Ethernet スタータキット RX65N

取扱説明書(1)

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 1

安全上のご注意 2

特徴 4

「Ethernet スタータキット RX65N」製品構成 5

「Ethernet スタータキット RX65N-100」製品内容 6

「Ethernet スタータキット RX65N-144」製品内容 6

「Ethernet スタータキット RX65N-176」製品内容 7

RX65N マイコンボード概要 8

USB-ADAPTER-RX14 概要 8

サンプルプログラム CD 9

マイコンボード向けプロジェクト 11

PC 向けアプリに関して 11

マニュアルに関して 13

1.　マイコン搭載の Ethernet 機能に関して 15

2.　Ethernet 通信の概要 17

2.1.　Ethernet(100BASE-TX)の接続形態 17

2.2.　MAC アドレス 17

2.3.　Ethernet フレーム 19

3.　ネットワーク接続時の注意点 21

3.1.　RX65N マイコンボードをネットワークに接続するにあたり 21

3.2.　Ethernet 通信のモニタに関して 21

3.3.　PC 上で動作させるサンプルプログラムに関して 21

4.　Ethernet フレームを送ってみる 22

4.1.　サンプルプロジェクトの開き方 22

4.1.1.　CS+の場合 22

4.1.2.　e2studio の場合 23

4.2.　マイコンボードにプログラムを書き込む方法 26

4.2.1.　ルネサス製エミュレータを使用する場合(CS+) 26

4.2.1.　ルネサス製エミュレータを使用する場合(e2studio) 29

4.2.2.　USB-ADAPTER-RX14 を使用する場合 32

4.2.3.　ルネサス製エミュレータを使用する場合(RenesasFlashProgrammer) 37

4.3.　MagicPacket(WOL)の送受信 39

4.3.1.　MagicPacket の受信 40

4.3.2.　MagicPacket の送信 45

4.3.3. RX65N マイコンボードを 2 台使用した MagicPacket の送受信 49
4.3.4. RX65N マイコンボードと PC の MagicPacket の相違点 52
4.4. PC に対しての MagicPacket(WOL)の送信 56
4.5. PC で純粋な MagicPacket を送受信する方法[参考] 61
4.6. RX65N_ETHER_NOAPI プロジェクトに関して 65
5. ping の応答 68
5.1. IP(InternetProtocol)ネットワークに関して 68
5.2. ARP(AddressResolutionProtocol)とは？ 73
5.3. ping がどうやって実現されているか 77
5.4. マイコンボードでの ping 動作 83
5.4.1. ping 応答（ping リプライ） 85
5.4.2. ping の送信（ping リクエスト） 92
5.5. IP ヘッダ内のチェックサムフィールドを Wireshark でモニタした場合 96
取扱説明書改定記録 99
お問合せ窓口 99

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。

特徴

本製品は、ルネサスエレクトロニクス製 RX65N 搭載マイコンボードで Ethernet の基礎を学ぶためのキットです。

Ethernet に関して全くなじみがなく、初めて Ethernet を取り扱いたいという方に向けた、ゼロから Ethernet のプログラムを学べるキットとして構成されています。

・Ethernet フレームの送受信（MagicPacket(WakeON LAN)）
・ping（ARP リクエスト、ARP リプライ）
・UDP を使用した通信
・TCP を使用した通信
・LWIP のプロトコルスタックの使用（UDP, TCP)

といった内容を、1 ステップずつ学べるようになっています。

Ethernet という性質上、マイコンボードと通信を行う通信相手が必要ですが、本キットでは

・マイコンボード←→PC
・マイコンボード←→マイコンボード （※RX65N 搭載マイコンボードがもう 1 台必要）

両方の組み合わせで使用できるようになっています。

最初は本キットと、PC で通信を行いながら Ethernet の動作を学習。必要になった時点でもう 1 台マイコンボードをご用意頂き、マイコンボード同士の Ethernet 通信を確認するといった使い方ができます。

「Ethernet スタータキット RX65N」製品構成

Ethernet スタータキット RX65N は、製品としては 3 種類のラインナップとなります。

製品名	マイコンボード型名	備考
Ethernet スタータキット RX65N-100	HSBRX65N100A	100pin マイコンチップ搭載
Ethernet スタータキット RX65N-144	HSBRX65N144A	144pin マイコンチップ搭載
Ethernet スタータキット RX65N-176	HSBRX65N176	176pin マイコンチップ搭載

搭載しているマイコンのピン数の違いで、キットの学習内容としてはどの製品でも相違ありません。

プログラムの変更、コンパイル、ビルドを行うためには、開発環境として、ルネサスエレクトロニクス製 CS+(CS+forCC)が別途必要になります。(CS+は、ルネサスエレクトロニクス Web よりダウンロードできます)

開発環境としては、ルネサスエレクトロニクス製 e2studio も使用可能です(CS+向けのプロジェクトを e2studio で読み込んで使用可能です。)

マイコンボードに対するプログラムの書き込みは、ルネサスエレクトロニクス製 RenesasFlashProgrammer(Ver3.x)が必要となります。(RenesasFlashProgrammer は、ルネサスエレクトロニクス Web よりダウンロードできます)

マイコンのプログラムのデバッグ(ステップ実行や、ブレーク、レジスタ値や変数のモニタ)を行う場合、ルネサスエレクトロニクス製の E1, E2, E2Lite, E20 いずれかのエミュレータが必要です。プログラムのデバッグを行う際は、エミュレータを別途ご用意ください。

PC 上で動作するプログラムは、C#, .NET10 で作成されています。プログラムを変更したい場合は、VisualStudio2026 のインストールが必要です。(プログラムの実行に関しては、.NET10 のランタイムのインストールが必要、詳細は後述。)

「Ethernet スタータキット RX65N-100」製品内容

本製品は、下記の品が同梱されております。ご使用前に必ず内容物をご確認ください。

- ・HSBRX65N100A（マイコンボード） ..... 1 枚
- ・USB-ADAPTER-RX14 ..... 1 枚
- ・USB ケーブル(USB-A - miniB) ..... 1 本
- ・サンプルプログラム CD ..... 1 枚
- ・DC 電源ケーブル ..... 1 本
  ※2P コネクタ片側圧着済み 30cm(JST)
- ・CAN 通信ケーブル ..... 1 本
  ※4P コネクタ片側圧着済み 50cm(JST)
- ・マイコンボード回路図 ..... 1 部

「Ethernet スタータキット RX65N-144」製品内容

本製品は、下記の品が同梱されております。ご使用前に必ず内容物をご確認ください。

- ・HSBRX65N144A（マイコンボード） ..... 1 枚
- ・USB-ADAPTER-RX14 ..... 1 枚
- ・USB ケーブル(USB-A - miniB) ..... 1 本
- ・サンプルプログラム CD ..... 1 枚
- ・DC 電源ケーブル ..... 1 本
  ※2P コネクタ片側圧着済み 30cm(JST)
- ・CAN 通信ケーブル ..... 1 本
  ※4P コネクタ片側圧着済み 50cm(JST)
- ・マイコンボード回路図 ..... 1 部

「Ethernet スタータキット RX65N-176」製品内容

本製品は、下記の品が同梱されております。ご使用前に必ず内容物をご確認ください。

- ・HSBRX65N176（マイコンボード）.....1 枚
- ・USB-ADAPTER-RX14.....1 枚
- ・USB ケーブル(USB-A - miniB).....1 本
- ・サンプルプログラム CD.....1 枚
- ・DC 電源ケーブル.....1 本
  ※2P コネクタ片側圧着済み 30cm(JST)
- ・CAN 通信ケーブル .....1 本
  ※4P コネクタ片側圧着済み 50cm(JST)
- ・マイコンボード回路図 .....1 部

RX65N マイコンボード概要

- RX65N（QFP-100/144/176 ピン）搭載
- エミュレータインタフェース(14P)搭載（E1/E2/E2Lite/E20 向け）
- 100BASE-TX コネクタ(RJ45)搭載, PHY-LSI 搭載
- CAN インタフェース(4P)搭載
- USB-fuction(USB-miniB)搭載
- USB-Host(USB-A)搭載
- リセットスイッチ搭載
- 評価用スイッチ搭載（Push-SW）
- 評価用 LED 搭載（1 つ）
- 24MHz 水晶振動子搭載（メインクロック）
- 32.768kHz 水晶振動子搭載（サブクロック）

「Ethernet スタータキット RX65N」には、100-BASE-TX(RJ45)コネクタが搭載されたマイコンボード（上記）が付属します。LAN ケーブルは別途ご用意ください。

マイコンボードの詳細は、マイコンボードの取扱説明書を参照願います。

USB-ADAPTER-RX14 概要

USB-ADAPTER-RX14 は、PC の USB-A コネクタに接続して、マイコンボードから PC に対して情報を送信したり、PC からマイコンボードに指示を出す用途で使用します。

PC 上では、仮想 COM ポートとして認識しますので、teraterm 等の端末ソフトと組み合わせて使用します。
（一般的な USB-Serial 変換機器と同じものです。）

また、マイコンボードに対してのプログラムの書き込みアダプタとしても機能します。

マイコンボード上では、SCI1 を使って、UAB-ADAPTER-RX14 と通信します。

USB-ADAPTER-RX14 の詳細に関しては、USB-ADAPTER-RX14 の取扱説明書を参照してください。

サンプルプログラム CD

製品に付属しているサンプルプログラム CD の内容を下記に示します。

フォルダ		内容
BIN¥	144_100¥	HSBRX65N144A,HSBRX65N100A 向け、mot ファイル格納フォルダ
	176¥	HSBRX65N176 向け、 mot ファイル格納フォルダ
manual¥		マニュアル格納フォルダ
PC_APPLI¥		PC アプリ格納フォルダ
SOURCE¥	144_100¥	HSBRX65N144A,HSBRX65N100A 向け、 CS+プロジェクト格納フォルダ
	176¥	HSBRX65N176 向け、 CS+プロジェクト格納フォルダ
VISUAL_STUDIO_REPOS¥		VisualStudio ソース格納フォルダ
DEMOS¥	CAN_MULTI¥	CAN マルチネットワークボードと組み合わせて、CAN と Etherent を使用するデモプログラム(*1)
	TCP_DEMO¥	マイコンボードで TCP サーバを動かし、PC からマイコンボードを制御するデモプログラム(*1)

(*1)Web でバイナリを公開しているデモプログラムのソース
※SOURCE 以下に含まれる mot ファイルと、BIN 以下の mot ファイルは同じものです

BIN はマイコンボード向けの mot ファイルが格納されているフォルダです。取扱説明書に従い動作を確認したい場合等に、BIN 以下の mot ファイルをマイコンボードに書き込んでください。BIN 以下は、144_100（144 ピンと 100 ピンのマイコンボード用）と 176（176 ピンのマイコンボード用）にフォルダが分かれています。
144 ピン(HSBRX65N144A)と 100 ピン(HSBRX65N100A)は、同じ mot ファイルで動作します。176 ピン(HSBRX65N176)は、書き込む mot ファイルが別になります。
（144,100 ピンでは、SCI の TX,RX が P26,P30 になります。176 ピンでは、PF0,PF2 になります。SCI の端子が異なるのみで、他に相違点はありません。）

manual 以下には、PDF 形式のマニュアルが格納されています。マニュアルは、当社 web ページで最新版が公開されていますので、必要に応じて web からダウンロードしてください。

