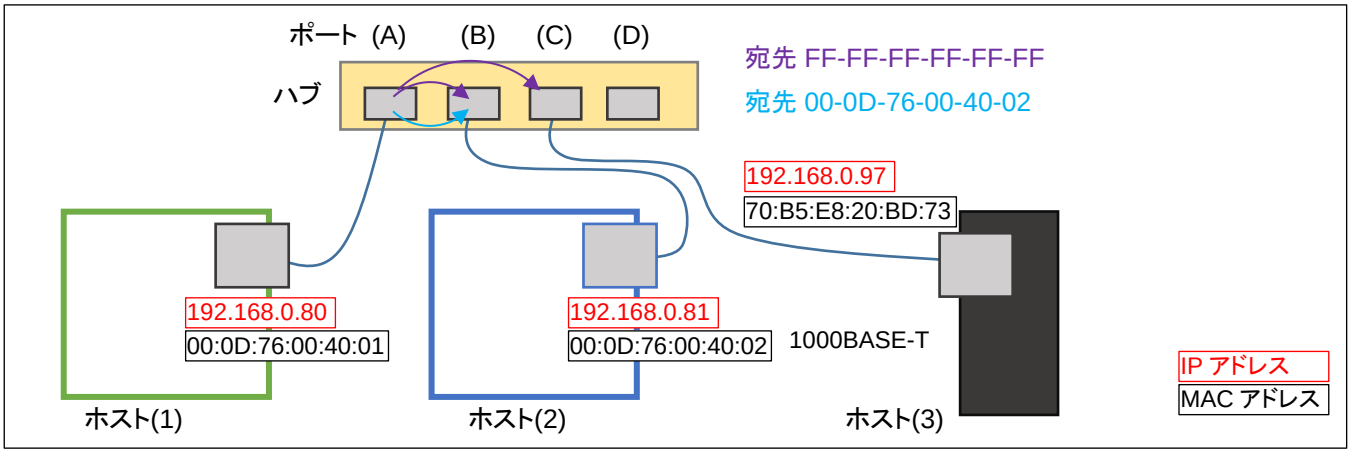
これは、ハブで接続しているのが原因です。

・昔の LAN



昔の LAN は、1 本の同軸ケーブルに全てのネットワークインタフェースが電気的に接続されており、ホスト(1)→ホスト(2)の通信はホスト(3)のネットワークインタフェースにも届く形です。

・現在主流の LAN



それに対し、ハブを使った接続の場合、ハブがどのポートにどの MAC アドレスの機器が接続されているかを把握しており、Ethernet フレームのデータを見て、宛先が FF-FF-FF-FF-FF-FF の時は全てのポートにデータを送る。宛先が 00-0D-76-00-40-02 の場合はポート(B)にしか送らないという動作を行っています。

よって、ホスト(2)からホスト(1)に対する ARP リプライや、ホスト(1)からホスト(2)に対する UDP データは、そもそもホスト(3)のネットワークインタフェース（ポート(C)）に物理的に届いては居ないのです。

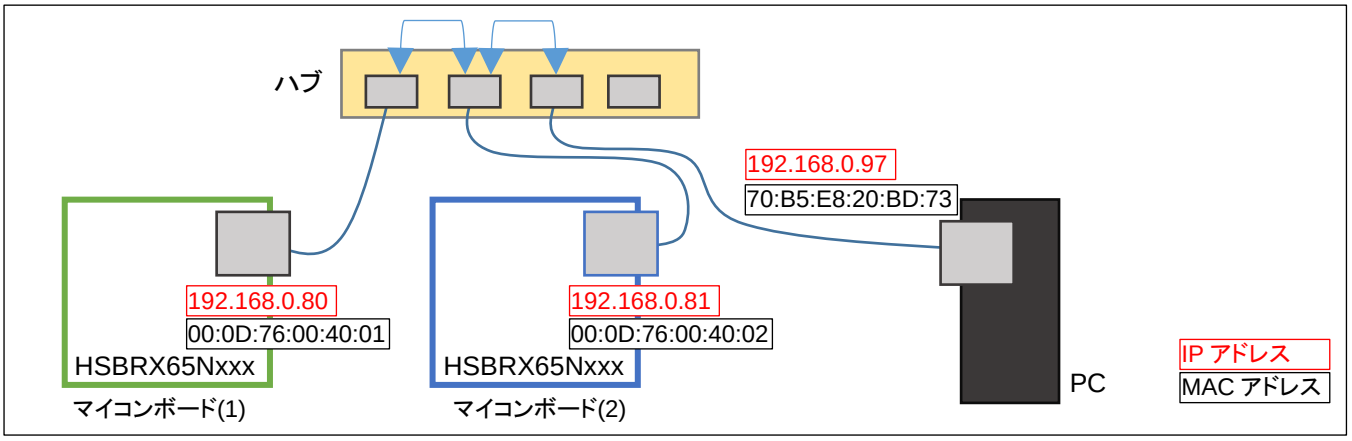
→よってホスト(3)で Wireshark を動かしても、ホスト(1)←→ホスト(2)のやり取りは観測できない

この様な Ethernet フレーム内の宛先 MAC アドレスを見てポートにデータを送るかどうかを決めるハブは、「スイッチングハブ」や「Layer2 スイッチ」と呼ばれ、現在市販されているハブは、ほぼこのタイプです。

（骨董品のハブは、ダムハブなどと呼ばれ、スイッチングの機能がないものもありますが、相当昔のものになると思います。）

ハブを使用して、ネットワーク内のパケットをダンプする方法としは、リピーターポートやマネージメント機能を持つハブ（ちょっと高価なハブ）を使用して、全てのパケットが流れる専用のモニタポート（RJ45 コネクタ）に、Wireshark を動かす PC を接続するという方法があります。

6.6.1. リピーターポートにマイコンボードを接続した場合の挙動



・PC とマイコンボード(1)間での UDP 通信
・リピーターポートのハブを使いマイコンボード(2)の LAN コネクタには PC とマイコンボード(1)の通信データは物理的には届いている

この条件の時、マイコンボード(2)では、192.168.0.80(00-0D-76-00-40-01)→192.168.0.97(70-B5-E8-20-BD-73)の通信は受信するかという話です。
結論から言うと、受信しません。
マイコンボード(2)は、MAC アドレスが 00-0D-76-00-40-02 に設定されているために、Ethernet ヘッダ内の宛先アドレスが、00-0D-76-00-40-02 または、FF-FF-FF-FF-FF-FF、マルチキャストビット=1 のデータのみ受信します。
なお、RX65N マイコンにおける MAC アドレス設定とは、ETHERC0.MAHR, ETHERC0.MALR レジスタに MAC アドレスの値を書き込む事です。
但し、 RX65N マイコンのイーサネットコントローラのレジスタで、ETHERC0.ECMR.BIT.PRM=1 に設定した場合は、プロミスキャスモード（＝無差別モード）での動作となり、Ethenet ヘッダ内の宛先アドレスに拘わらず全てのデータを受信する様になります。（デフォルトは、通常モード（＝非プロミスキャスモード）です。）
→RX65N マイコンは、マイコン内蔵のイーサネットコントローラのハードウェアレベルで Ethernet ヘッダの宛先アドレスを見て受信するか破棄するかを決めています。

7. TCP 通信に関して

TCP(Transport Control Protocol)は、ネットワークでデータを送るプロトコルですが、セッション（接続関係）を維持しながら通信を行う事が特徴のプロトコルです。

UDP とは異なり、事前に「これからデータをやり取りするための関係構築を図りましょう」と話しかけます。送った後で「ちゃんと届きましたよ」の返答も行います。複数のデータを送った際も、送った順番にデータを並び替えたりもしてくれます。結構至れり尽くせりのプロトコルです。現在、ファイル転送や、Web ページの閲覧。メールのやり取りなど、ネットワークを通して行われている通信の大多数が TCP と IP アドレスを使用した、TCP/IP によって成り立っています。ネットワーク通信の花形プロトコルといったところです。

但し、PC に比べリソースが限られる RX65N マイコンで TCP を使って通信を行う場合、TCP のルールに従って通信を行う事は結構大変です。複雑な処理をこなさなければならないというデメリットも存在します。

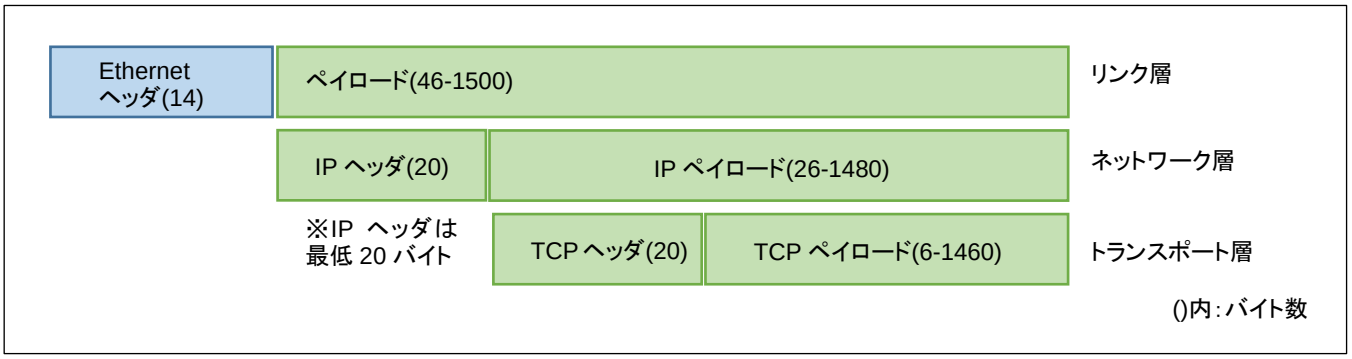


図 7-1 TCP の構造

まずはマイコンボードと PC 間で実際にデータのやり取りを行わせてみましょう。

7.1. PC に対してのデータ送信

マイコンボードには、RX65N_TCP プロジェクトの mot ファイルを書き込んでください。
PC 用のアプリケーションとして、PC_APPLI¥TCPTextServer.exe を実行してください。

・マイコンボード側

```
Copyright (C) 2026 HokutoDenshi. All Rights Reserved.
RX65N Ether TCP text client/server sample program.

COMMAND:
   t : send text message set
   p : send text message print
   s : send text message
   c : connect tcp session
   d : disconnect tcp session
   D : force disconnect tcp session

-setting address-
---
communication target IP address -> 192.168.0.81
this board IP address                -> 192.168.0.80 (MAC address -> 00-0D-76-00-40-01)
---
-setting port-
---
TCP server listen port -> 20000
TCP client dest     port -> 20000
TCP client source port -> 50000
---

address/port setting change -> Please input 'A' within 10 seconds.
>
```

起動すると上記のメッセージが出ます。

項目	規定値	備考
通信相手の IP アドレス	192.168.0.81	A コマンドで変更
マイコンボードの IP アドレス	192.168.0.80	A コマンドで変更
マイコンボードの MAC アドレス	00-0D-76-00-40-01	A コマンドで変更

項目	規定値	備考
データを受信するポート番号	20000	A コマンドで変更
データを送信する送信先ポート番号	20000	A コマンドで変更
データを送信する際に使用するポート番号	50000	A コマンドで変更

デフォルトでは上記設定になっています。通信先の PC の IP アドレスが 192.168.0.81 以外の場合は、起動後 10 秒以内に A コマンドを入力してください。（10 秒経過すると、自動的に動作開始となります）
（10 秒経過してしまった際は、ボードのリセットボタンを押してください）

・A コマンド入力時

```
>
-- IP/MAC address settings change --

COMMAND:
   1 : communication target IP address
   2 : this board IP address
   3 : this board MAC address
   4 : TCP server listen port (default = 20000)
   5 : TCP client dest port     (default = 20000)
   6 : TCP client source port (default = 50000)
   p : print setting
   e : exit setting
```

上記表示が出ますので、通信先の PC の IP アドレスを設定するために、1 を入力してください。

・1 コマンド入力時

```
communication target IP set(please input 3 letters(0-9) [or 1,2 letters(0-9) + Enter] number x 4)

IP[0](b31-24) >192
IP[1](b23-16) >168
IP[2](b15-8)   >0
IP[3](b7-0)    >97
IP address set to -> 192.168.0.97
```

UDP のプログラム同様、この画面では

1. 通信相手の IP アドレス
2. ボードの IP アドレス
3. ボードの MAC アドレス

4~6. TCP 通信で使用するポート番号

p. 現在の設定の表示

e: 設定終了(exit)

が選べます。

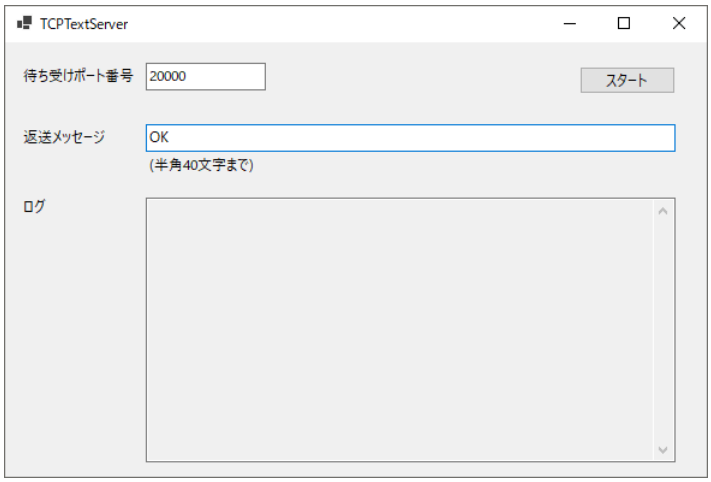
設定に問題が無ければ、e コマンド(exit)で入力を終了してください。

・e コマンドの入力

```
network operation start!
Ether0: LINK-UP
```

LINK-UP 表示が出ない場合は、LAN ケーブルを確認してください。

次に、PC 側で UDPTextServer.exe を実行します。



スタートを押してください。

マイコンボード側では、最初に c コマンドを入力してください。

・c コマンドの入力

```
>command=c
connect 192.168.0.97 port 20000
tcp_connection_start: 192.168.0.97 MAC address?
ARP request (192.168.0.97-> MAC address?) send
ARP reply received from 192.168.0.97, MAC address = 70-B5-E8-20-BD-73
TCP flag packet SYN send to 192.168.0.97:20000
TCP handshake ACK & SYN flag received OK.
TCP flag packet ACK send to 192.168.0.97:20000
-- CONNECTION ESTABLISHED --
```