PC_APPLI には、マイコンの Ether 機能の動作確認用の PC 側で動作するアプリケーションプログラムが格納されています。.NET10 で開発されていますので、実行には.NET10 のランタイムが必要になります。詳細は、4.3.1 を参照してください。

SOURCE 以下には、マイコンボード用のプログラムのソース、CS+のプロジェクトが格納されています。マイコンボード様のプログラムの中身に関しては、ソフトウェア編マニュアルに記載されています。

VISUAL_STUDIO_REPOS 以下には、PC 用のアプリケーションのソースが格納されています。VisualStudio2026/C#/.NET10 の環境でプログラムを作成しています。ソースを修正してビルドする場合、VisualStudio2026 と C#, .NET10 の開発環境（VisualStudio インストール時に、C#デスクトップ開発環境を有効化してインスト―ル）してください。

マイコンボード向けプロジェクト

サンプルプログラム CD の SOURCE 以下に格納されているマイコンボード向けプロジェクト一覧です。

表の上から順に、マイコンの Ethernet 機能を学んでいく内容になっています。
SOURCE 以下には、144_100 と 176（マイコンのピン数に応じて）フォルダが分かれていますので、使用するマイコンボードのピン数に応じて、144_100 か 176 以下のフォルダを PC にコピーして、必要に応じてソースの変更、ビルドしてください。

プロジェクト名	内容
RX65N_MAGIC_PACKET	単純な Ethernet フレームである MagicPacket（WOL パケット）の送受信
RX65N_MAGIC_PACKET2	PC で受信可能な MagicPacket の送信 Ethernet フレームの受信データの取り扱い
RX65N_ETHER_NOAPI	r_ether_rx コンポーネントを使用しない Ethernet 通信
RX65N_PING	ARP, ICMP プロトコルの取り扱い
RX65N_UDP	UDP 通信
RX65N_TCP	TCP 通信
RX65N_LWIP	LWIP プロトコルスタックを使用した、TCP/UDP 通信
RX65N_UART_ETHER_UDP	UDP を使用したファイル転送（LWIP プロトコルスタック使用）
RX65N_UART_ETHER_TCP_DMAC	TCP を使用したファイル転送（LWIP プロトコルスタック使用）

PC 向けアプリに関して

サンプルプログラム CD の PC_APPLI 以下に、下記の exe ファイルが格納されています。実行する場合は、.NET10 のラインタイムのインストール後に、PC_APPLI 以下の exe ファイルを実行してください。

プログラムを修正したい場合は、VISUAL_STUDIO_REPOS 以下に、下記プロジェクト名のフォルダの下に、VisualStudio のプロジェクトが格納されていますので、VisualStudio(2026)で開いて、必要に応じてソースの変更、ビルドしてください。

プロジェクト名	内容
MagicPacketSend	MagicPacket（WOL パケット）の送信（IP/UDP 使用）
MagicPacketReceive	MagicPacket（WOL パケット）の受信（IP/UDP 使用）
MagicPacketSendRaw	MagicPacket（WOL パケット）の送信（純粋な MagicPacket）
MagicPacketReceiveRaw	MagicPacket（WOL パケット）の受信（純粋な MagicPacket）
UDPTextServer	UDP 通信（サーバ動作）
UDPTextClient	UDP 通信（クライアント動作）
TCPTextServer	TCP 通信（サーバ動作）
TCPTextClient	TCP 通信（クライアント動作）

プロジェクト名	内容
UDPFileReceive	UDP を使用したファイル転送（受信側）
UDPFileSend	UDP を使用したファイル転送（送信側）
TCPFileReceive	TCP を使用したファイル転送（受信側）
TCPFileSend	TCP を使用したファイル転送（送信側）

マニュアルに関して

ファイル名	タイトル
ETHERNET_STARTER_KIT_RX65N_1_REV_x_x_x_x.pdf	Ethernet スタータキット RX65N 取扱説明書(1) ※本書
ETHERNET_STARTER_KIT_RX65N_2_REV_x_x_x_x.pdf	Ethernet スタータキット RX65N 取扱説明書(2)
ETHERNET_STARTER_KIT_RX65N_3_REV_x_x_x_x.pdf	Ethernet スタータキット RX65N 取扱説明書(3)
ETHERNET_STARTER_KIT_RX65N_Software_1_REV_x_x_x_x.pdf	Ethernet スタータキット RX65N ソフトウェア編　取扱説明書(1)
ETHERNET_STARTER_KIT_RX65N_Software_2_REV_x_x_x_x.pdf	Ethernet スタータキット RX65N ソフトウェア編　取扱説明書(2)
ETHERNET_STARTER_KIT_RX65N_Software_3_REV_x_x_x_x.pdf	Ethernet スタータキット RX65N ソフトウェア編　取扱説明書(3)
ETHERNET_STARTER_KIT_RX65N_Software_PC_REV_x_x_x_x.pdf	Ethernet スタータキット RX65N PC ソフトウェア編　取扱説明書

※x_x_x_x には、バージョン番号が入ります

「Ethernet スタータキット RX65N　取扱説明書」では、プログラムの動作やネットワークのプロトコルに関して説明しています。「Ethernet スタータキット RX65N　ソフトウェア編　取扱説明書」では、マイコン側で動作するプログラムの構成やソースの解説を行っています。「Ethernet スタータキット RX65N　PC ソフトウェア編　取扱説明書」では、PC 向けのプログラムに関して簡単に説明しています（本キットは、マイコン側の Ethernet を使用するプログラムの学習を主眼にしています。PC のアプリは、マイコンボードの動作確認用に用意しているもので、PC のネットワークプログラミングを習得する事を目的とはしていません。）

タイトル	扱っている内容
Ethernet スタータキット RX65N 取扱説明書(1) ※本書	Ethernet 通信の概要 ネットワーク接続時の注意点 Ethernet フレームを送ってみる ping の応答
Ethernet スタータキット RX65N 取扱説明書(2)	UDP 通信に関して TCP 通信に関して
Ethernet スタータキット RX65N 取扱説明書(3)	LWIP プロトコルスタックの使用 UDP を使用したファイル転送 TCP を使用したファイル転送
Ethernet スタータキット RX65N ソフトウェア編　取扱説明書(1)	RX65N_MAGIC_PACKET プロジェクト RX65N_MAGIC_PACKET2 プロジェクト RX65N_ETHER_NOAPI プロジェクト RX65N_PING プロジェクト
Ethernet スタータキット RX65N ソフトウェア編　取扱説明書(2)	RX65N_UDP プロジェクト RX65N_TCP プロジェクト
Ethernet スタータキット RX65N ソフトウェア編　取扱説明書(3)	RX65N_LWIP プロジェクト RX65N_UART_ETHER_UDP プロジェクト RX65N_UART_ETHER_TCP_DMAC プロジェクト

タイトル	扱っている内容
Ethernet スタータキット RX65N PC ソフトウェア編　取扱説明書	PC 上で動作するソフトウェアの説明

1. マイコン搭載の Ethernet 機能に関して

RX65N は Ethernet 機能が内蔵されたマイコンとなります。

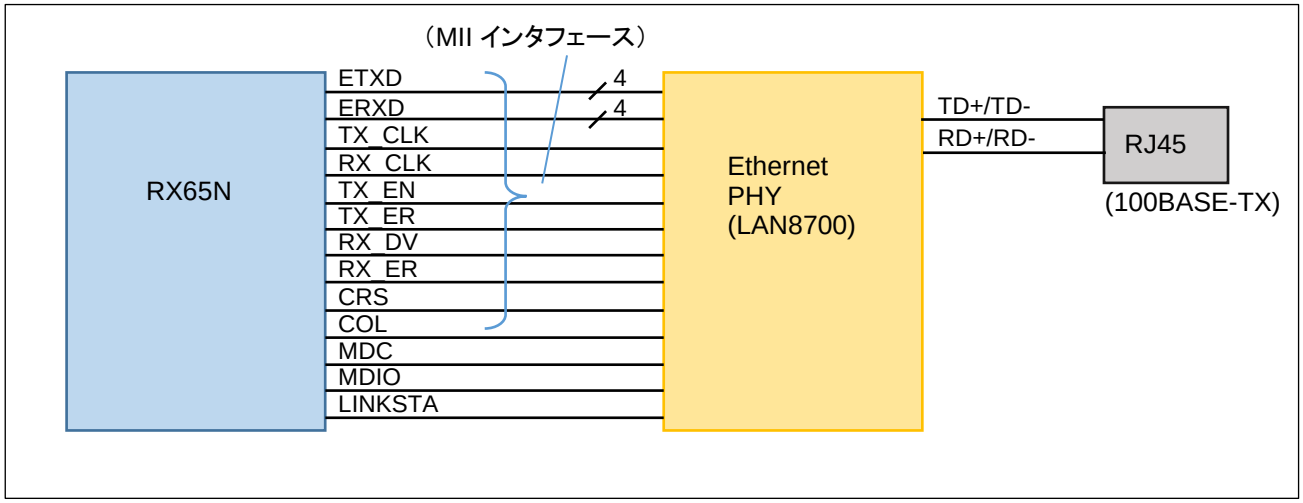


図 1-1　マイコンと LAN コネクタの接続

LAN 側のコネクタ(RJ45)は、100BASE-TX という規格で、電気的には信号を TD+/TD-, RD+, RD-の差動信号で駆動する仕様です。RX65N マイコンには、100BASE-TX の信号を直接やり取りする機能は内蔵されていません。

そこで使用されるのが、マイコンの信号（0-3.3V のデジタル振幅）を 100BASE-TX に電気的に変換を行うチップで、このチップは Ethernet PHY（ファイ：物理的な変換を行う physical といった意味合い）と呼ばれます。当社の RX65N 搭載マイコンボードには、EthernetPHY チップとして LAN8700（Microchip 製）という型番の LSI が搭載されています。

マイコンと、EthernetPHY 間の接続には、2 通りの方法があり、

・MII インタフェース

・RMII インタフェース

のどちらかが使われます。当社の RX65N 搭載マイコンボードでは、MII インタフェースが使われています。

MII インタフェースでは、25Mbps の信号線を送信用(ETXD)と受信用(ERXD)にそれぞれ 4 本使用して、100Mbps の通信(100BASE-TX)を実現します。（その他クロックや制御信号などを合わせると、20 本程度の信号接続となります。）

もう一方の RMII 接続では、50Mbps の信号線を送信用、受信用にそれぞれ 2 本使用する形となります。（ある程度低速で多数の信号線を使うか、高速にして使用する信号線を減らすかの違いです。どちらの接続でも、LAN 側は 100Mbps となります。）

MDC, MDIO は、マイコンから EthernetPHY チップの制御に用いる信号線です。EthernetPHY チップはレジスタ（揮発性の記憶領域＝メモリ）を持っており、レジスタに速度などのパラメータを設定する様になっています。マイコンから、EthernetPHY チップのレジスタにアクセスする事により、EthernetPHY チップの設定を行ったり、状態を把握する事が出来ます。

LINKSTA は、EthernetPHY のその先の RJ45 コネクタに LAN ケーブルがつながり他の機器と接続されたことを検出して、マイコンに知らせるための信号線です。

図 1-1 に示したのが、マイコンと LAN コネクタの物理的な接続形態です。
一方マイコンのソフト的な動作は、図 1-2 の様になります。

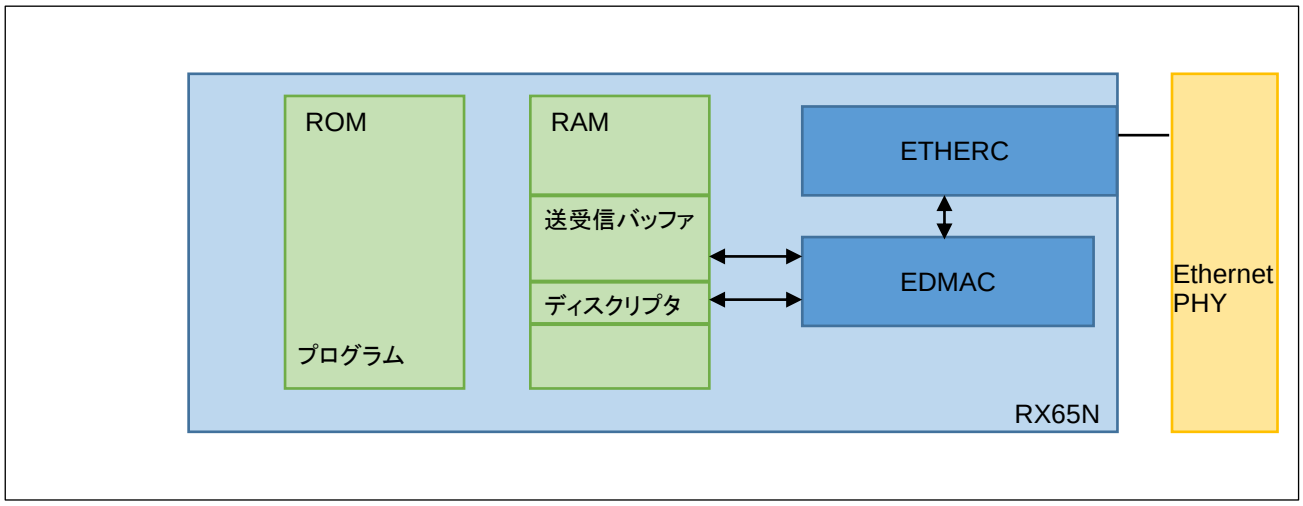


図 1-2 マイコン内部のデータのやり取り

UART(SCI)や SPI のプログラムでは、プログラムで SCI や SPI のデータ送信レジスタに書き込みを行う事でデータが送信され、データ受信レジスタの読み出しを行う事で、外部から受信したデータを得ることが出来ます。

Ethernet 機能の場合は、多少複雑で、外部の EthernetPHY チップと電気的に信号をやり取りをするのは、ETHERC（Ethernet コントローラ）となりますが、外部とのデータをやり取りをするのは EDMAC（Ethernet 専用の DMA コントローラ）となります。EDMAC は RAM 上に確保した送信バッファに格納されているデータを送信し、外部から受信したデータを RAM 上の受信バッファにコピーします。レジスタ経由でデータを送受信する訳ではなく、RAM 上で、直接送受信データを取り扱うのが特徴です。
（UART や SPI では、1 バイトや 4 バイトといった単位でデータを送受信しますが、Ethernet は 1 フレーム最大 1514 バイトのデータを取り扱います。扱うデータサイズが大きいので、レジスタを経由するのではなく、直接 RAM 上のデータにアクセスする方式となっています。）

また、EDMAC に送受信バッファのアドレスを設定する訳ではなく、ディスクリプタ（RAM 上に配置した単なるデータ）に、送受信バッファのアドレスを記載して、EDMAC にはディスクリプタのアドレスを知らせるやり方（送受信バッファにアクセスするのに、ディスクリプタを一旦経由する）というのも、ちょっと変わっています。

－DMA コントローラとは？－

DMA コントローラ(DirectMemoryAccess Controller)は、CPU を介さずに、メモリのデータコピーを行う機能です。EDMAC は Ether 専用の DMA コントローラで、RAM と EDMAC 内のバッファ間のデータコピーを行います。

2. Ethernet 通信の概要

2.1. Ethernet(100BASE-TX)の接続形態

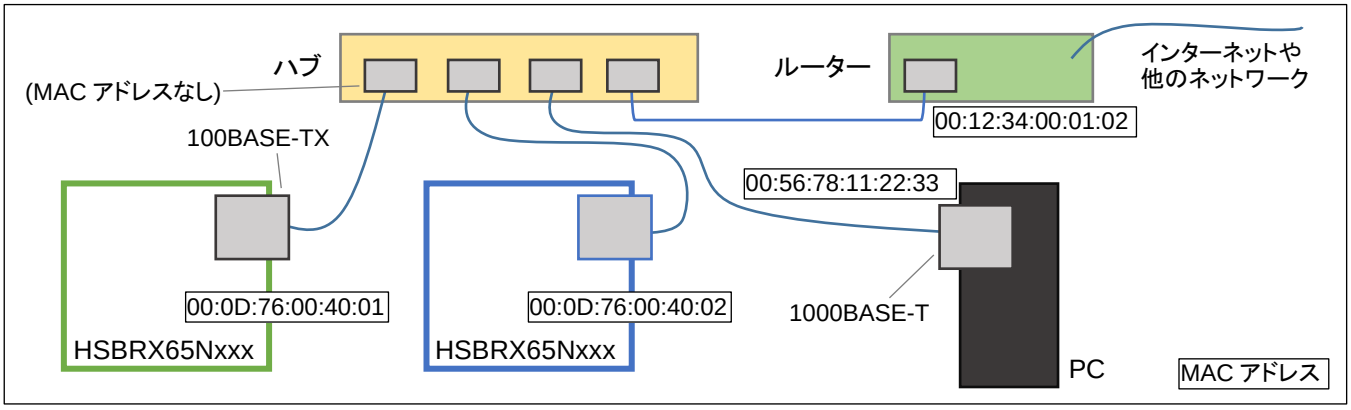


図 2-1 xxBASE-Tx 接続形態

RX65N マイコンボード側の Ethernet インタフェースは、100BASE-TX です。PC の Ethernet コネクタは最近ですと 1000BASE-T（や 2.5GbE, 5GbE のネットワークインタフェース）です。通常は、これらのネットワークをハブで接続する形となります。PC のネットワークインタフェースの規格と、RX65N マイコンボード搭載の 100BASE-TX は規格が異なりますが、相互に接続して通信が可能です。

2.2. MAC アドレス

Ethernet で通信を行う際に、各機器に割り振られている MAC(MediaAccessControl)アドレスを使用して個々の機器を識別します。MAC アドレスは、Ethernet アドレスとも呼ばれます。どちらも同じものです。本書では、MAC アドレスという言い方を使用します。

MAC アドレスは、（同一ネットワーク内に）同じアドレスの機器が存在するのは NG です。基本的には、PC やルータなど、ネットワーク機器には出荷時に一意の MAC アドレスが割り振られており、アドレスが重複しない様になっています。

マイコンボードにおける MAC アドレスですが、主に

・出荷時に一意の MAC アドレスが割り振られている

・マイコンボードを購入したユーザが任意のアドレスを設定する

の 2 通りの方式があります。RX65N マイコンボードでは、後者の方式で、マイコンボードを購入したユーザが任意の MAC アドレスを設定する事となります。

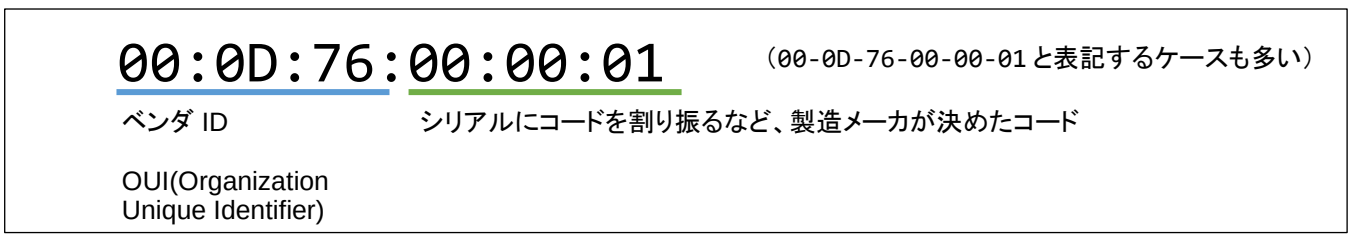


図 2-2 MAC アドレスの構造

MAC アドレスは、48bit で構成されており、8bit ずつコロン(:)やハイフン(-)で区切って表示します。値は、16 進数(0~9, A~F)です。先頭の 24bit がベンダ ID でネットワーク機器を製造するメーカ毎に割り振られた数値です。

（大手のメーカは、複数のベンダ ID を持っていますので、ベンダ ID が異なる＝別メーカの機器とも限りません。）
北斗電子のベンダ ID は、00:0D:76 で、本キットのサンプルプログラムではこのベンダ ID が使用されています。最終的に、マイコンボードを社内ネットワークに接続する場合は、サンプルプログラムのまま北斗電子のベンダ ID を使用するか、御社で取得しているベンダ ID を使用するか、ローカルアドレスとなる 0x02 で始まるベンダ ID（テスト用）を使用する事が推奨です。（基本的には、ネットワーク内に同じ MAC アドレスを持つ機器が居なければ問題はありません。）

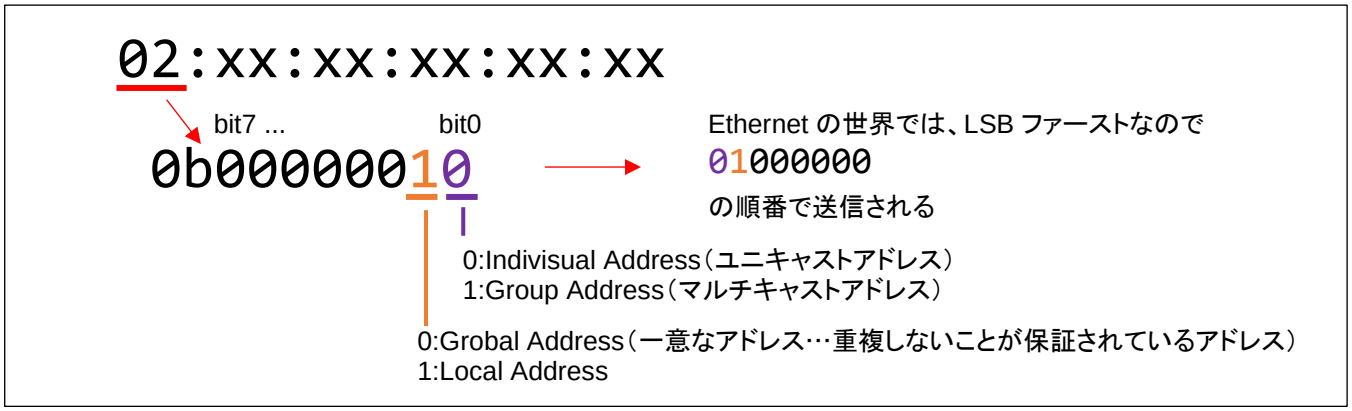


図 2-3 MAC アドレスの構造(2)