最初は、192.168.0.97 の MAC アドアレスが判らないので、ARP リクエストで MAC アドレスの問い合わせを行います。この辺りは、UDP の動作と同じです。

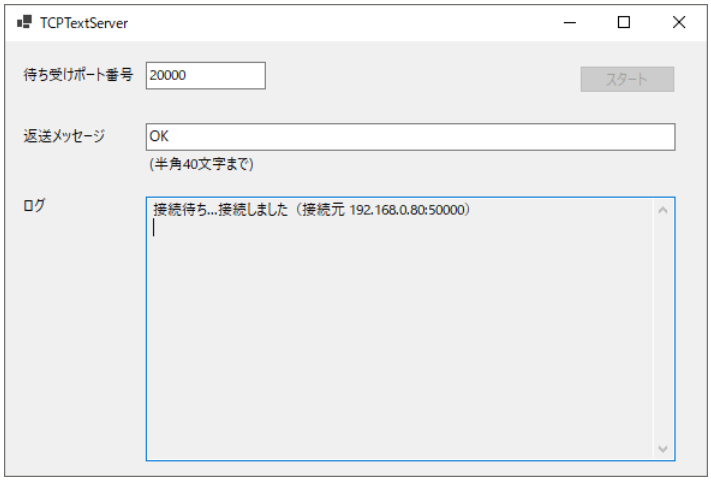
続いて、TCP のセッション（通信路を開く）を開通させます。

--CONNECTION ESTABLISHED—

の表示が出た場合は、接続成功です。

ここで接続先が、192.168.0.97:20000 と表示されていますが、PC 側の TCPTextServer.exe では、ポート番号 20000 で接続を待ち受ける仕様です。

この時 PC 側では、



上記の様に、「接続しました」の表示が出ます。（この時、接続を行ってきた機器の IP アドレスとポート番号(192.168.0.80:50000)を表示します。）（マイコン側では、ポート番号 50000 を使用しています。）

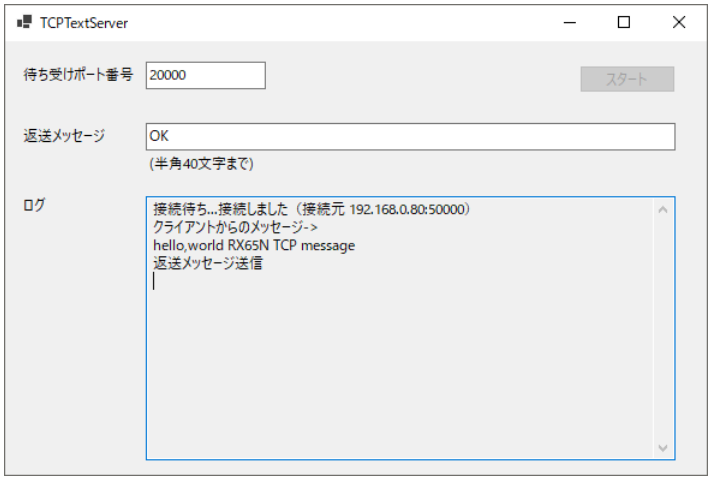
この状態は、マイコンボードと PC でお互いに TCP でデータの送受信ができる準備が整ってる状態、TCP のセッションが確立してデータのやり取りを行う通路が出来上がった形となります。

続いて、マイコンボードから s（送信）コマンドを入力してみます。

・s コマンドの入力

```
>command=s
send text message to 192.168.0.97 port 20000
TCP data packet (40 bytes) (flag = ACK PSH ) send to 192.168.0.97:20000
TCP frame received from 192.168.0.97:20000 Flag = ACK PSH , 40 bytes data
>OK
TCP flag packet ACK send to 192.168.0.97:20000
```

この時 PC 側では、

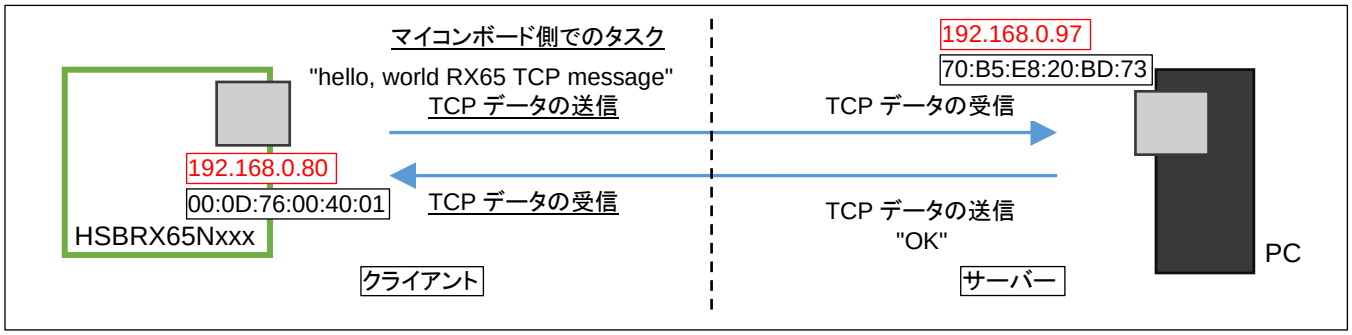


上記の様な表示となりますが、「hello, world RX65N TCP message」の部分がマイコンボードから送信したメッセージ（データ）です。また、「返送メッセージ送信」と表示されていますが、これは返送メッセージのテキストボックスに入力されているメッセージを返送したという意味です。

マイコンボード側では、

```
>command=s
send text message to 192.168.0.97 port 20000
TCP data packet (40 bytes) (flag = ACK PSH ) send to 192.168.0.97:20000
TCP frame received from 192.168.0.97:20000 Flag = ACK PSH , 40 bytes data
>OK
TCP flag packet ACK send to 192.168.0.97:20000
```

上記の OK の部分が、PC アプリケーションから返送されたメッセージとなります。



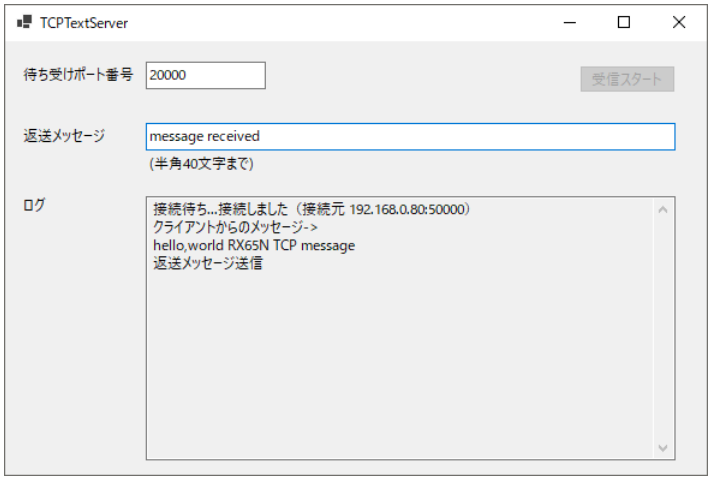
本プログラムでは、データ(メッセージ)送信時に相手側(この場合は PC アプリケーション)は、受け取った旨のメッセージを返すようにしています。

・t コマンドの入力

```
>command=t
send text message set (please input 40 letters(max), (ESC or CNTL-C for end)) > "abcde!"--end
```

t コマンドを入力し、メッセージ(この場合は abcde!)を入力。入力の終了は、ESC か CNTL-C です。メッセージは最大 40 文字までです。(40 文字というのは、本プログラムで設定しているだけで、プログラムを変更すれば最大 1460 バイトまで送れます。(なお、複数のパケットに分割して送信する場合は、1460 バイトの制約はありません。)

PC 側では、

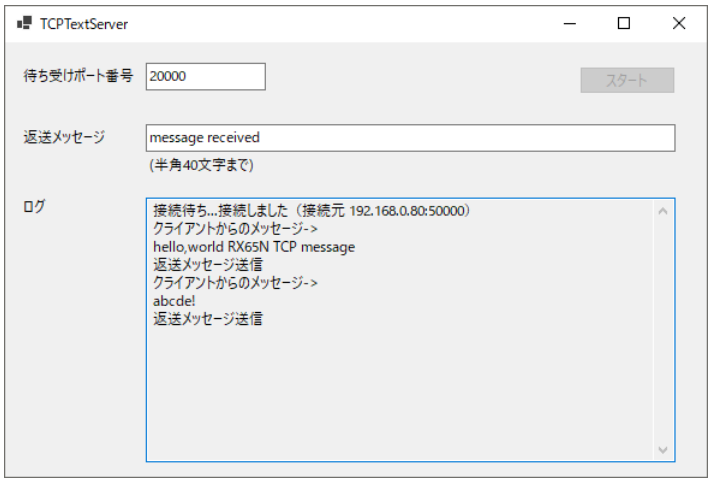


返送メッセージのボックスに好きなメッセージを入力しておきます。

メッセージ設定後、マイコンボード側で s コマンドを入力すると、

```
>command=s
send text message to 192.168.0.97 port 20000
TCP data packet (40 bytes) (flag = ACK PSH ) send to 192.168.0.97:20000
TCP frame received from 192.168.0.97:20000 Flag = ACK PSH , 40 bytes data
>message received
TCP flag packet ACK send to 192.168.0.97:20000
```

PC 側では



マイコン側で受け取るメッセージと、PC 側でのクライアントからのメッセージが変わります。

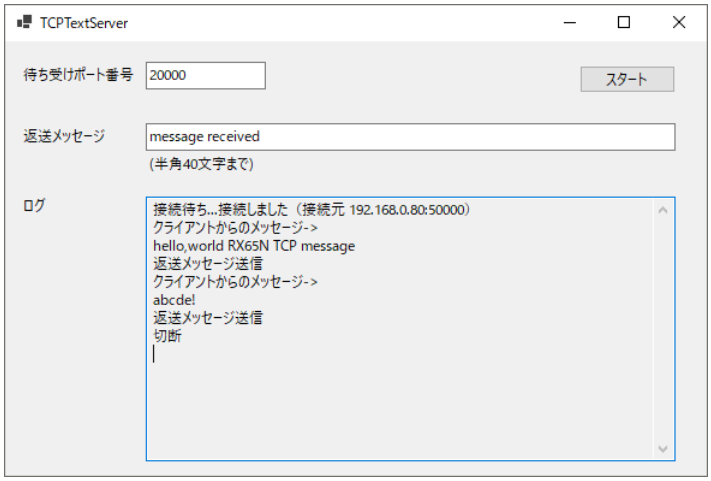
この時点では、TCP のセッションが生きている（マイコンボードと PC で通信路が通じている）状態です。TCP のセッションを終了させる場合は、マイコンボード側で d を入力してください。

・d コマンドの入力

```
>command=d
disconnect
TCP flag packet ACK FIN send to 192.168.0.97:20000
TCP handshake ACK flag received OK.
TCP handshake ACK & FIN flag received OK.
TCP flag packet ACK send to 192.168.0.97:20000
-- CONNECTION CLOSED --
```

-- CONNECTION CLOSED – と表示されれば、TCP のセッションは終了です。

PC 側でも、

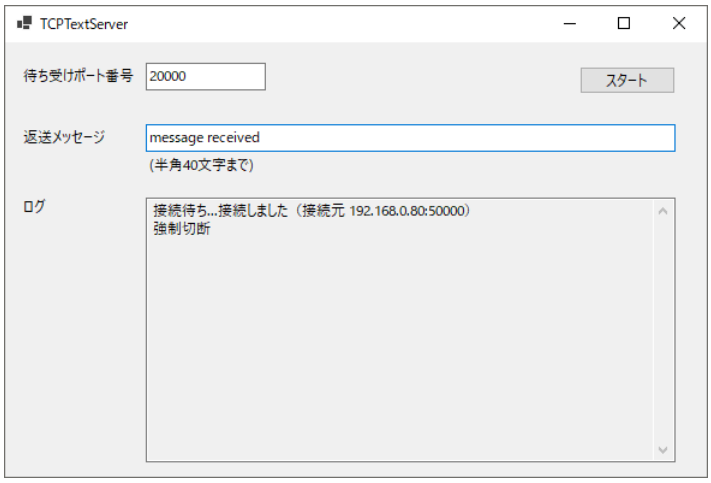


切断となります。（一度切断したら、再度「スタート」が押せるようになります。）

接続状態から d コマンドで切断となりますが、接続状態で大文字 D コマンドを入力すると、

・D コマンドの入力

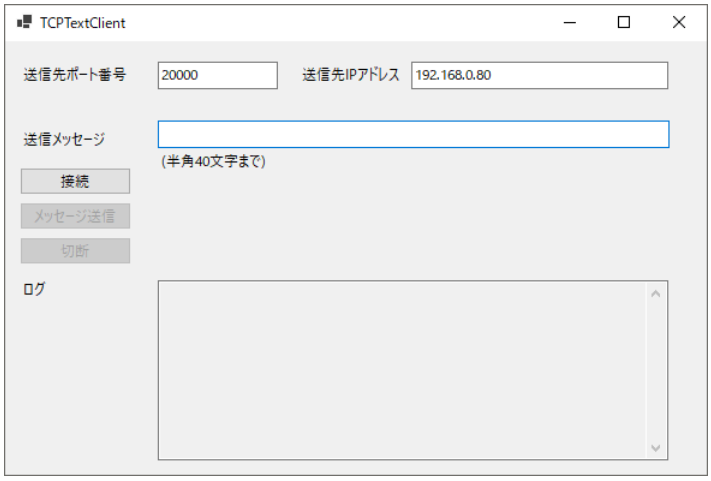
```
>command=D
disconnect(force)
TCP flag packet RST send to 192.168.0.97:20000
```