MAC アドレスの先頭ビット（この場合は、先頭バイトの bit0）は、1 だとマルチキャスト（一斉送信の意味）になるので、（テストで運用する際でも）0 の設定が良いです。2 ビット目は、基本的には 0 ですが、テスト向けであれば 1（ローカルアドレス）でも可といったところです。

なお、RX65N マイコンを組み込んで自社の製品として販売する場合は、IEEE から取得した御社固有のベンダ ID を設定する必要があります。

（MAC アドレスは、ハードウェアに割り振られているものですが、最近は端末を利用している個人を特定するのを防止する観点で OS レベルで MAC アドレスをランダムに変更して運用するケースもあります。）

この MAC アドレスですが、RX65N の場合はソフトウエアでマイコン内部のレジスタに値を設定するという形となります。出荷時固有の MAC アドレスが与えられている訳ではない点にご注意ください。（任意の値を設定可能ですが、逆に言えば、RX65N ではプログラム作成者がボード毎に MAC アドレスを決めて値を設定する必要があるという事となります。）

2.3. Ethernet フレーム

Ethernet での通信は Ethernet フレーム単位での通信となります。

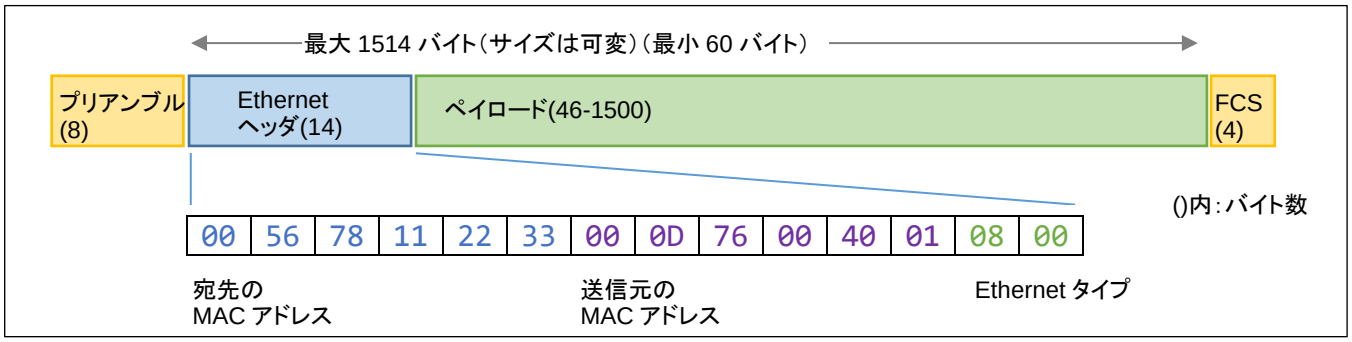


図 2-2 Ethernet II フレームの構造

Ethernet のフレーム（データ）は、プリアンブル、Ethernet ヘッダ、ペイロード（貨物室の意）、FCS（FrameCheckSequence，誤り検出）から構成されています。この中で、プリアンブルと FCS は自動的に付与されるものなので、ユーザが設定可能なのは、Ethernet ヘッダと、ペイロードの部分になります。

※プリアンブルは、データが流れる伝送線路（LAN ケーブルなど）に 0/1/0/1 と変化する信号を送って、コリをほぐす（実際にデータを送る前に温めておく）様な意味合いがあります
※FCS は、送ったデータと受信したデータに変化（データ化け）が無いかどうかを検出する目的で付与されているものです

（ここでは、Ethernet フレームと言う表記としていますが、Ethernet フレーム、Ethernet パケット、Ethernet データなどは、大体同じ意味合いで使用されます。）

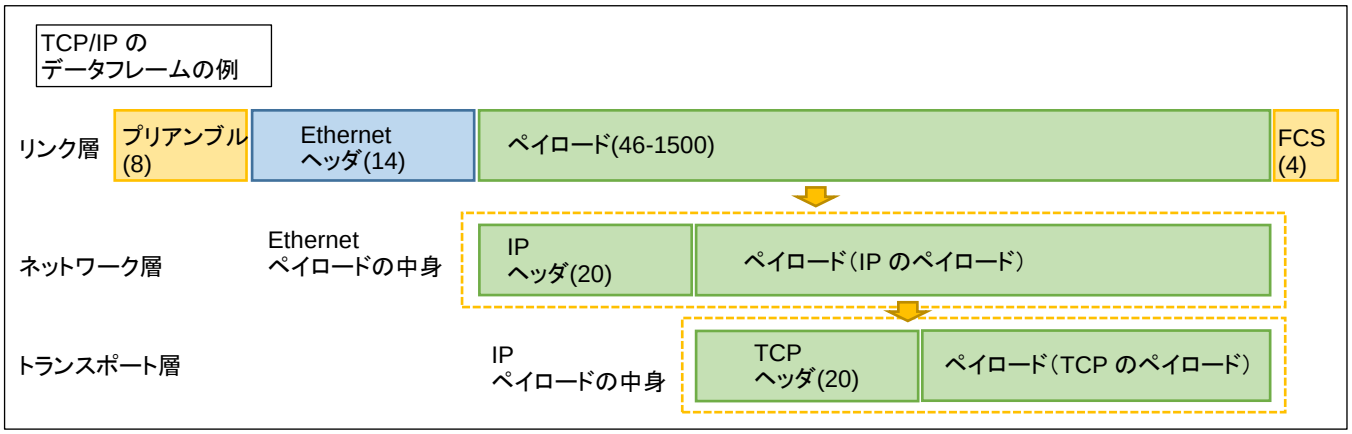
―1 バイトと 1 オクテットに関して―

本書では、8bit を 1 バイトとして表記します。8bit を 1 オクテットと表記するケースもあります。ストレージ容量はバイトが良く使用されますが、通信データはオクテットという使い分けを行っているケースもあります。1 オクテット=必ず 8bit を表す。1 バイト=8bit で使われるケースが圧倒的に多いが、まれに 8bit でない場合もある。よって、厳密に 8bit である事を示す容量として、オクテットが使用されるケースがあります。本書では一般的に広く使われているバイトを使用して、1 バイト=8bit を表す事とします。

Ethernet フレーム中で、ヘッダ＋ペイロードという構成となっています。ヘッダは荷物に貼ってある伝票のようなもので、ペイロードが荷物本体（送りたいもの）です。

Ethernet ヘッダは、14 バイトで「宛先」と「送り主」、「Ethernet タイプ」（どのようなタイプのデータを送るか）が含まれています。

－Ethernet フレームのペイロードに関して－



現在広く使用されている TCP/IP の場合、Ethernet フレームのレベル（階層）で見た場合のペイロードの中身が、「IP ヘッダ＋IP のペイロード」で構成されています。また、IP のペイロードは、「TCP ヘッダ＋TCP のペイロード」で構成されています。

構造としては、マトリョーシカ（もしくは、過剰梱包の荷物？）の様に、箱の中にさらに箱が入っている様な構造です。ネットワーク機器は、Ethernet ヘッダを見て、宛先 MAC アドレス対象の機器まで、Ethernet フレームを届ける。Ethernet フレームを受け取った機器は IP ヘッダを見て自分宛てのデータであれば、さらに TCP ヘッダを見て所定のアプリケーションにデータを届けるといった様な動作となります。ここでは、Ethernet のフレームは多層構造になっているという説明に留めます。各ヘッダの中身に関しては、IP や TCP を扱う際の説明で詳細を示します。

－ジャンボフレームに関して－

Ethernet フレームは（FCS を除くと）最大 1514 バイトとなりますので、ネットワーク経由でサイズの大きなファイルをダウンロードする際は、データとしては小さく分割された状態でデータを受信します。1514 バイトの内、実際のデータは Ethernet, IP, TCP のヘッダで容量を食われるので、1460 バイトとなります。例えば 1.4MB（ここでは、1,460,000 バイトとします）のファイルをダウンロードする際、1460000/1460=1000 個のデータに分割されて、さらに 1 つのデータに付き、54 バイトのヘッダで容量を食われます。最終的に送信されるデータは、1000×(1514+8+4)=1,526,000 バイトとなり、1,460,000/1,526,000=95.7%...4.3%のオーバヘッド（無駄）が生じる形となります(*1)。ここで、フレームサイズを大きくすると無駄が減るのでは？という考え方に基づいて導入されたのが、ジャンボフレームという方式で、一般的にはフレームサイズを 9000 バイト程度に拡張して大きなデータの分割数を減らす方式です。分割されたデータ毎にヘッダが付与される形ですので、分割数を減らす（＝フレームサイズを大きくする）と、データの送信効率（ここでは実行的な通信速度＝ビットレート）が上がります。また、フレーム間の時間もゼロではありませんので、送信フレーム数を減らす事は、通信速度向上に寄与します。

なお、RX65N マイコンでは、ジャンボフレームは取り扱う事ができません。本書では、Ethernet ヘッダ＋ペイロードが最大 1514 バイトのデータで取り扱います。

(*1)TCP の場合は、コントロール用のデータが使用されるので、さらにオーバヘッドは増加します。

3. ネットワーク接続時の注意点

3.1. RX65N マイコンボードをネットワークに接続するにあたり

本キットは RX65N マイコンボードを LAN ネットワークに接続して動作を確認するものですが、ユーザがネットワークのプログラムを行って動作させる過程で意図しない動作やタチの悪い挙動を行う事があります。
ネットワークにおいては、1 台でもルール違反の機器が存在すると、ネットワーク全体がダウンしたり、正常な通信が阻害されるといった事が十分起こり得ます。

キットのプログラムの動作確認やユーザ作成のプログラムのデバッグ中などは、RX65N マイコンボードを安易に社内のハブに接続すると、社内の一部または全部のネットワークに影響を及ぼす可能性があります。そのため、本キットの動作確認においては、切り離されたネットワークでのテスト（独自にハブと PC を用意（もしくは普段使用している PC を社内ネットワークから切り離し）テスト用のネットワークに、RX65N マイコンボードと共に接続））を行う事を強く推奨致します。

また、プログラムが安定して動作する事が確認できた後でも、社内ネットワークに接続する際は、ネットワーク管理者の承認を得た後で、ネットワーク管理者の指示に従い、IP アドレスや MAC アドレスを適切に設定して動作させてください。

3.2. Ethernet 通信のモニタに関して

本マニュアルでは、Ethernet で通信を行う際の動作確認様に、Wireshark というツールを使用する記載があります。Wireshark は、LAN 内の通信をモニタ（通信パケットのダンプ）を行うツールです。プログラムのデバッグや、問題点を洗い出すのに大変役に立ちますが、ネットワークをのぞき見するツールである性質上、社内の規定で使用の禁止や制限が掛っている事が考えられます。（場合によっては、ソフトを起動しただけで、情報システム管理者に通知が行き、PC が使用不可となったり、なんらかのペナルティを課せられる事も考えられます。）
ツールのインストールや使用に関しては、社内の情報システム部署に確認を取る様にしてください。

3.3. PC 上で動作させるサンプルプログラムに関して

本キットでは、RX65N マイコンボードの通信相手として、PC 上で動作するサンプルプログラムが付属しています。このサンプルプログラムに関しても、ネットワークアクセスを伴うソフトウェアとなりますので、社内ネットワークに接続された PC 上で実行する場合は、情報システム部署やネットワーク管理者の確認の上、ご利用ください。

4. Ethernet フレームを送ってみる

4.1. サンプルプロジェクトの開き方

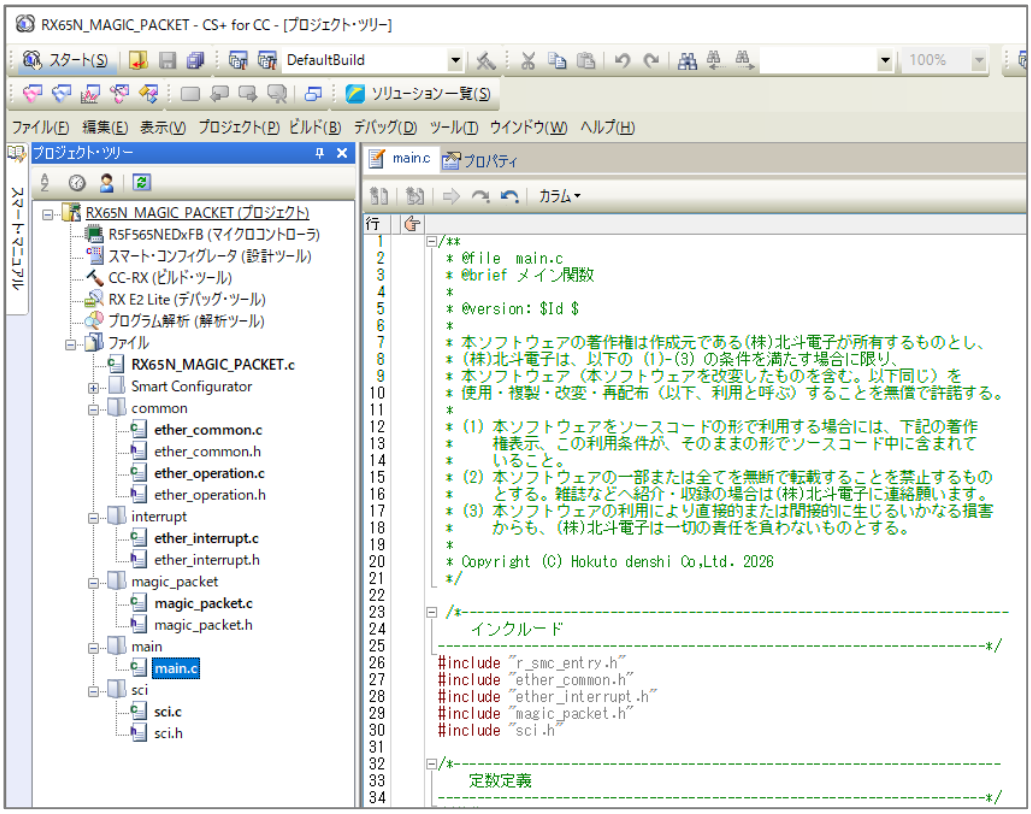
2 章の様な説明ばかりが続くと飽きてくると思いますので、ここで最初のサンプルプログラムを動かしてみたいと思います。

CD 内の、
100pin/144pin のボードをお使いの方は、SOURCE¥144_100¥RX65N_MAGIC_PACKET
176pin のボードをお使いの方は、SOURCE¥176¥RX65N_MAGIC_PACKET
フォルダを、PC のストレージにコピーしてください。

4.1.1. CS+の場合

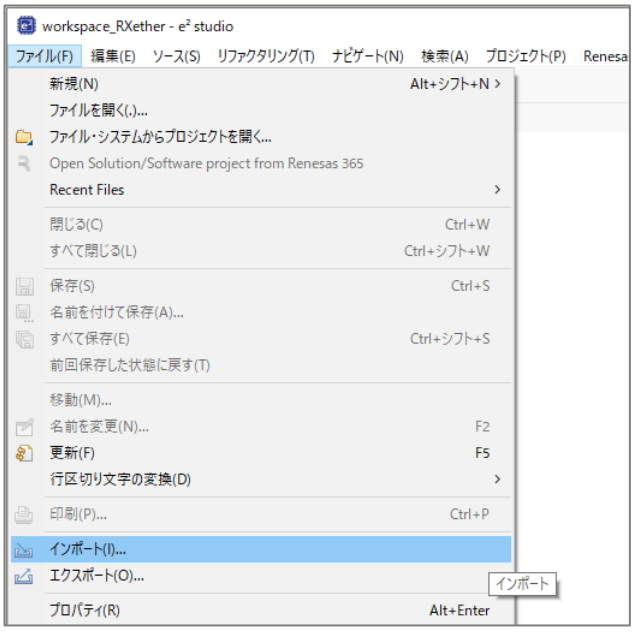
CS+forCC（V8.15 以降）、RX スマート・コンフィグレータ（Version2.28.0 以降）を使いますので予めインストールを済ませておいてください。

PC のストレージにコピーした RX65N_MAGIC_PACKET フォルダ内の、RX65N_MAGIC_PACKET.mtpj をダブルクリックして CS+を起動してください。

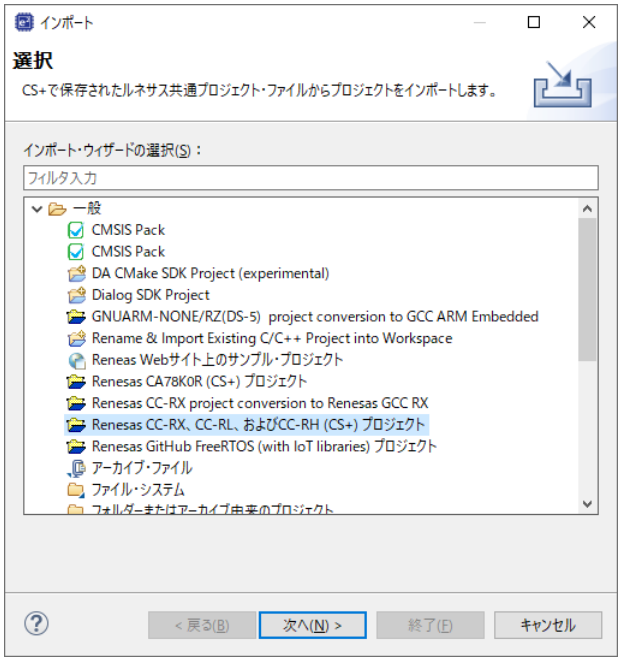


4.1.2. e2studio の場合

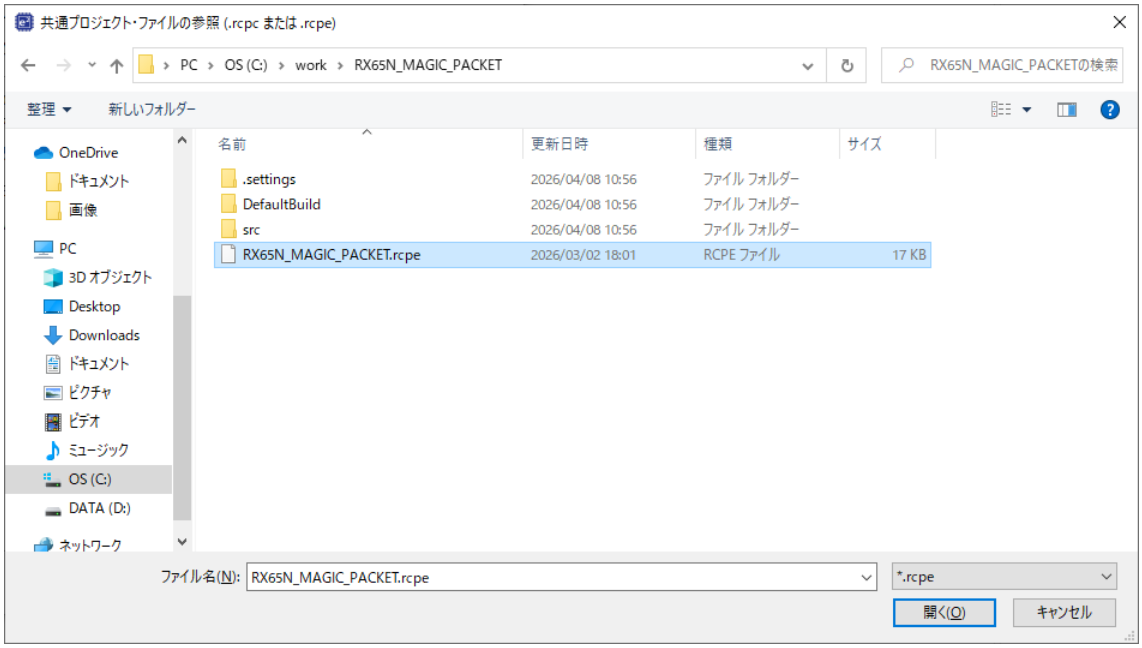
e2studio と CC-RX（V3 系）、RX スマート・コンフィグレータ（Version2.28.0 以降）を使いますので予めインストールを済ませておいてください。



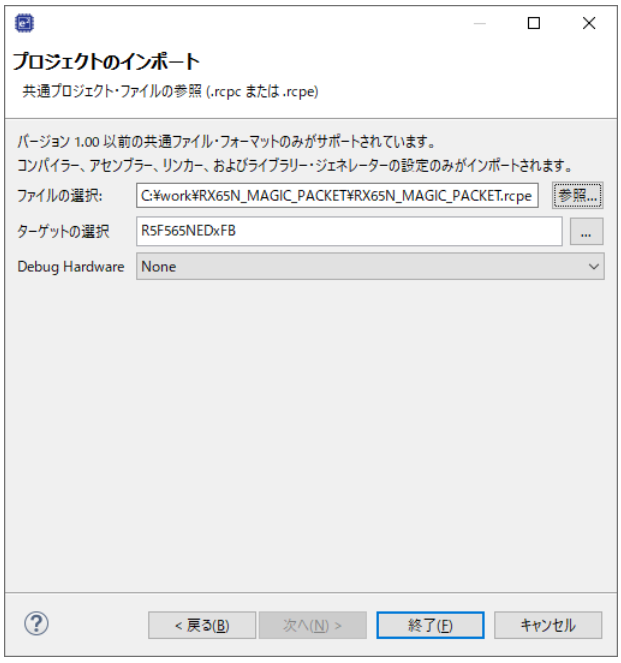
ファイルーインポート



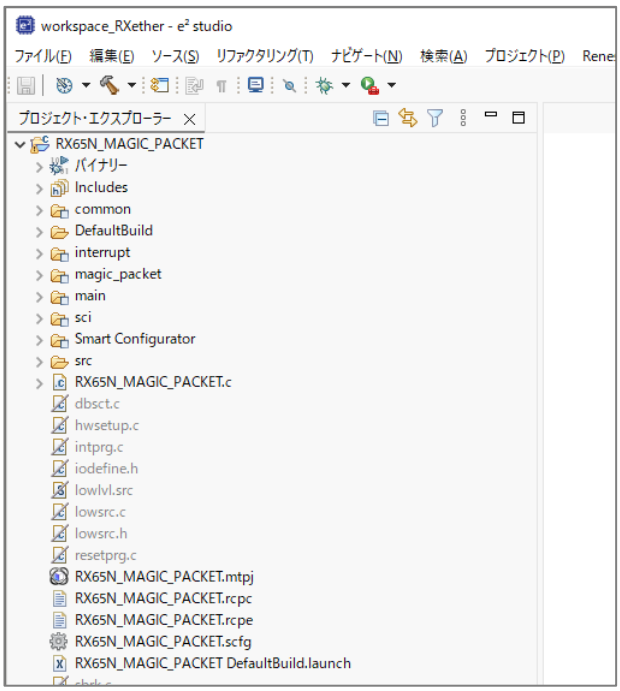
Renesas CC-RX、CC-RL、及び CC-RH(CS+)プロジェクトを選択して「次へ」