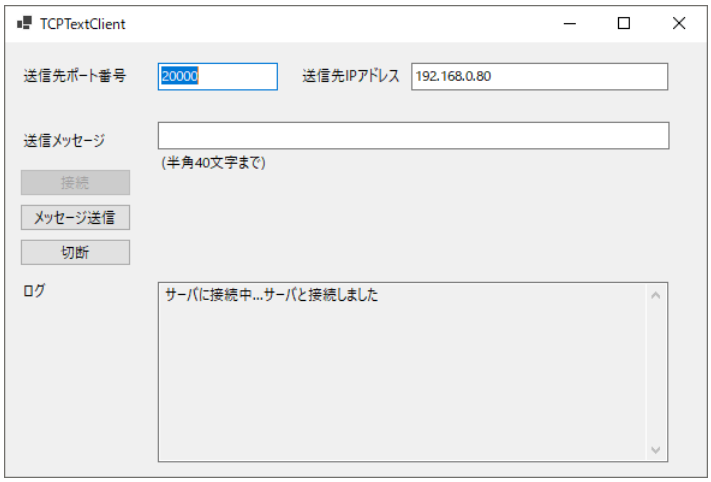
強制的に接続をリセット（切断）する動作となります。d と D の違いに関しては後述します。

7.2. PC からのデータ受信

PC_APPLI¥UDPTextClient.exe を起動してください。



上記の様な画面となりますので、接続先 IP アドレスにマイコンボードの IP アドレスを設定して、「接続」ボタンを押してください。（マイコンボードの IP アドレスを、192.168.0.80 から変えている場合は、送信先 IP アドレスの設定を変えてください。）



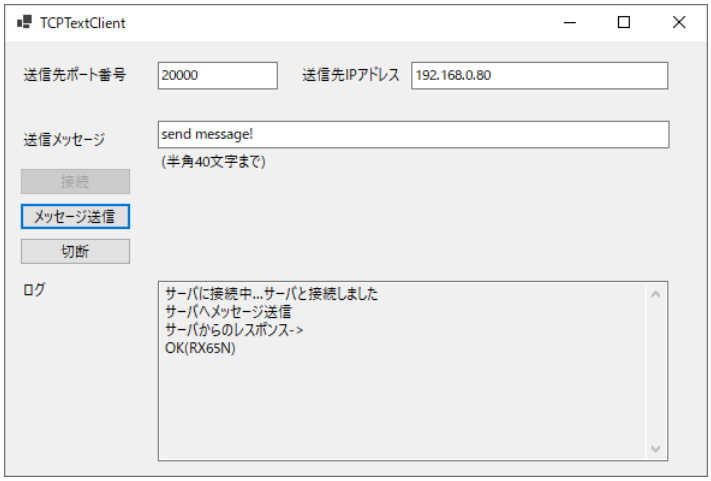
「サーバと接続しました」という表示となれば、接続は成功です。この状態は、TCP のセッションが成立した状態です。

・マイコンボード側の表示

```
tcp_connection_start: 192.168.0.97:62506
TCP handshake SYN flag received OK.
TCP flag packet ACK SYN send to 192.168.0.97:62506
TCP handshake ACK flag received OK.
-- CONNECTION ESTABLISHED --
```

-- CONNECTION ESTABLISHED – と表示されます。

次に、PC 側から何かメッセージを送ってみます。



送信メッセージの欄に、何か適当な文字列を設定して「メッセージを送信」ボタンを押します。

この時、サーバからのレスポンスとして、「OK(RX65N)」と返ってきたら、マイコンボード側は送信メッセージに反応しています。

※返送メッセージはプログラム内で「OK(RX65N)」に固定しています。(変更時はソースコードを編集してください)

マイコンボード側では、

```
TCP frame received from 192.168.0.97:62506 Flag = ACK PSH , 40 bytes data
>send message!
TCP data packet (40 bytes) (flag = ACK PSH ) send to 192.168.0.97:62506
TCP frame received from 192.168.0.97:62506 Flag = ACK
```

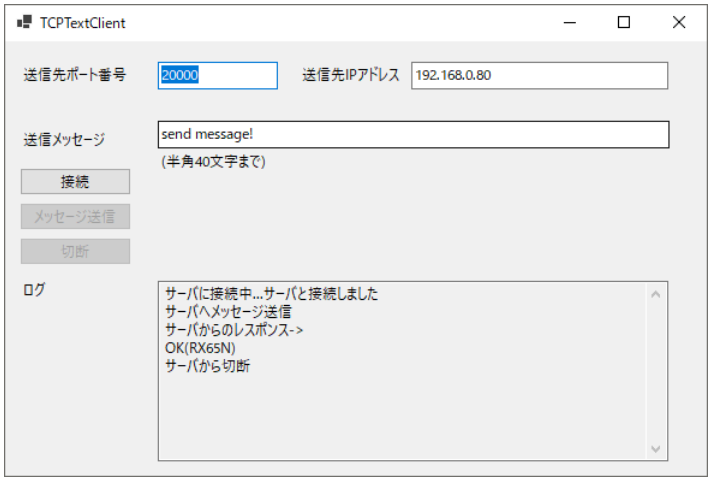
send message!　というメッセージを受信した旨の表示が出ます。

なお、上記での PC 側の使用ポートは 62506 です。この番号は、ランダムに変わります。UDP のプログラムでは、送信元ポートを 50000 に設定(テキストボックスに指定した番号)としていましたが、本プログラムでは、プログラム内では特定の値を設定せず自動的に設定される様にしています。なお、マイコンボード側は、ポート番号 20000 で接続を待ち受ける仕様です。(待ち受けポート番号は、PC アプリケーションと同じです。)マイコンボード側は、接続元の使用ポート(ここではたまたま 62506)と通信します。(接続元のポート番号は何番でも通信できます。)

メッセージ送信ボタンを押す度に、送信メッセージのボックスに入力したメッセージが送信されます。

※TCP のソフトでは、PC 側がクライントになる場合、ランダムにポート番号が決められます
マイコンボード側は、(デフォルト値)50000 で固定です

最後に、切断ボタンを押してください。



サーバから切断されて、再度接続可能な状態（起動時の状態）に戻ります。

この時、マイコンボード側は、

```
TCP frame received from 192.168.0.97:62507 Flag = ACK FIN
tcp_connection_idle: message: FIN flag received.
TCP flag packet ACK send to 192.168.0.97:62507
TCP flag packet ACK FIN send to 192.168.0.97:62507
TCP handshake ACK flag received OK.
-- CONNECTION CLOSED --
```

となり、TCP のセッションが終了します。

TCP では、

・TCP のセッションを確立する
・2 者間でデータのやり取りを行う（何回でも可）
・TCP のセッションを終了する

というのが一連の流れになります。
（UDP の様に前置きなしに、いきなりデータを送り付ける事ができない）

7.3. 2 台のマイコンボード間のデータ送信

2 台のマイコンボード間でデータ送信を行う場合は、書き込むプログラムは同じもので問題ありません。一方のボードの DIP-SW(3)を ON にして起動してください。(もう一方は OFF)

○1 台目

起動後 10 秒待つ。

```
Copyright (C) 2026 HokutoDenshi. All Rights Reserved.
RX65N Ether TCP text client/server sample program.

COMMAND:
   t : send text message set
   p : send text message print
   s : send text message
   c : connect tcp session
   d : disconnect tcp session
   D : force disconnect tcp session

-setting address-
---
communication target IP address -> 192.168.0.81
this board IP address           -> 192.168.0.80 (MAC address -> 00-0D-76-00-40-01)
---
-setting port-
---
TCP server listen port -> 20000
TCP client dest     port -> 20000
TCP client source port -> 50000
---

address/port setting change -> Please input 'A' within 10 seconds.
>.....
network operation start!
Ether0: LINK-UP
```

接続先は
192.168.0.81

○2 台目

```
(前略)
-setting address-
---
communication target IP address -> 192.168.0.80
this board IP address           -> 192.168.0.81 (MAC address -> 00-0D-76-00-40-02)
---
(後略)
```

サーバ側は communication target IP address の設定は未使用

このボードは
192.168.0.81

1 台目と 2 台目で、相互に通信が可能な様に設定されます。なお、3 台以上のマイコンボードを接続する場合は、起動後 10 秒以内に A コマンドを入力して重複しない様に適宜 IP アドレス等を設定してください。

ここでは、2 台目をサーバ(接続される側)、1 台目をクライアント(接続する側)とします。

※通信先 IP アドレス(communication target IP address)の設定は、クライアント動作時の接続先を決めます
サーバ動作時は、どの IP アドレスからの通信も受け入れます

UDP の場合は、2 台のボードで、通信相手(communication target IP address)を相互に設定する事が重要でしたが、TCP の場合

・ボード自体の IP アドレスが（MAC アドレスも）重複しない事

・クライアント側の通信相手の IP アドレスを適切に（＝サーバの IP アドレスに）設定する事

が重要です。

サーバ側の通信相手の IP アドレス設定は使われません。（サーバ側は、TCP の接続を試みてきた IP アドレスの相手と TCP セッションを確立し、TCP セッションを確立した相手にデータを返送します。）

・1 台目（クライアント側）から c コマンドを入力

```
>command=c
connect 192.168.0.81 port 20000
tcp_connection_start: 192.168.0.81 MAC address?
ARP request (192.168.0.81-> MAC address?) send
ARP reply received from 192.168.0.81, MAC address = 00-0D-76-00-40-02
TCP flag packet SYN send to 192.168.0.81:20000
TCP handshake ACK & SYN flag received OK.
TCP flag packet ACK send to 192.168.0.81:20000
-- CONNECTION ESTABLISHED --
```

・2 台目（サーバ側）の画面表示

```
ARP reply send (this board MAC address = 00-0D-76-00-40-02) to 00-0D-76-00-40-01
tcp_connection_start: 192.168.0.80:50000
TCP handshake SYN flag received OK.
TCP flag packet ACK SYN send to 192.168.0.80:50000
TCP handshake ACK flag received OK.
-- CONNECTION ESTABLISHED -
```

ARP の表示は、起動後はお互いの MAC アドレスを知りませんので、ARP リクエストを使って MAC アドレスの解決を試みます。その後、c コマンドを入力した側のボードから TCP セッションを確立する要求が飛んで、何度かやり取りを行った後に、接続が確立します。

・1 台目（クライアント側）から s コマンドを入力

```
>command=s
send text message to 192.168.0.81 port 20000
TCP data packet (40 bytes) (flag = ACK PSH ) send to 192.168.0.81:20000
TCP frame received from 192.168.0.81:20000 Flag = ACK PSH , 40 bytes data
>OK(RX65N)
TCP flag packet ACK send to 192.168.0.81:20000
```

「OK(RX65N)」の部分が、相手から受け取ったメッセージです。

・2 台目（サーバ側）の表示

```
TCP frame received from 192.168.0.80:50000 Flag = ACK PSH , 40 bytes data
>hello,world RX65N TCP message
TCP data packet (40 bytes) (flag = ACK PSH ) send to 192.168.0.80:50000
TCP frame received from 192.168.0.80:50000 Flag = ACK
```

2 台目（サーバ側）では、「hello, world RX65N TCP message」の部分が 1 台目（クライアント）から受け取ったメッセージです。

・1 台目（クライアント側）で送信するメッセージを変更して送信

t コマンドで送信メッセージを設定

```
>command=t
send text message set (please input 40 letters(max), (ESC or CNTL-C for end)) > "TCP data send"--end
```

s コマンドでサーバ側に送信

```
>command=s
send text message to 192.168.0.81 port 20000
TCP data packet (40 bytes) (flag = ACK PSH ) send to 192.168.0.81:20000
TCP frame received from 192.168.0.81:20000 Flag = ACK PSH , 40 bytes data
>OK(RX65N)
TCP flag packet ACK send to 192.168.0.81:20000
```

（赤字部分はサーバ側からのメッセージを受け取った旨の返信）

・2 台目（サーバ側）で受信

```
TCP frame received from 192.168.0.80:50000 Flag = ACK PSH , 40 bytes data
>TCP data send
TCP data packet (40 bytes) (flag = ACK PSH ) send to 192.168.0.80:50000
TCP frame received from 192.168.0.80:50000 Flag = ACK
```

・TCP セッションの終了

1 台目（クライアント側）から d コマンドを入力

```
>command=d
disconnect
TCP flag packet ACK FIN send to 192.168.0.81:20000
TCP handshake ACK flag received OK.
TCP handshake ACK & FIN flag received OK.
TCP flag packet ACK send to 192.168.0.81:20000
-- CONNECTION CLOSED --
```

・2 台目（サーバ側）の反応

```
TCP frame received from 192.168.0.80:50000 Flag = ACK FIN
tcp_connection_idle: message: FIN flag received.
TCP flag packet ACK send to 192.168.0.80:50000
TCP flag packet ACK FIN send to 192.168.0.80:50000
TCP handshake ACK flag received OK.
-- CONNECTION CLOSED --
```

サーバ、クライアント共、接続が切れた状態となります。（この状態の場合は、再度どちら側からでも c コマンドで接続可能）

本プログラムでは、s コマンドでメッセージ送信、d コマンドで接続できるのは c コマンドで接続した側（クライアント側）としています。

クライアント側からサーバに接続して、メッセージ送信、切断という流れです。

本来は、TCP セッションが確立した後は、サーバ、クライアントという区別はなく、サーバ（接続された側）からクライアント（最初に接続をリクエストしてきた側）に、データを送信しても、サーバ側から切断を行っても良いですが、本プログラムでは、接続後はクライアント発信で動作する様に構成しています。PC 側のアプリケーションも同様の構成としています。これは、別マニュアルで説明するプログラムの動作的に、そのような流れとした方が動作が判り易くなると考えたためです。