PC にコピーしたフォルダの、RX65N_MAGIC_PACKET.rcpe ファイルを選択して「開く」



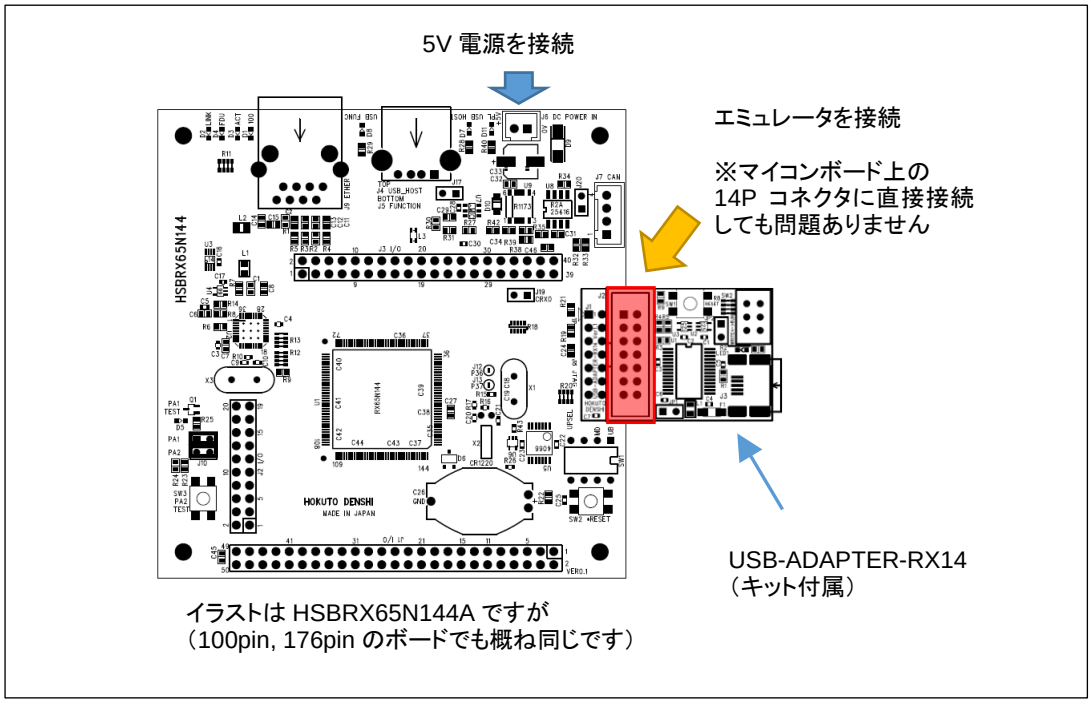
終了



上記の様にワークスペースに、RX65N_MAGIC_PACKET プロジェクトが見えていれば OK です。

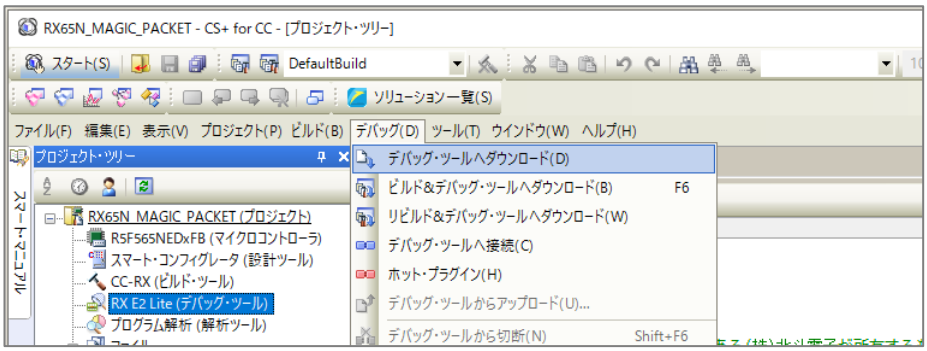
4.2. マイコンボードにプログラムを書き込む方法

4.2.1. ルネサス製エミュレータを使用する場合(CS+)

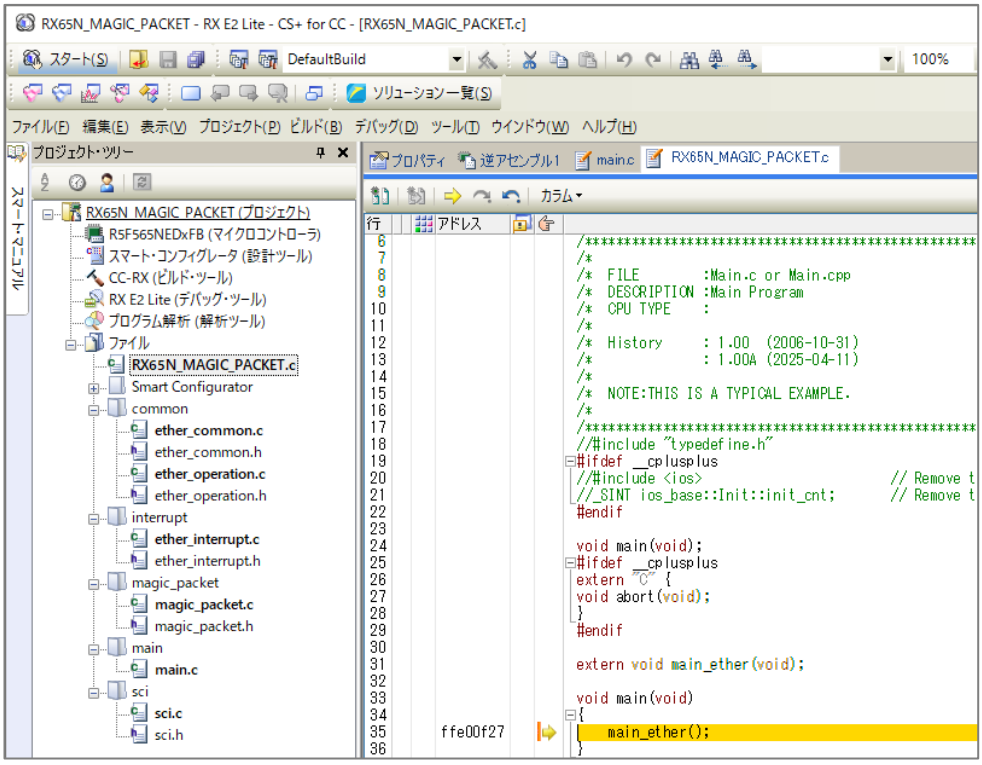


ボードには、電源を接続してください。エミュレータからの給電には対応していません。

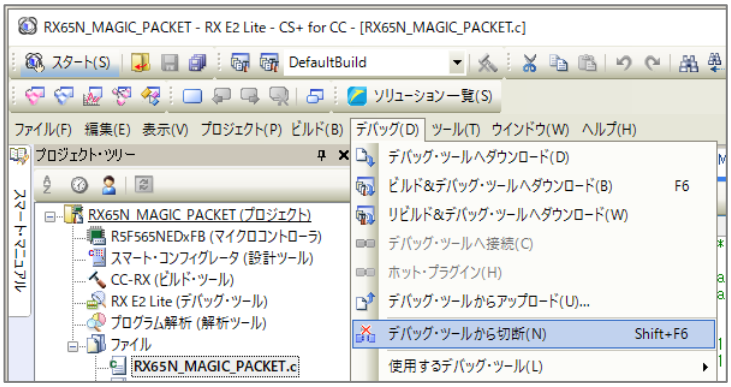
エミュレータ（E2Lite など）を、マイコンボード上（または、USB-ADAPTER-RX14 上）の 14P コネクタに接続してください。



デバッグ　ー　デバッグ・ツールへダウンロード　を実行



main_ether()のところに黄色の矢印が出れば OK です。

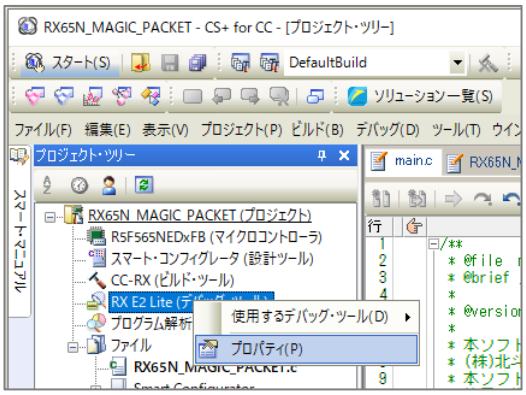


デバッグ　－　デバッグ・ツールから切断でエミュレータ動作を終了させてください。

エミュレータを USB-ADAPTER-RX14 上の 14P コネクタから取り外してください。

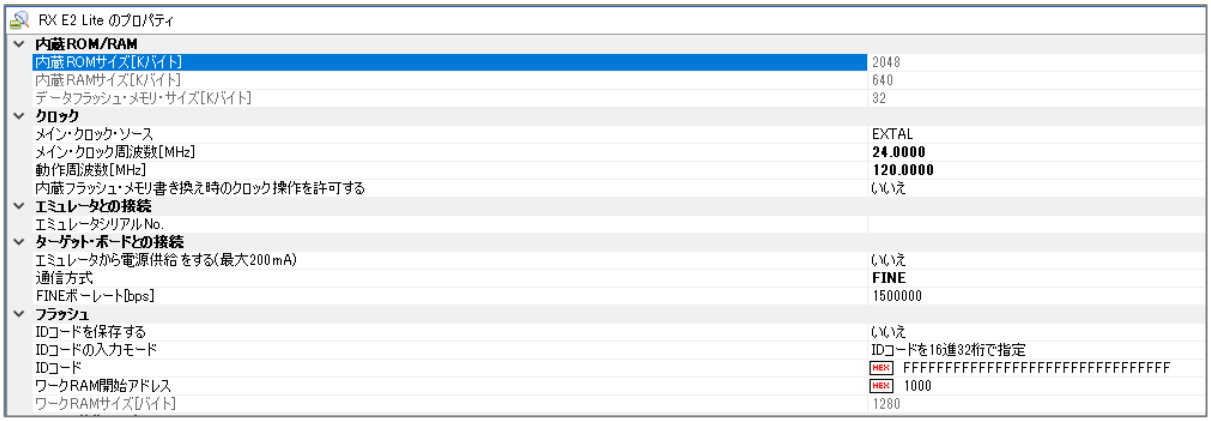
（上記操作で、プロジェクト内のプログラムがマイコンボードに書き込まれた状態となります。）

なお、E2Lite 以外のエミュレータをお使いの場合は、RX E2Lite（デバッグ・ツール）をお使いのエミュレータに変更してください。（下記画面の「使用する・デバッグツール」で使用しているエミュレータを選択）