TCP の動作上は、接続された側（ここではサーバ側と呼んでいます）から切断のリクエストを行ったり、接続された側（サーバ側）からの発信でデータを送る事も問題ありません。

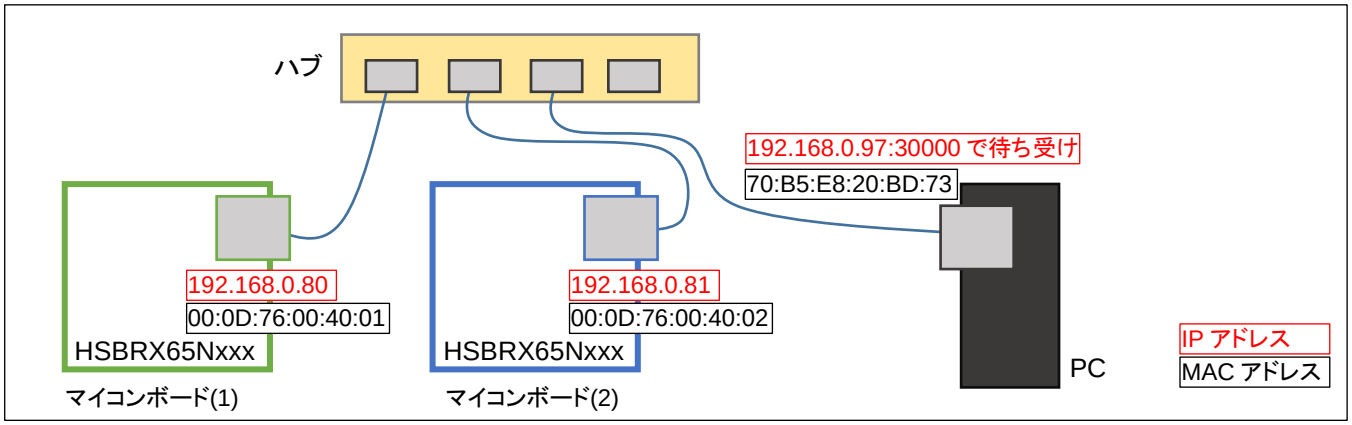
実際のマイコンボードや PC アプリケーションの動作を通して、なんとなく TCP 通信の流れが伝わったのではないかと思います。

※サーバ側とクライアント側の動作は独立しています
192.168.0.80 のサーバに、192.168.0.81 のクライアントが接続する
192.168.0.80 のクライアントが、192.168.0.82 のサーバに接続する
事を同時に行えます

この後、TCP の具体的な通信の流れに関して説明していきます。

7.4. UDP と TCP の違い

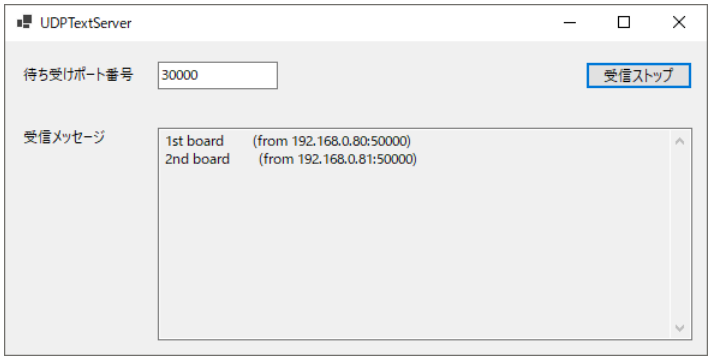
・UDP の場合



PC 側では、UDPTextServer.exe を起動して受信を開始します。

2 台のマイコンボードでそれぞれ、192.168.0.97:30000 に対して、UDP データ(メッセージ)を送信します。

・PC 側



・1 台目 s コマンドで送信(送信メッセージは"1st board")

```
>command=s
send text message to 192.168.0.97 port 30000

send target 192.168.0.97 -> MAC address not resolved.
ARP request (192.168.0.97-> MAC address?) send
ARP reply received from 192.168.0.97, MAC address = 70-B5-E8-20-BD-73
UDP text data send to 192.168.0.97:30000
```

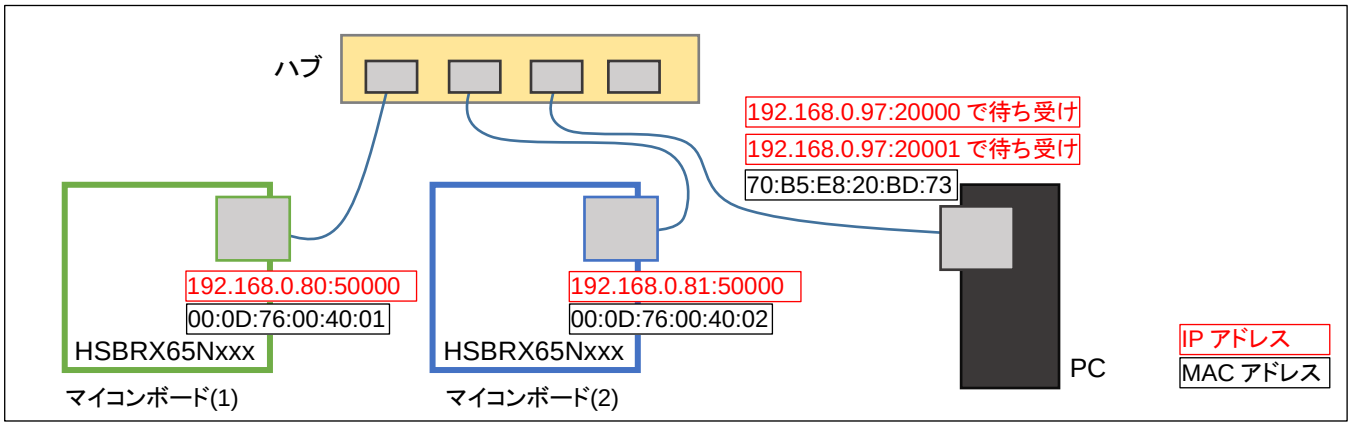
・2 台目 s コマンドで送信(送信メッセージは"2nd board")

```
>command=s
send text message to 192.168.0.97 port 30000

send target 192.168.0.97 -> MAC address not resolved.
ARP request (192.168.0.97-> MAC address?) send
ARP reply received from 192.168.0.97, MAC address = 70-B5-E8-20-BD-73
UDP text data send to 192.168.0.97:30000
```

この場合、PC は 1 台目(192.168.0.80:50000)からのメッセージと、2 台目(192.168.0.81:50000)からのメッセージはを両方受信して表示します。

UDP は、待ち受け側（この場合は PC）は、30000 番ポートに来たデータを表示する。どこから来ても問題はない。最初に 192.168.0.80 から来たからといって、192.168.0.80 の機器と紐付けされる訳でも無いという動作です。

・TCP の場合



PC 側では、TCPTextServer.exe を起動して受信を開始します。

2 台のマイコンボードでそれぞれ、192.168.0.97 に対して、TCP で接続を試みます。

・1 台目接続先設定 A コマンド内の p コマンド

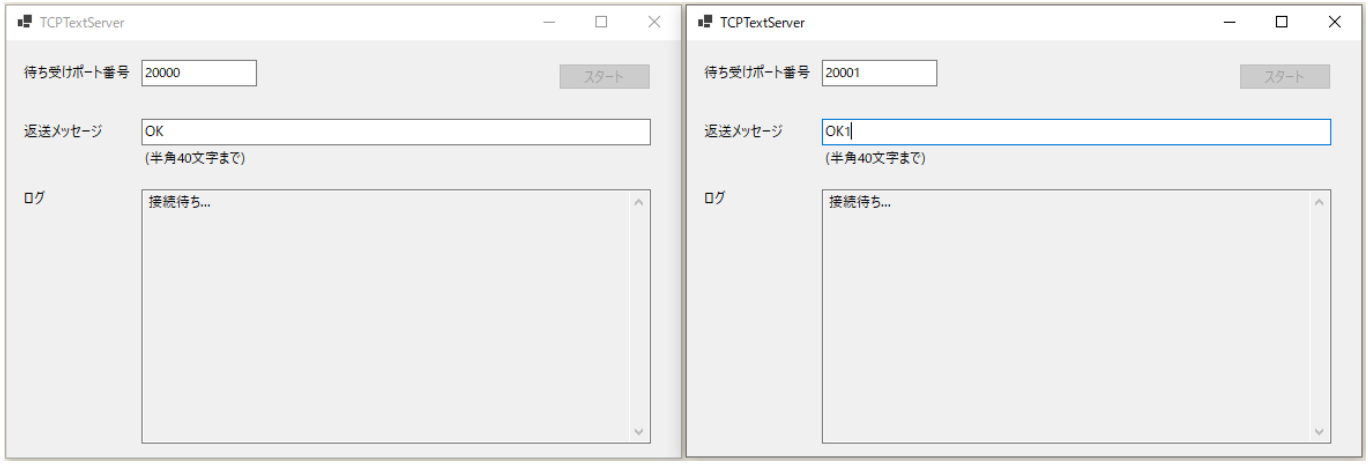
```
>-setting address-
---
communication target IP address -> 192.168.0.97
this board IP address           -> 192.168.0.80 (MAC address -> 00-0D-76-00-40-01)
---
-setting port-
---
TCP server listen port -> 20000
TCP client dest     port -> 20000
TCP client source port -> 50000
---
```

・2 台目接続先設定 A コマンド内の p コマンド

```
>TCP client dest port set>20001
>-setting address-
---
communication target IP address -> 192.168.0.97
this board IP address           -> 192.168.0.81 (MAC address -> 00-0D-76-00-40-02)
---
-setting port-
---
TCP server listen port -> 20000
TCP client dest     port -> 20001
TCP client source port -> 50000
---
```

接続先ポート番号を変更します

・PC 側



待ち受けポート 20000 と 20001 で、TCPTextServer.exe を 2 つ起動します。

・1 台目 c コマンドで接続

```
>command=c
connect 192.168.0.97 port 20000
tcp_connection_start: 192.168.0.97 MAC address?
ARP request (192.168.0.97-> MAC address?) send
ARP reply received from 192.168.0.97, MAC address = 70-B5-E8-20-BD-73
TCP flag packet SYN send to 192.168.0.97:20000
TCP handshake ACK & SYN flag received OK.
TCP flag packet ACK send to 192.168.0.97:20000
-- CONNECTION ESTABLISHED --
```

・2 台目 c コマンドで接続

```
>command=c
connect 192.168.0.97 port 20001
tcp_connection_start: 192.168.0.97 MAC address?
ARP request (192.168.0.97-> MAC address?) send
ARP reply received from 192.168.0.97, MAC address = 70-B5-E8-20-BD-73
TCP flag packet SYN send to 192.168.0.97:20001
TCP handshake ACK & SYN flag received OK.
TCP flag packet ACK send to 192.168.0.97:20001
-- CONNECTION ESTABLISHED --
```

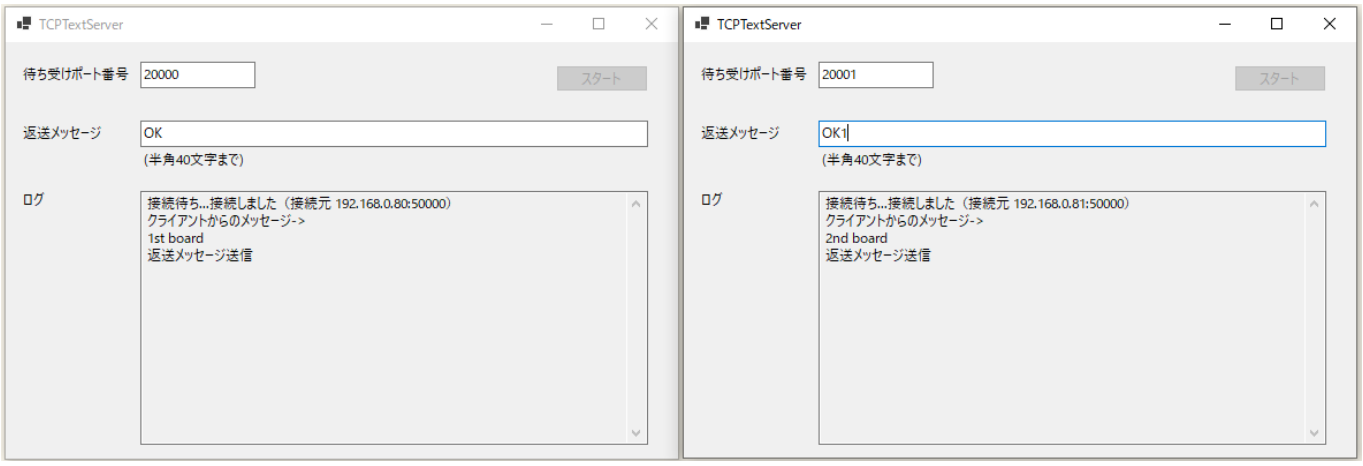
・1 台目 s コマンドで送信（送信メッセージは"1st board"）

```
>command=s
send text message to 192.168.0.97 port 20000
TCP data packet (40 bytes) (flag = ACK PSH ) send to 192.168.0.97:20000
TCP frame received from 192.168.0.97:20000 Flag = ACK PSH , 40 bytes data
>OK
TCP flag packet ACK send to 192.168.0.97:20000
```

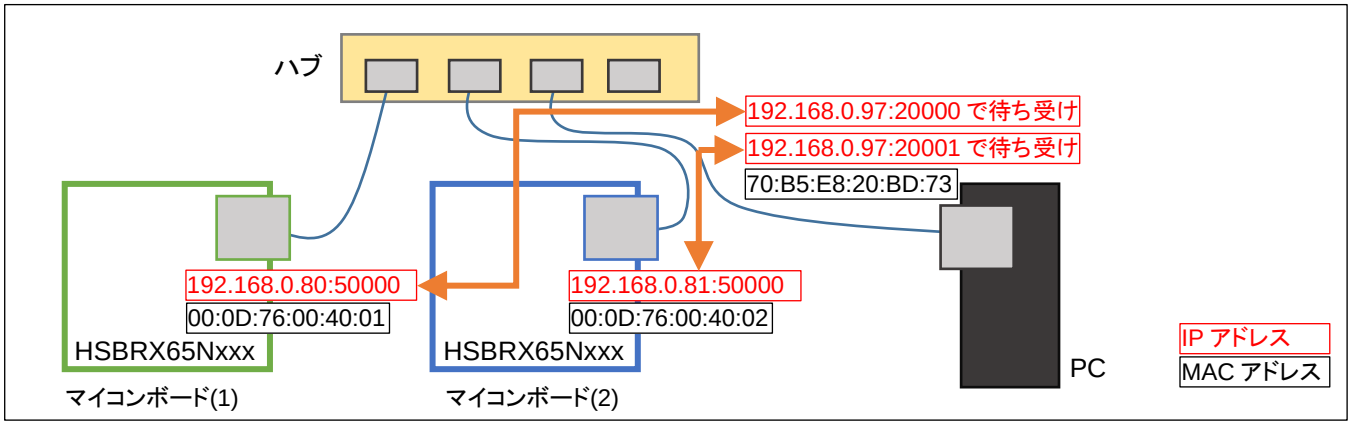
・2 台目 s コマンドで送信（送信メッセージは"2nd board"）

```
>command=s
send text message to 192.168.0.97 port 20001
TCP data packet (40 bytes) (flag = ACK PSH ) send to 192.168.0.97:20001
TCP frame received from 192.168.0.97:20001 Flag = ACK PSH , 40 bytes data
>OK1
TCP flag packet ACK send to 192.168.0.97:20001
```

・PC 側



それぞれのボードと通信ができています。



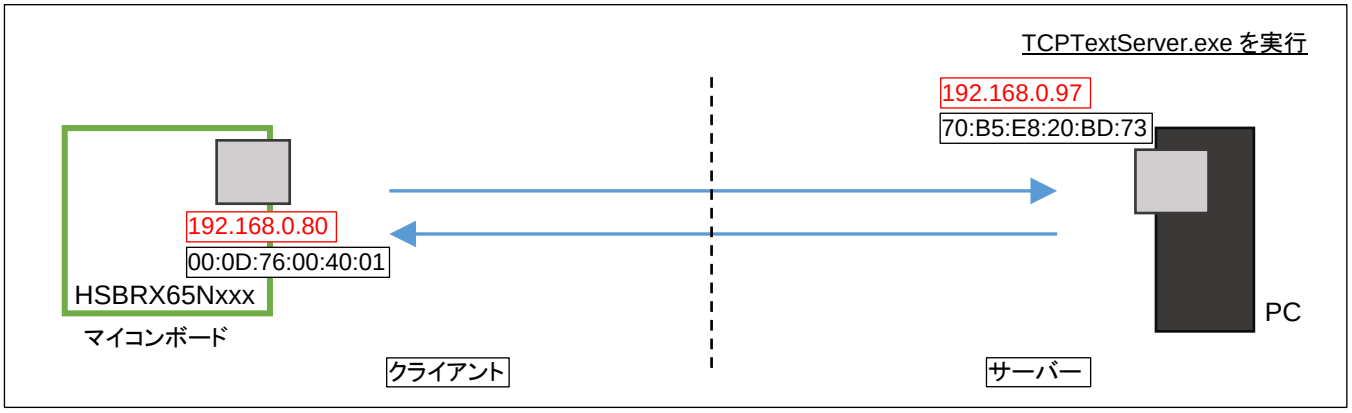
この場合は、192.168.0.80:50000 と 192.168.0.97:20000 が接続関係にあり、同じく 192.168.0.81:50000 と 192.168.0.97:20001 が接続関係にあります。メッセージのやり取り（データの送受信）が行えるのは、接続関係を保っている間です。

UDP は一度データを送ったら、その後は関係がないというドライな関係ですが、TCP の場合は、一連の通信が終わるまでは接続している状態を維持しています。UDP はコネクションレスな通信。TCP はコネクション型というのが、UDP と TCP の一番の違いとなるかと考えます。

※TCP で複数のホストと通信を行う場合は、サーバ側は接続ホスト毎に複数のポートを使う必要があります
（マイコンボード側の RX65N_TCP プロジェクトでは、サーバは 1 つしか起動できません）
（プログラムに手を入れれば、複数のサーバの起動は可能です）

7.5. TCP パケットの中身

・データを例示する系



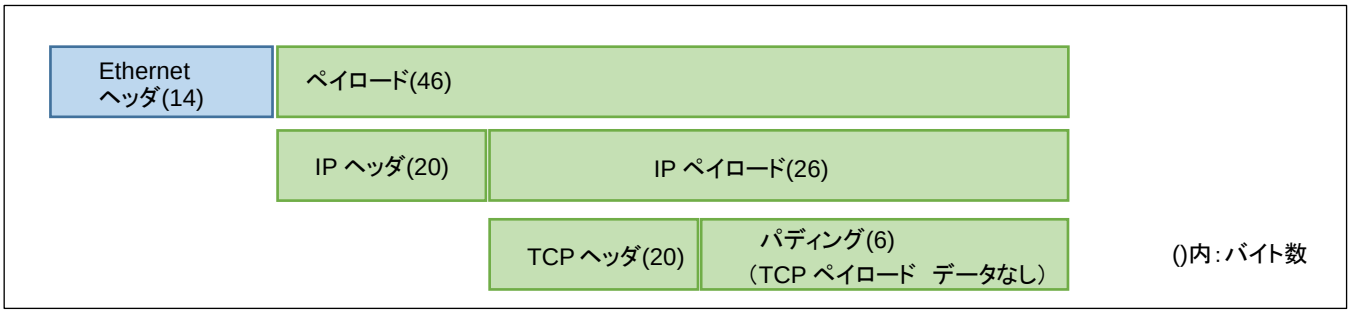
ここでは、具体的な通信パケットの中身として、PC 側で TCPTextServer.exe を実行した場合のデータで説明します。（マイコンボードがサーバ側となる場合でも、送信元と送信先の IP アドレスや MAC アドレスが逆になるだけで、基本的には変わりません。）

7.5.1. 接続時（3 ウェイハンドシェイク）

マイコンボード側では、接続先を 192.168.0.97 に設定して、c コマンドで接続を確立する過程です。

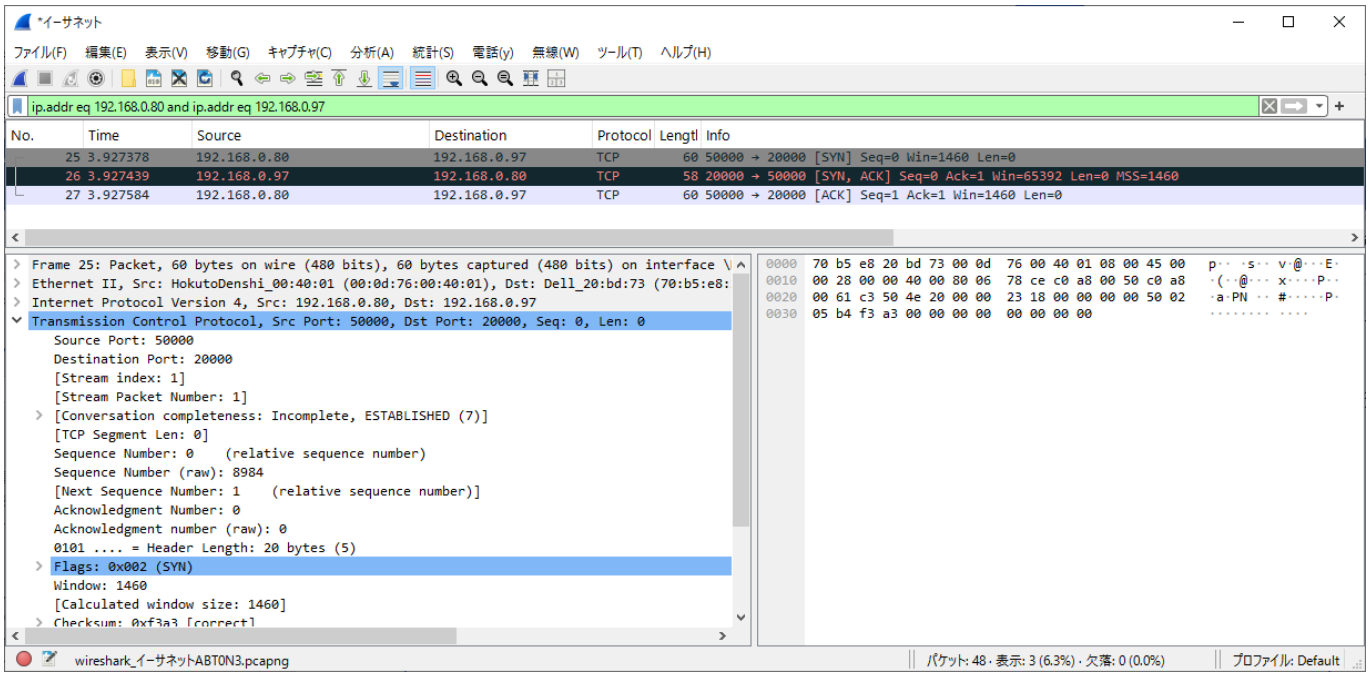
```
>command=c
connect 192.168.0.97 port 20000
tcp_connection_start: 192.168.0.97 MAC address?
ARP request (192.168.0.97-> MAC address?) send
ARP reply received from 192.168.0.97, MAC address = 70-B5-E8-20-BD-73
TCP flag packet SYN send to 192.168.0.97:20000
TCP handshake ACK & SYN flag received OK.
TCP flag packet ACK send to 192.168.0.97:20000
-- CONNECTION ESTABLISHED --
```

－TCP データの構造－



Ethernet フレームとしては 60 バイトのデータ。Ethernet フレームのペイロードは、IP ヘッダ＋IP ペイロードで構成。IP のペイロードは、さらに TCP のヘッダ＋TCP ペイロードで構成される形です。最初の接続時の通信では、TCP データの中身（ペイロード）はなく、ヘッダ情報のみのやり取りになります。（メッセージを送る際は、TCP ペイロードの部分にデータが入ります。）

・Wireshark でのパケットダンプ



マイコンボードから c コマンドで接続を確立する過程ですが、

(1)マイコンボード(192.168.0.80)→PC(192.168.0.97)

(2)PC(192.168.0.97)→マイコンボード(192.168.0.80)

(3)マイコンボード(192.168.0.80)→PC(192.168.0.97)

の合計 3 回のデータが送信されています。(そのため、3 ウェイハンドシェイクと呼ばれます)

(TCP のデータのやり取りの前に ARP リクエストや ARP リプライのパケットのやり取りはありますが、ここでは省略します。)

－(1)のマイコンボードから PC の通信パケット例－

・Ethernet ヘッダ(14)

70	B5	E8	20	BD	73	00	0D	76	00	40	01	08	00
宛先 MAC アドレス						送信元 MAC アドレス						Ethernet type (IPv4)	

・IP ヘッダ(20)

45	00	00	28	00	00	40	00	80	06	78	CE	C0	A8	00	50
IPv4 ヘッダ 20B	Service type	Total Length 40 バイト		ID (*1)		Fragment		TTL	Protocol TCP	Checksum		送信元 IP アドレス (192.168.0.80)			

C0	A8	00	61
送信先 IP アドレス (192.168.0.97)			

・TCP ヘッダ(20)

C3	50	4E	20	00	00	23	18	00	00	00	00	50	02	05	B4
Source port (50000)		Destination Port (20000)		Sequence Number				Acknowledgment Number				Headersize + Flags		Window 1460	

F3	A3	00	00
Checksum		Urgent Pointer	

・パディング(6)　　…Ethernet フレーム最低 60 バイトのためパディングデータ追加

00	00	00	00	00	00

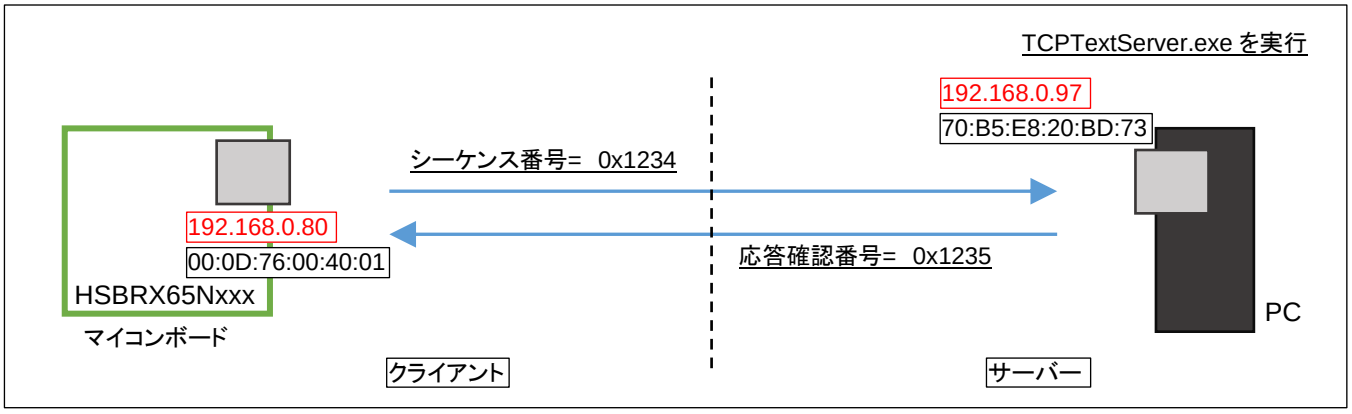
(*1)IP ヘッダ内の ID は、マイコンボードから送信する場合、0x0000→0x0001→0x0002 の様に単純にインクリメントした値としています。