E1, E20 をお使いの場合は、RX E1(Serial), RX E20(Serial)の方を選択してください。

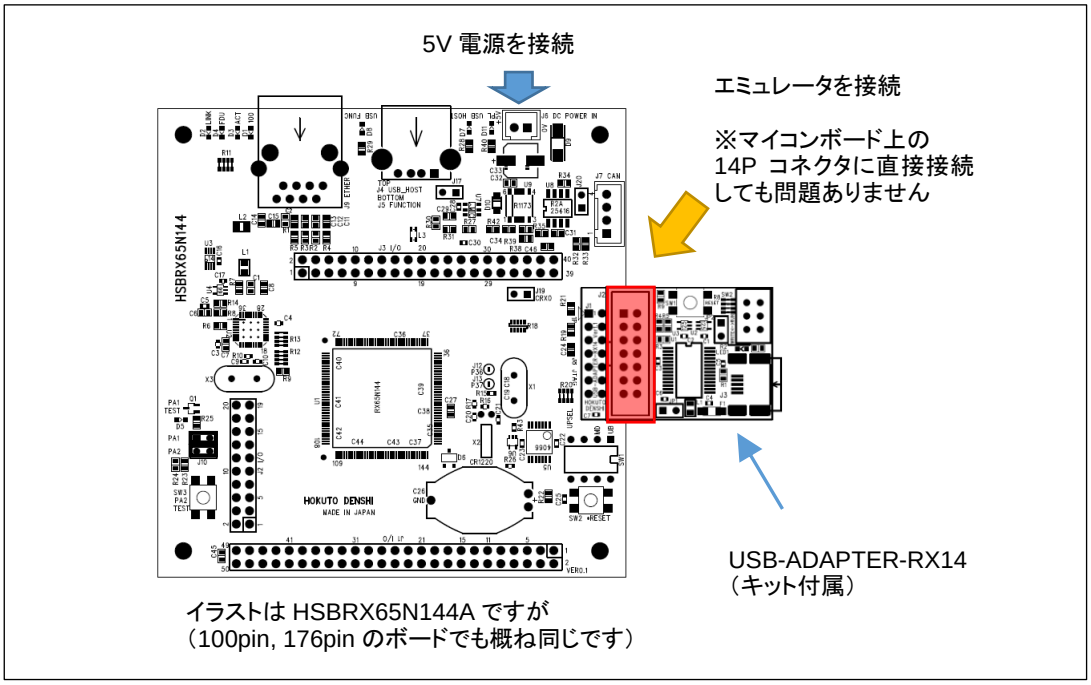
上記のプロパティを開く。



エミュレータの設定は、
メインクロック周波数[MHz]　24
動作周波数[MHz]　120
エミュレータから電源供給をする　いいえ
通信方式　FINE　（※E2, E2Lite の場合）
を選択してください。

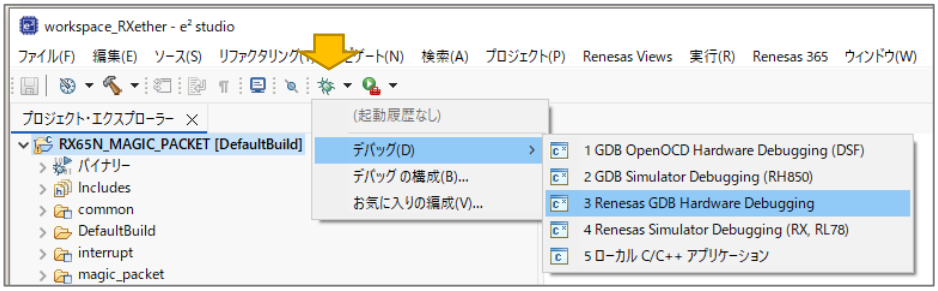
（E2Lite をお使いの方は、最初から E2Lite が選択されており設定済みなので変更は不要です。）

4.2.1. ルネサス製エミュレータを使用する場合(e2studio)