〇TCP ヘッダ（20 バイト）

フィールド	bit 数	内容	データ例
Source Port	16	送信元ポート番号	C3 50
Distination Port	16	送信先ポート番号	4E 20
SequenceNumber	32	シーケンス番号	00 00 23 18
Acknowledgment Number	32	応答確認番号	00 00 00 00
Headersize + Flags	16	ヘッダサイズ＋TCP フラグ	50 02
Window	16	ウィンドウサイズ	05 B4
Checksum	16	チェックサム	F3 A3
Urgent Pointer	16	緊急ポインタ	00 00
Option		オプション	

TCP においても UDP 同様、IP アドレスに加え、ポート番号でやり取りされるデータの交通整理を行います。

TCP は、2 者（ここではマイコンボードと PC）間でコネクション（接続関係）を維持して通信を行うプロトコルです。また、パケットを正常に受信したかを相手に返す動作が入ります。一方がシーケンス番号 0x1234 のパケットを投げると、相手はシーケンス番号+1=(0x1235)の応答確認番号を返す動作となります。



マイコンボード側のシーケンス番号は、乱数で値を決めていますが、乱数の種は固定しているので、毎回同じ番号から始まる動作となります。

初回の応答確認番号は、0 から始まります。（0 を送っています）

ヘッダサイズ＋TCP フラグは、0x5002=0b0101 0000 0000 0010 の場合、以下の様になります。

0b**0101** 0000 0000 0010	b15-b12:ヘッダサイズ	5×4=20 バイト
0b0101 **000**0 0000 0010	b11-b9:予約	
0b0101 000**0** 0000 0010	b8:Accurate ECN	
0b0101 0000 **0**000 0010	b7:CongestionWindowReduced	
0b0101 0000 0**0**00 0010	b6:ECN-Echo	
0b0101 0000 00**0**0 0010	b5:Urgent	
0b0101 0000 000**0** 0010	b4:Acknowledgment	応答
0b0101 0000 0000 **0**010	b3:Push	データを送信（バッファリングせずアプリに送る）
0b0101 0000 0000 0**0**10	b2:Reset	強制切断
0b0101 0000 0000 00**1**0	b1:Syn	接続の開始
0b0101 0000 0000 001**0**	b0:Fin	接続の終了

TCP のヘッダサイズは可変で、最小サイズが 20 バイトで、4 の倍数のバイト数となります。その他、各種フラグがあり、ヘッダサイズとフラグを合わせて 16bit のデータで構成されています。

今回は、接続の初回なので Syn フラグを立ててパケットを送信しています。

ウィンドウサイズは、相手からの応答を待たずに連続してどのぐらいのデータを送ってよいかを示す値です。マイコンボードの場合は、1 つの Ethernet フレームで送れる TCP の最大サイズである 1460 バイトを相手に通知しています。（PC の場合は、もっと大きな値を通知したりします。）

チェックサムは、TCP ヘッダ＋データのチェックサムです。（今回のパケットではデータはありません）UDP 同様、疑似ヘッダを使用した計算アルゴリズムとなります。

〇疑似ヘッダ（12 バイト） ※UDP と同様

フィールド	bit 数	内容	データ例
Source IP Address	32	送信元ポート番号	C0 A8 00 50
Distination IP Address	32	送信先ポート番号	C0 A8 00 61
Zero padding	8	パディング(0x00)	00
Protocol	8	プロトコル（TCP の場合は 0x06）	06
Length	16	TCP セグメントの長さ（ヘッダ＋データ）[バイト]	00 14

疑似ヘッダは、IP アドレスとプロトコルと TCP セグメント（TCP のヘッダと TCP のデータを合わせた長さ）の情報から構成されます。これら、疑似ヘッダ＋TCP ヘッダ＋TCP データで、UDP の章で説明した 1 の補数和で計算する形です。

・疑似ヘッダ

C0A8	+	0050	=	C0F8
C0F8	+	C0A8	=	181A0
81A1	+	0061	=	8202
8202	+	0006	=	8208
8208	+	0014	=	821C

→81A1（bit16 を 0 にして、1 を加算する）

・TCP ヘッダ

821C	+	C350	=	1456C	→456D
456D	+	4E20	=	938D	
938D	+	0000	=	938D	
938D	+	2318	=	B6A5	
B6A5	+	0000	=	B6A5	
B6A5	+	0000	=	B6A5	
B6A5	+	5002	=	106A7	→06A8
06A8	+	05B4	=	0C5C	
0C5C	+	0000	=	0C5C	チェックサムは 0x0000 で計算
0C5C	+	0000	=	0C5C	

・TCP データ

（今回はなし）

~0C5C	=	F3A3

上記の様な計算で、TCP ヘッダ内の Checksum を計算可能です。

緊急ポインタは、通常のパケットでは 0 とします。

オプション、TCP ヘッダサイズが 20 バイトより大きい場合は、この部分にオプションのヘッダのデータが入ります。

接続時（3 ウェイハンドシェイクの 1 回目）のパケットは、Syn フラグを立てたパケットを送信するという点がポイントとなります。

3 ウェイハンドシェイクの 2 回目は以下の様になります。

－(2)の PC からマイコンボードの通信パケット例－

・Ethernet ヘッダ(14)

00	0D	76	00	40	01	70	B5	E8	20	BD	73	08	00
宛先 MAC アドレス						送信元 MAC アドレス						Ethernet type (IPv4)	

・IP ヘッダ(20)

45	00	00	2C	4D	8B	40	00	80	06	00	00	C0	A8	00	61
IPv4 ヘッダ 20B	Service type	Total Length 44 バイト		ID		Fragment		TTL	Protocol TCP	Checksum (*1)		送信元 IP アドレス (192.168.0.97)			

C0	A8	00	50
送信先 IP アドレス (192.168.0.80)			

・TCP ヘッダ(24)

4E	20	C3	50	65	14	0F	03	00	00	23	19	60	12	FF	70
Source port (20000)		Destination Port (50000)		Sequence Number (*2)				Acknowledgment Number (*3)				Headersize + Flags		Window 65392	

82	20	00	00	02	04	05	B4
Checksum		Urgent Pointer		Option (*4)			

・パディング(2)　　Ethernet フレーム最低 60 バイトのためパディングデータ追加

（※Wireshark では見えていない、ネットワークインタフェースで付与）

00	00

(*1)Checksum はネットワークボード側で付与されるので、ここでは 0x0000 で見えています

(*2)シーケンス番号 Sequence Number は、PC からの送信は初回なので、ここではランダムな値が設定されます

(*3)応答確認番号 Acknowledgment Number は、マイコンボードから送信したシーケンス番号+1 が設定されます

(*4)オプション(Maximum Segument Size=2, Length=4, MSS Value=1460)

PC から送信した TCP パケットでは、ヘッダサイズが 24 バイトになっています。

ヘッダサイズとフラグは、0x6012 となっており、

0b**0110** 0000 0001 0010	b15-b12:ヘッダサイズ	6×4=20 バイト
0b0110 0000 000**1** 0010	b4:Acknowledgment	応答
0b0110 0000 0001 00**1**0	b1:Syn	接続の開始

Acknowledgment と Syn のフラグがセットされています。

3 ウェイハンドシェイクの 3 回目は以下の様になります。

－(3)のマイコンボードから PC の通信パケット例－

・Ethernet ヘッダ(14)

70	B5	E8	20	BD	73	00	0D	76	00	40	01	08	00
宛先 MAC アドレス						送信元 MAC アドレス						Ethernet type (IPv4)	

・IP ヘッダ(20)

45	00	00	28	00	01	40	00	80	06	78	CD	C0	A8	00	50
IPv4 ヘッダ 20B	Service type	Total Length 40 バイト		ID (*1)		Fragment		TTL	Protocol TCP	Checksum		送信元 IP アドレス (192.168.0.80)			

C0	A8	00	61
送信先 IP アドレス (192.168.0.97)			

・TCP ヘッダ(20)

C3	50	4E	20	00	00	23	19	65	14	0F	04	50	10	05	B4
Source port (50000)		Destination Port (20000)		Sequence Number (*2)				Acknowledgment Number (*3)				Headersize + Flags		Window 1460	

7F	7C	00	00
Checksum		Urgent Pointer	

・パディング(6)　　…Ethernet フレーム最低 60 バイトのためパディングデータ追加

00	00	00	00	00	00

(*1)マイコンボードからの送信 2 回目なので、単純に 0x0000 を 1 回インクリメントした値（本プログラムでは、そのようにしています）

(*2)ここでは、シーケンス番号 Sequence Number は、相手が返してきた応答確認番号を返します

(*3)応答確認番号 Acknowledgment Number は、相手が送信したシーケンス番号+1 を返します

ヘッダサイズとフラグは、0x0510 としており、

0b0101 0000 0001 0000	b4:Acknowledgment	応答

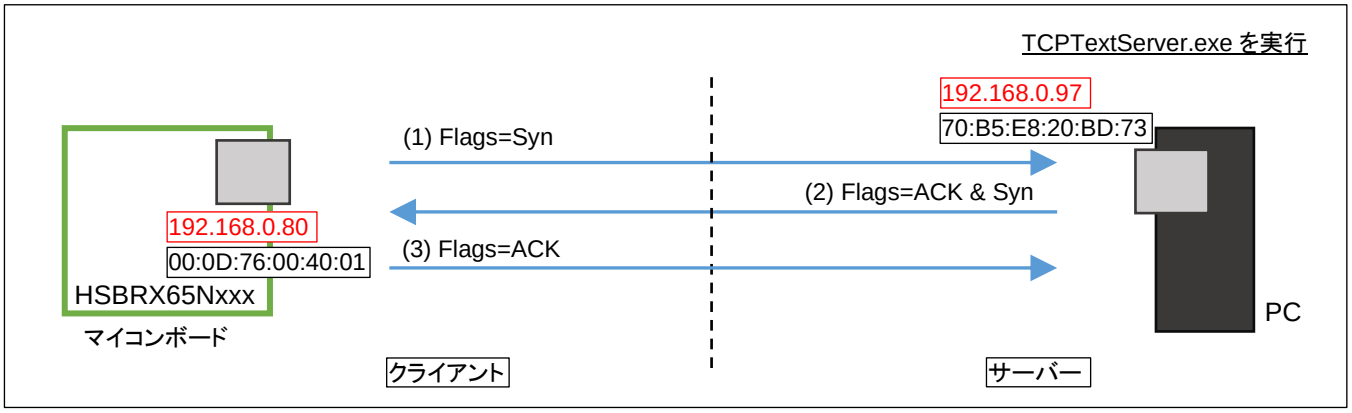
Acknowledgment のフラグを立てて送信します。

この(1)～(3)で接続のハンドシェイクの処理は終わりです。

・TCP フラグを適切に立てる

・シーケンス番号、応答確認番号を正しく処理する

点が、TCP の通信で必要な事項になります。



(1)クライントからサーバに対して Syn フラグを送る

(2)サーバからクライントに対して Syn と ACK フラグを送る

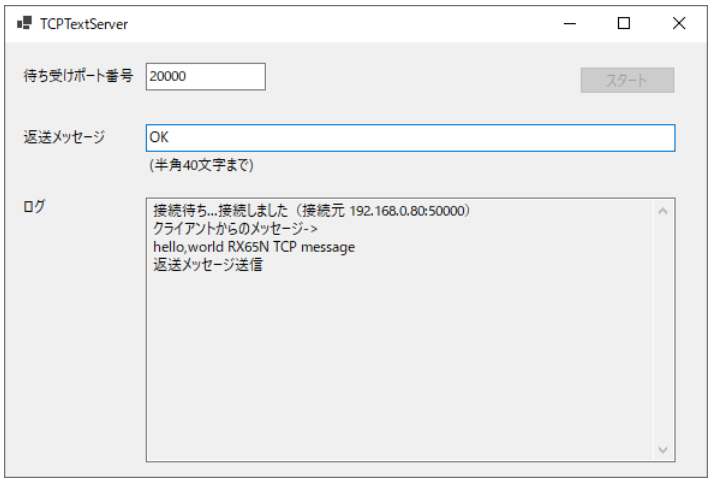
(3)クライアントからサーバに対して ACK フラグを送る

上記が、3 ウェイハンドシェイクの一連の動作です。

7.5.2. マイコンボードからのメッセージの送信

マイコンボード側から、s コマンドでメッセージを送信する過程です。

```
>command=s
send text message to 192.168.0.97 port 20000
TCP data packet (40 bytes) (flag = ACK PSH ) send to 192.168.0.97:20000
TCP frame received from 192.168.0.97:20000 Flag = ACK PSH , 40 bytes data
>OK
TCP flag packet ACK send to 192.168.0.97:20000
```



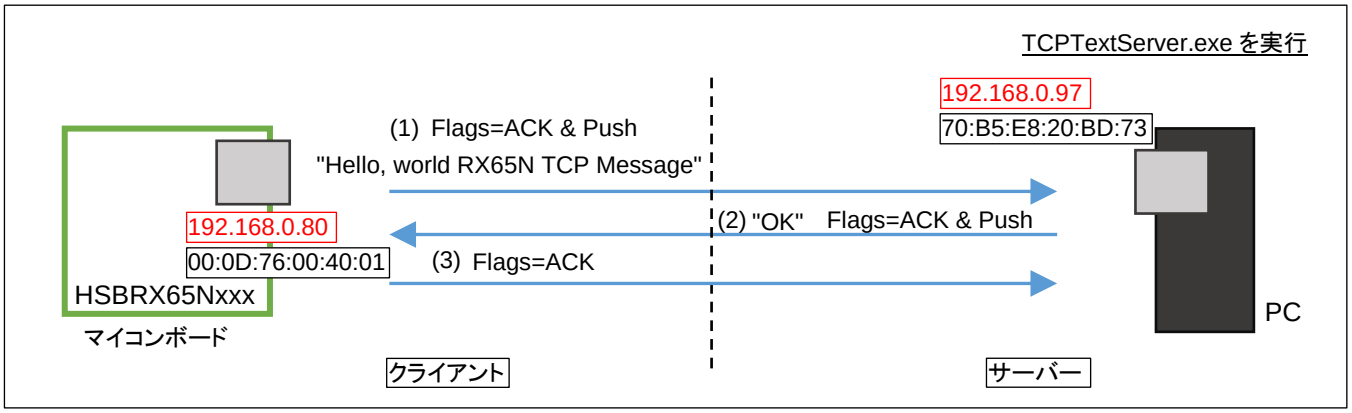
ここでは、

(1)マイコンボードから PC に対してメッセージの送信

(2)PC からマイコンボードに受け取った旨の応答及び返信メッセージの送信

(3)マイコンボードから PC に受け取った旨の応答

の 3 回のパケットのやり取りが行われます。



(1)ACK と Push フラグとデータを送る

(2)ACK と Push フラグとデータを送る

（※TCP 通信という観点ではここで ACK フラグのみを送って(1)(2)で一連の動作を構成する形でも問題ありません）

(3)ACK を送る

というのが一連の動作です。

－(1)のマイコンボードから PC の通信パケット例－

・Ethernet ヘッダ(14)

70	B5	E8	20	BD	73	00	0D	76	00	40	01	08	00
宛先 MAC アドレス						送信元 MAC アドレス						Ethernet type (IPv4)	

・IP ヘッダ(20)

45	00	00	50	00	02	40	00	80	06	78	A4	C0	A8	00	50
IPv4 ヘッダ 20B	Service type	Total Length 80 バイト		ID		Fragment		TTL	Protocol TCP	Checksum		送信元 IP アドレス (192.168.0.80)			

C0	A8	00	61
送信先 IP アドレス (192.168.0.97)			

・TCP ヘッダ(20)

C3	50	4E	20	00	00	23	19	65	14	0F	04	50	18	05	B4
Source port (50000)		Destination Port (20000)		Sequence Number (*1)				Acknowledgment Number (*2)				Headersize + Flags		Window 1460	

2A	3F	00	00
Checksum		Urgent Pointer	

・TCP データ(40)

68	65	6C	6C	6F	2C	77	6F	72	6C	64	20	52	58	36	35
'h'	'e'	'l'	'l'	'o'	','	'w'	'o'	'r'	'l'	'd'	' '	'R'	'X'	'6'	'5'

4E	20	54	43	50	20	6D	65	73	73	61	67	65	20	20	20
'N'	' '	'T'	'C'	'P'	' '	'm'	'e'	's'	's'	'a'	'g'	'e'	' '	' '	' '

20	20	20	20	20	20	20	20
' '	' '	' '	' '	' '	' '	' '	' '

(*1)シーケンス番号 Sequence Number は、前のパケット（ここではハンドシェイク）から引き継がれます

(*2)応答確認番号 Acknowledgment Number も、前のパケット（ここではハンドシェイク）から引き継がれます

TCP は、接続（セッションの確立）後に通信を行いますが、一連のセッションの中でシーケンス番号と応答確認番号は、連続性を持ちます。前回の通信で相手が送ってきた値を保持しておく必要があります。

TCP フラグ

0b0101 0000 0001 1000	b4:Acknowledgment	応答
0b0101 0000 0001 1000	b3:Push	データを送信（バッファリングせずアプリに送る）

－(2)の PC からマイコンボードの通信パケット例－

・Ethernet ヘッダ(14)

00	0D	76	00	40	01	70	B5	E8	20	BD	73	08	00
宛先 MAC アドレス						送信元 MAC アドレス						Ethernet type (IPv4)	

・IP ヘッダ(20)

45	00	00	50	4D	8C	40	00	80	06	00	00	C0	A8	00	61
IPv4 ヘッダ 20B	Service type	Total Length 80 バイト		ID (*1)		Fragment		TTL	Protocol TCP	Checksum		送信元 IP アドレス (192.168.0.97)			

C0	A8	00	50
送信先 IP アドレス (192.168.0.80)			

・TCP ヘッダ(24)

4E	20	C3	50	65	14	0F	04	00	00	23	41	50	18	FF	48
Source port (20000)		Destination Port (50000)		Sequence Number (*2)				Acknowledgment Number (*3)				Headersize + Flags		Window 65352	

82	44	00	00
Checksum		Urgent Pointer	

・TCP データ(40)

20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
'O'	'K'	''	''	''	''	''	''	''	''	''	''	''	''	''	''

20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
''	''	''	''	''	''	''	''	''	''	''	''	''	''	''	''

20	20	20	20	20	20	20	20
''	''	''	''	''	''	''	''

(*1)IP ヘッダ内の ID はパケットが分割されて届いた場合の再構成などに使用されます
PC 側のプログラムで埋め込まれる値は前回の値+1 となっていますが、プログラムでは ID の指定はしていません
（一意な値になっていれば問題ないと思います）
(*2)シーケンス番号 Sequence Number は、前のパケットから引き継がれます
(*3)応答確認番号 Acknowledgment Number は、マイコンボードが送ったシーケンス番号+送信バイト数が設定されます

TCP フラグ

0b0101 0000 0001 1000	b4:Acknowledgment	応答
0b0101 0000 0001 1000	b3:Push	データを送信（バッファリングせずアプリに送る）

－(3)のマイコンボードから PC の通信パケット例－

・Ethernet ヘッダ(14)

70	B5	E8	20	BD	73	00	0D	76	00	40	01	08	00
宛先 MAC アドレス						送信元 MAC アドレス						Ethernet type (IPv4)	

・IP ヘッダ(20)

45	00	00	28	00	03	40	00	80	06	78	CB	C0	A8	00	50
IPv4 ヘッダ 20B	Service type	Total Length 40 バイト		ID		Fragment		TTL	Protocol TCP	Checksum		送信元 IP アドレス (192.168.0.80)			

C0	A8	00	61
送信先 IP アドレス (192.168.0.97)			

・TCP ヘッダ(20)

C3	50	4E	20	00	00	23	41	65	14	0F	2C	50	10	05	B4
Source port (50000)		Destination Port (20000)		Sequence Number (*1)				Acknowledgment Number (*2)				Headersize + Flags		Window 1460	

7F	2C	00	00
Checksum		Urgent Pointer	

・パディング(6)　　…Ethernet フレーム最低 60 バイトのためパディングデータ追加

00	00	00	00	00	00

(*2)シーケンス番号 Sequence Number は、前のパケットから引き継がれます
（＝前回相手が送ってきた応答確認番号）
(*3)応答確認番号 Acknowledgment Number は、PC が送ったシーケンス番号+送信バイト数を設定します

TCP フラグ

0b0101 0000 0001 0000	b4:Acknowledgment	応答

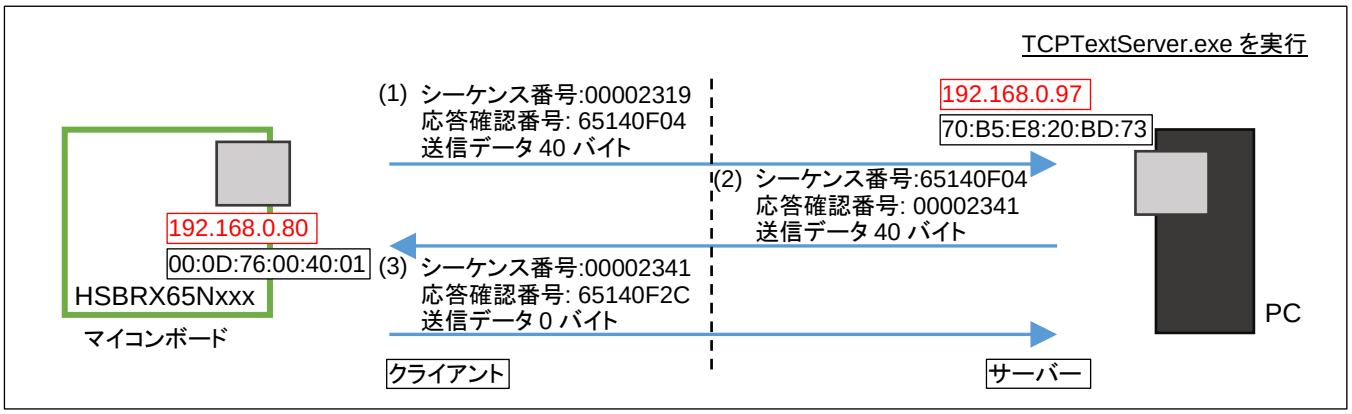
一連のデータのやり取りのパケットは上記の様になります。

TCP では、データを受け取った際、相手に ACK（Acknowledgment フラグ）を返送するのが決まりなので、データ送信時、最低 2 回のやり取りとなります。

ここでは、(2)のタイミングで PC からマイコンボードにデータを送っているので、3 回 1 まとまりのやり取りとなります。

※(2)のタイミングで、データを push せずに Acknowledgment フラグを返送して一連のやり取りを終了させる形でも問題ありません。本プログラムでは、サーバ動作時、PC とマイコンボードのプログラムでメッセージを返送する様にしているので、3 回 1 まとまりのやり取りとなっています。

TCP のプログラムで、ポイントとなるのが、シーケンス番号と応答確認番号です。



	(1)で送信（マイコンボード）	(2)で送信(PC)	(3)で送信（マイコンボード）
シーケンス番号	00002319	65140F04	00002341
応答確認番号	65140F04	00002341 →受信したシーケンス番号+ 受信データバイト数(0x28)	65140F2C →受信したシーケンス番号+ 受信データバイト数(0x28)

応答確認番号は、相手が送ってきたシーケンス番号に受信データバイト数を加算した値を送り返します。シーケンス番号は、シーケンス番号＋送信バイト数（＝相手が送ってきた応答確認番号と同じになります）を設定します。

※TCP は、送信したパケットの順番と受信したパケットの順番が入れ替わっても、受信側でパケットを並び替えできるように、この様なデータの順番が判る様な仕組みを導入しています

7.5.3. 切断時（4 ウェイハンドシェイク）

マイコンボード側では、d コマンドで接続を終了する過程です。

```
>command=d
disconnect
TCP flag packet ACK FIN send to 192.168.0.97:20000
TCP handshake ACK flag received OK.
TCP handshake ACK & FIN flag received OK.
TCP flag packet ACK send to 192.168.0.97:20000
-- CONNECTION CLOSED --
```

マイコンボードから d コマンドで接続を終了する過程ですが、

(1)マイコンボード(192.168.0.80)→PC(192.168.0.97)

(2)PC(192.168.0.97)→マイコンボード(192.168.0.80)

(3)PC(192.168.0.97)→マイコンボード(192.168.0.80)

(4)マイコンボード(192.168.0.80)→PC(192.168.0.97)

の合計 4 回のデータが送信されています。（そのため、4 ウェイハンドシェイクと呼ばれます）

－(1)のマイコンボードから PC の通信パケット例－

・Ethernet ヘッダ(14)

70	B5	E8	20	BD	73	00	0D	76	00	40	01	08	00
宛先 MAC アドレス						送信元 MAC アドレス						Ethernet type (IPv4)	

・IP ヘッダ(20)

45	00	00	28	00	04	40	00	80	06	78	CA	C0	A8	00	50
IPv4 ヘッダ 20B	Service type	Total Length 40 バイト		ID		Fragment		TTL	Protocol TCP	Checksum		送信元 IP アドレス (192.168.0.80)			

C0	A8	00	61
送信先 IP アドレス (192.168.0.97)			

・TCP ヘッダ(20)

C3	50	4E	20	00	00	23	41	65	14	0F	2C	50	11	05	B4
Source port (50000)		Destination Port (20000)		Sequence Number				Acknowledgment Number				Headersize + Flags		Window 1460	

7F	2B	00	00
Checksum		Urgent Pointer	

・パディング(6)　　…Ethernet フレーム最低 60 バイトのためパディングデータ追加

00	00	00	00	00	00

TCP フラグ

0b0101 0000 0001 0001	b4:Acknowledgment	応答
0b0101 0000 0001 0001	b0:Fin	接続の終了

－(3)の PC からマイコンボードの通信パケット例－

・Ethernet ヘッダ(14)

00	0D	76	00	40	01	70	B5	E8	20	BD	73	08	00
宛先 MAC アドレス						送信元 MAC アドレス						Ethernet type (IPv4)	

・IP ヘッダ(20)

45	00	00	28	4D	8D	40	00	80	06	00	00	C0	A8	00	61
IPv4 ヘッダ 20B	Service type	Total Length 40 バイト		ID		Fragment		TTL	Protocol TCP	Checksum		送信元 IP アドレス (192.168.0.97)			

C0	A8	00	50
送信先 IP アドレス (192.168.0.80)			

・TCP ヘッダ(20)

4E	20	C3	50	65	14	0F	2C	00	00	23	42	50	10	FF	48
Source port (20000)		Destination Port (50000)		Sequence Number				Acknowledgment Number				Headersize + Flags		Window 65352	

82	1C	00	00
Checksum		Urgent Pointer	

・パディング(6)　　…Ethernet フレーム最低 60 バイトのためパディングデータ追加

00	00	00	00	00	00

TCP フラグ

0b0101 0000 0001 0000	b4:Acknowledgment	応答

－(3)の PC からマイコンボードの通信パケット例－

・Ethernet ヘッダ(14)

00	0D	76	00	40	01	70	B5	E8	20	BD	73	08	00
宛先 MAC アドレス						送信元 MAC アドレス						Ethernet type (IPv4)	

・IP ヘッダ(20)

45	00	00	28	4D	8E	40	00	80	06	00	00	C0	A8	00	61
IPv4 ヘッダ 20B	Service type	Total Length 40 バイト		ID		Fragment		TTL	Protocol TCP	Checksum		送信元 IP アドレス (192.168.0.97)			

C0	A8	00	50
送信先 IP アドレス (192.168.0.80)			

・TCP ヘッダ(20)

4E	20	C3	50	65	14	0F	2C	00	00	23	42	50	11	FF	48
Source port (20000)		Destination Port (50000)		Sequence Number				Acknowledgment Number				Headersize + Flags		Window 65352	

82	1C	00	00
Checksum		Urgent Pointer	

・パディング(6)　　…Ethernet フレーム最低 60 バイトのためパディングデータ追加

00	00	00	00	00	00

TCP フラグ

0b0101 0000 0001 0001	b4:Acknowledgment	応答
0b0101 0000 0001 0001	b0:Fin	接続の終了

ー(4)のマイコンボードから PC の通信パケット例ー

・Ethernet ヘッダ(14)

70	B5	E8	20	BD	73	00	0D	76	00	40	01	08	00
宛先 MAC アドレス						送信元 MAC アドレス						Ethernet type (IPv4)	

・IP ヘッダ(20)

45	00	00	28	00	05	40	00	80	06	78	C9	C0	A8	00	50
IPv4 ヘッダ 20B	Service type	Total Length 40 バイト		ID		Fragment		TTL	Protocol TCP	Checksum		送信元 IP アドレス (192.168.0.80)			

C0	A8	00	61
送信先 IP アドレス (192.168.0.97)			

・TCP ヘッダ(20)

C3	50	4E	20	00	00	23	42	65	14	0F	2D	50	10	05	B4
Source port (50000)		Destination Port (20000)		Sequence Number				Acknowledgment Number				Headersize + Flags		Window 1460	

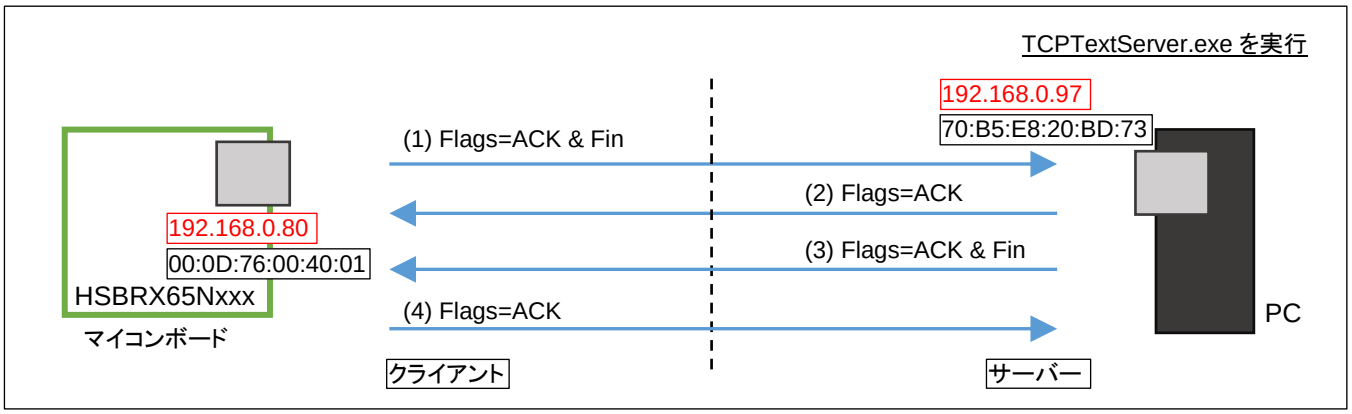
7F	2A	00	00
Checksum		Urgent Pointer	

・パディング(6)　　…Ethernet フレーム最低 60 バイトのためパディングデータ追加

00	00	00	00	00	00

TCP フラグ

0b0101 0000 0001 0000	b4:Acknowledgment	応答



(1)ACK と FIN フラグを送る（切断をリクエストする側）
(2)ACK フラグを送る（切断を受領する側）
(3)ACK と FIN フラグを送る（切断を受領する側）
(4)ACK フラグを送る（切断をリクエストする側）

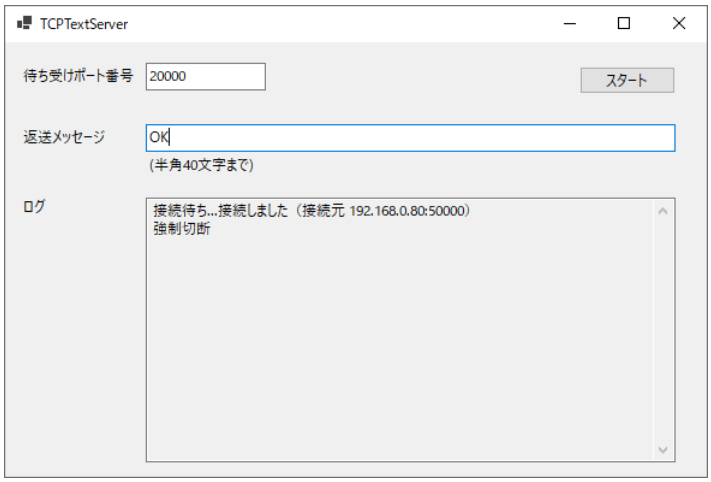
切断時のやり取りは、TCP フラグを合計 4 回やり取りします。

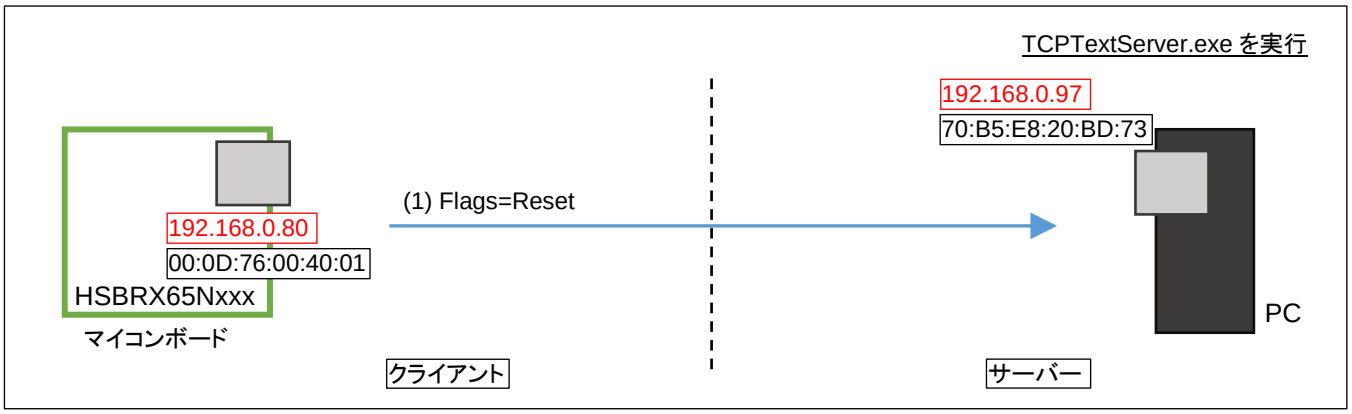
TCP は UDP に比べると、データを送る前に接続関係を構築したり、データを受け取った際に ACK フラグを返したり、シーケンス番号や応答確認番号を適切に設定したりしなければならないので多少面倒です。
面倒な分、データが届かなかったときに再送したり、パケットが順番通りに届かなかった場合でも正しい順番に並び替えたりする仕組みが組み込まれていますので、データを送信する際の信頼性は TCP の方が優れています。

7.5.4. 切断時（強制切断）

マイコンボード側では、D コマンドで接続を終了する過程です。

```
>command=D
disconnect(force)
TCP flag packet RST send to 192.168.0.97:20000
```





(1)Reset フラグを送る

片方のホストから、Reset フラグを送るだけで、その後のやり取りはありません。

－(1)のマイコンボードから PC の通信パケット例－

・Ethernet ヘッダ(14)

70	B5	E8	20	BD	73	00	0D	76	00	40	01	08	00
宛先 MAC アドレス						送信元 MAC アドレス						Ethernet type (IPv4)	

・IP ヘッダ(20)

45	00	00	28	00	02	40	00	80	06	78	CC	C0	A8	00	50
IPv4 ヘッダ 20B	Service type	Total Length 40 バイト		ID		Fragment		TTL	Protocol TCP	Checksum		送信元 IP アドレス (192.168.0.80)			

C0	A8	00	61
送信先 IP アドレス (192.168.0.97)			

・TCP ヘッダ(20)

C3	50	4E	20	00	00	23	19	1C	F1	21	CE	50	04	05	B4
Source port (50000)		Destination Port (20000)		Sequence Number				Acknowledgment Number				Headersize + Flags		Window 1460	

B4	E1	00	00
Checksum		Urgent Pointer	

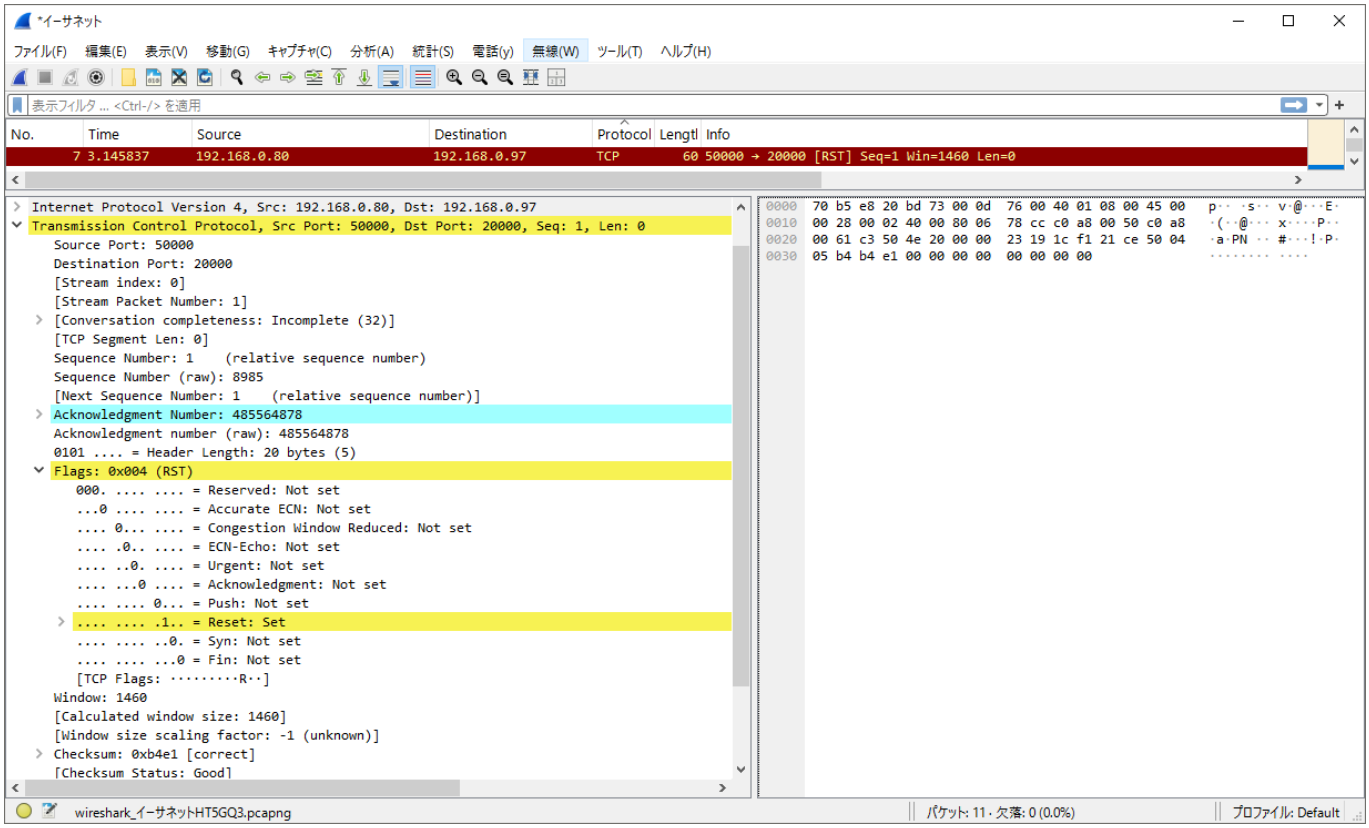
・パディング(6)　　…Ethernet フレーム最低 60 バイトのためパディングデータ追加

00	00	00	00	00	00

TCP フラグ

0b0101 0000 0000 0100	b2:Reset	強制切断

－Wireshark でのパケットダンプ例－



4 ウェイハンドシェイクとは異なり、1 回の通信で終了しています。

Reset フラグ付きのデータを受信した側(TCPTextServer.exe)は、例外処理によりテキストボックスに「強制切断」というメッセージを表示させています。

取扱説明書改定記録

バージョン	発行日	ページ	改定内容
REV.1.0.0.0	2026.6.5	—	初版発行

お問合せ窓口

最新情報については弊社ホームページをご活用ください。

ご不明点は弊社サポート窓口までお問合せください。

株式会社 北斗電子

〒060-0042 札幌市中央区大通西 16 丁目 3 番地 7

TEL 011-640-8800　FAX 011-640-8801

e-mail：support@hokutodenshi.co.jp (サポート用)、order@hokutodenshi.co.jp (ご注文用)

URL:https://www.hokutodenshi.co.jp

ルネサス エレクトロニクス RX マイコン搭載
HSB シリーズマイコンボード 評価キット

Ethernet スターターキット RX65N
取扱説明書(2)

株式会社 北斗電子

 2026 年 6 月 5 日改訂 REV.1.0.0.0 (260605)