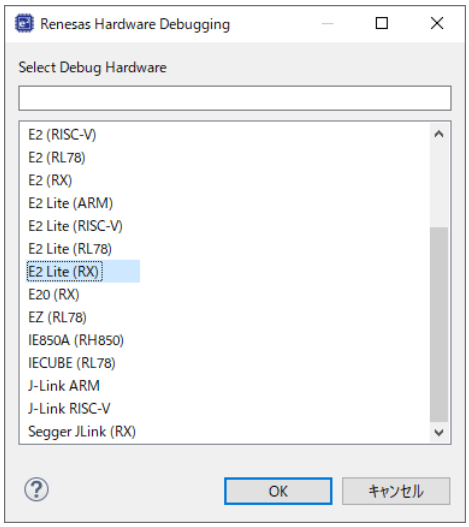


ボードには、電源を接続してください。エミュレータからの給電には対応していません。

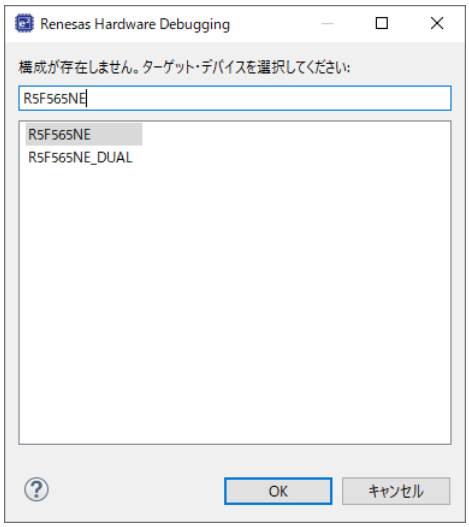
e2studio の場合は、



デバッグアイコンを押して、デバッグ－Renesas GDB Hardware Debugging を選択

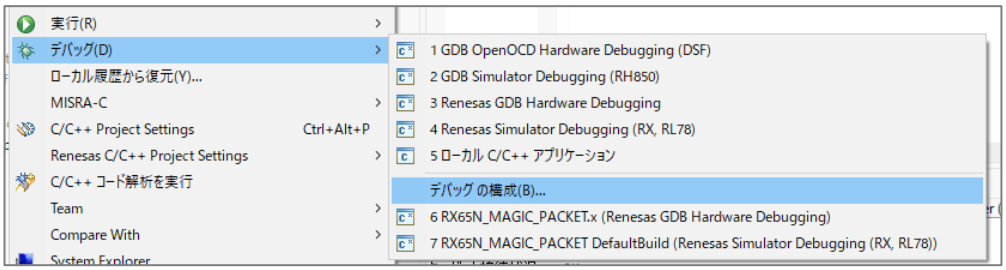


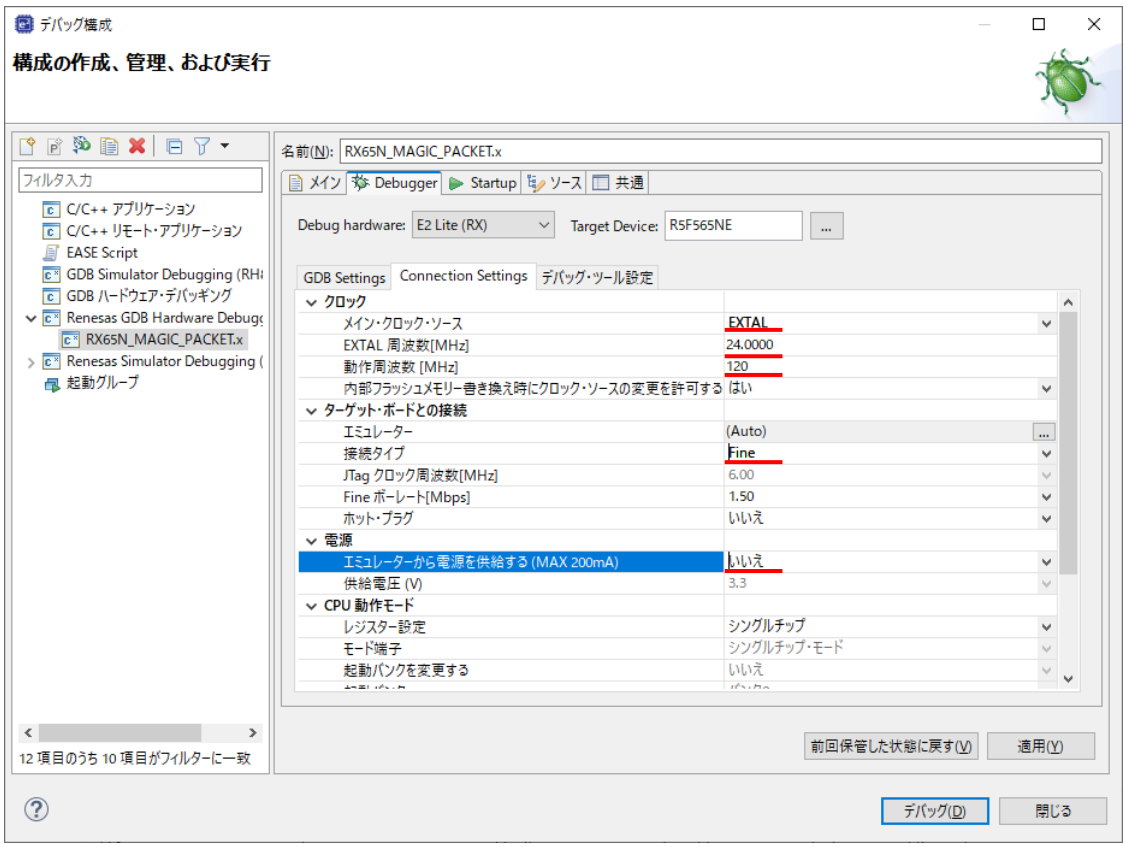
お使いのエミュレータを選択



R5F565NE を選択 「OK を押す」

プロジェクト名のところを右クリックして　デバッグ－でバックの構成





メイン・クロック・ソース　EXTAL

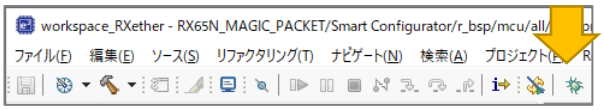
EXTAL 周波数[MHz]　24

動作周波数[MHz]　120

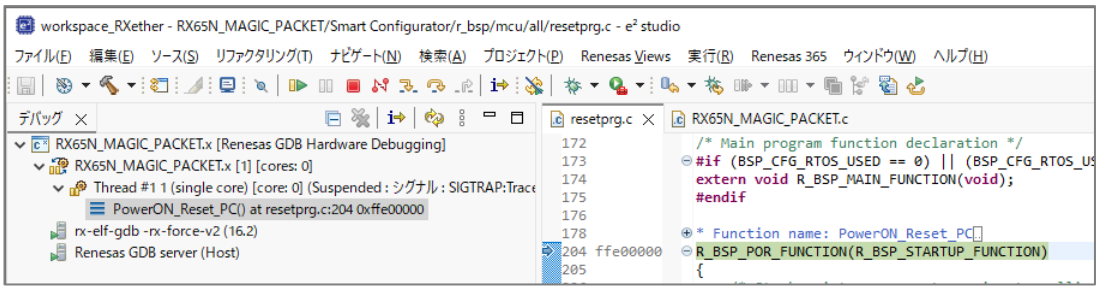
接続タイプ　Fine

エミュレータから電源を接続する　いいえ　（※重要）

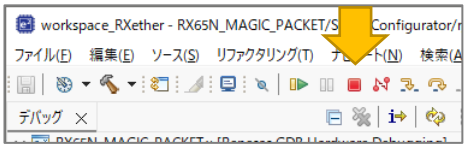
を設定してください。



デバッグアイコンを押す



コンソールにはダウンロード終了のメッセージ。fffe0000 番地でブレークした状態となれば Ok です。

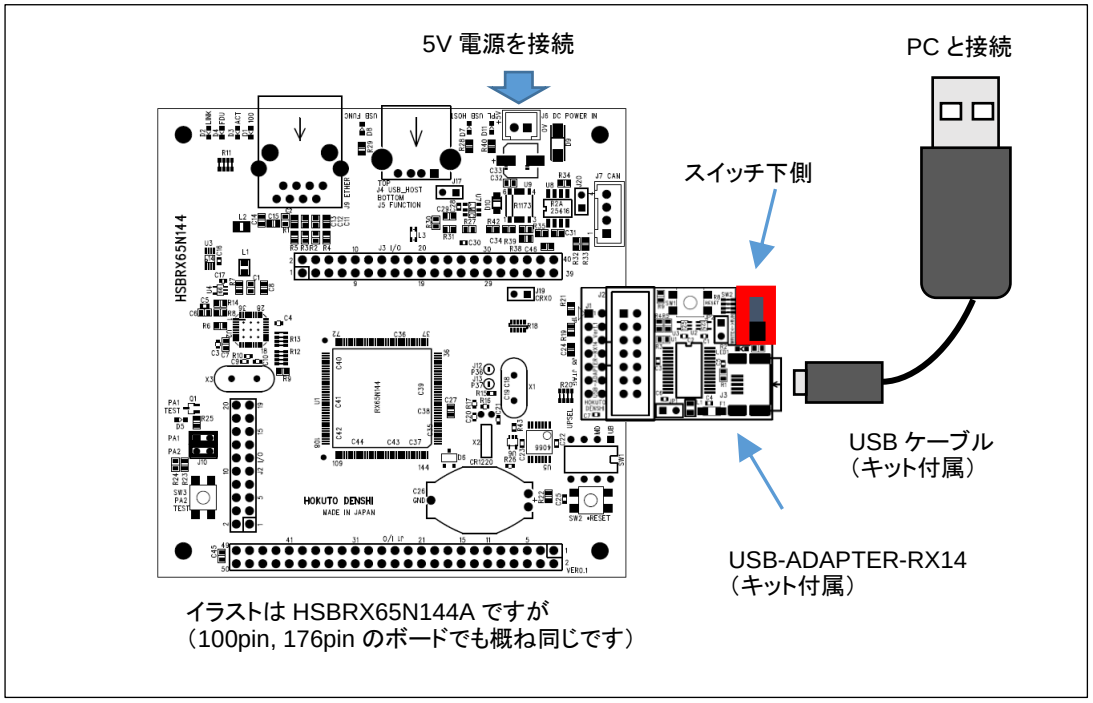


赤の四角のアイコンを押す。

エミュレータを USB-ADAPTER-RX14 上の 14P コネクタから取り外してください。

（上記操作で、プロジェクト内のプログラムがマイコンボードに書き込まれた状態となります。）

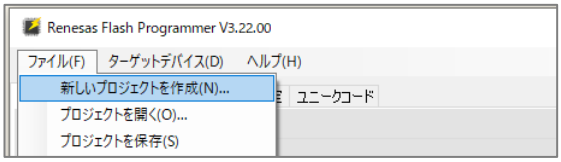
4.2.2. USB-ADAPTER-RX14 を使用する場合



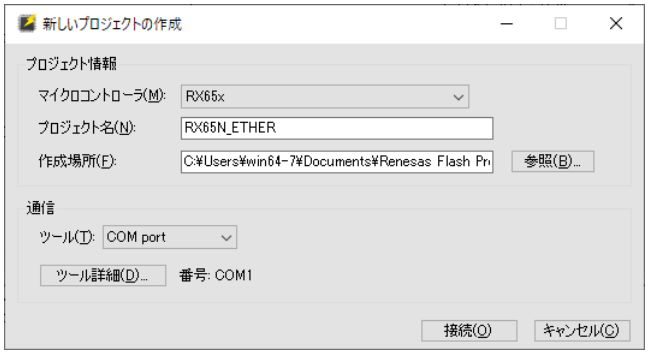
マイコンボードに、USB-ADAPTER-RX14 を接続し、USB ケーブルで PC と接続してください。USB-ADAPTER-RX14 のスイッチは上図での下側（USB コネクタ側）に切り替えてください。

RenesasFlashProgrammer（以下 RFP）を、ルネサスエレクトロニクス社の Web からダウンロードして、インストールしてください。

RFP を起動して、



ファイル　ー　新しいプロジェクトを作成

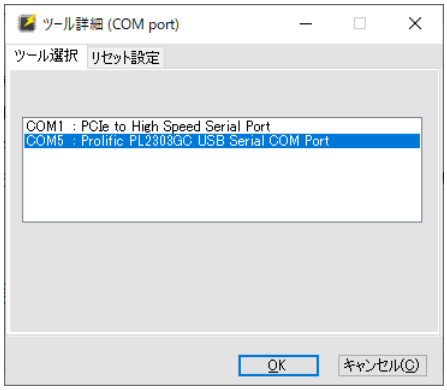


マイクロコントローラ　RX65x　を選択

プロジェクト名　任意の名称を入力

ツール　COM port　を選択

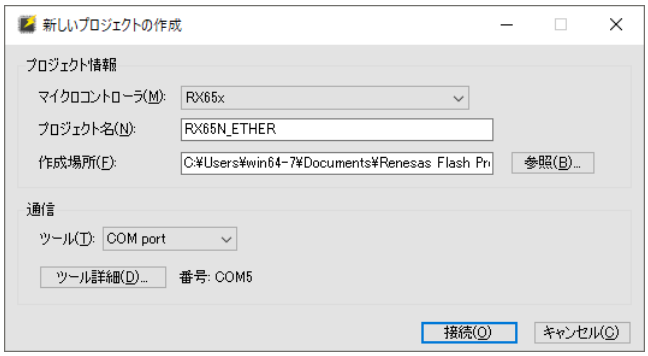
ツール詳細ボタンを押す



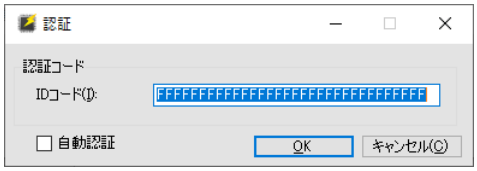
（複数の COM ポートが見えている場合は）

Prolific PL2303GC USB Serial COM Port を選択 「OK」

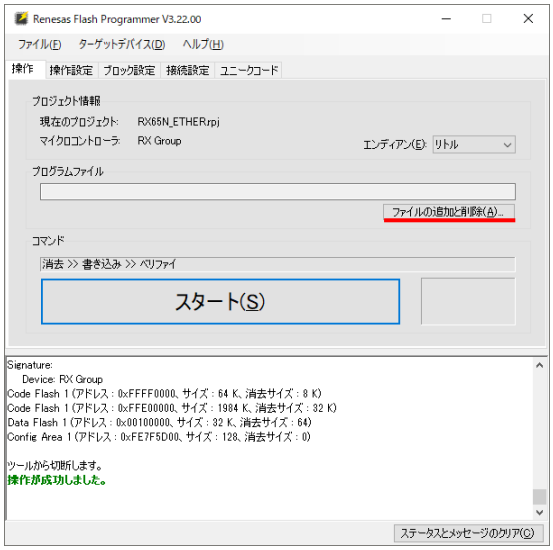
（※このマニュアルでは COM5 ですが、COM ポート番号は環境により異なります）



接続を押す

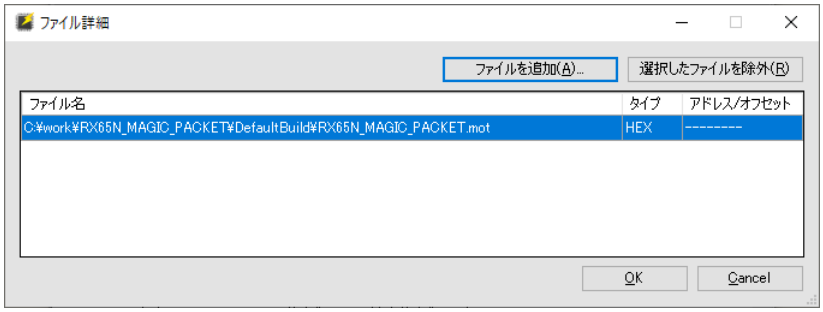


OK を押す



操作が成功しました　と表示されれば OK です

「ファイルの追加と削除」を押す



「ファイルを追加」を押して、プロジェクトフォルダ内の

RX65N_MAGIC_PACKET¥DefaultBuild¥RX65N_MAGIC_PACKET.mot

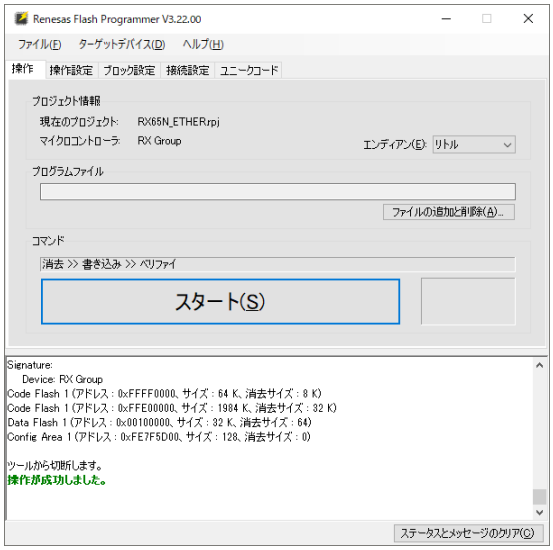
を選択

（DefaultBuild 内の拡張子.mot ファイルを選択してください）

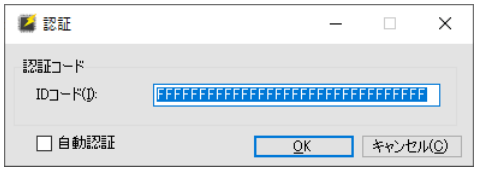
「OK」を押す

マイコンボード上の SW1（リセットスイッチ）を押す　※重要

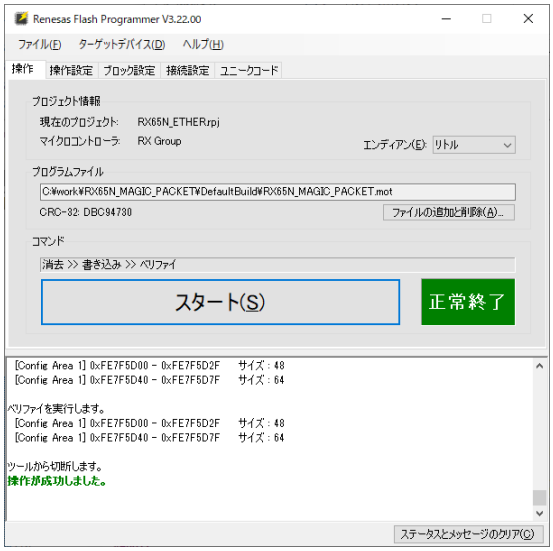
（または、USB-ADAPTER-RX14 上のプッシュスイッチを押す）



「スタート」を押す。

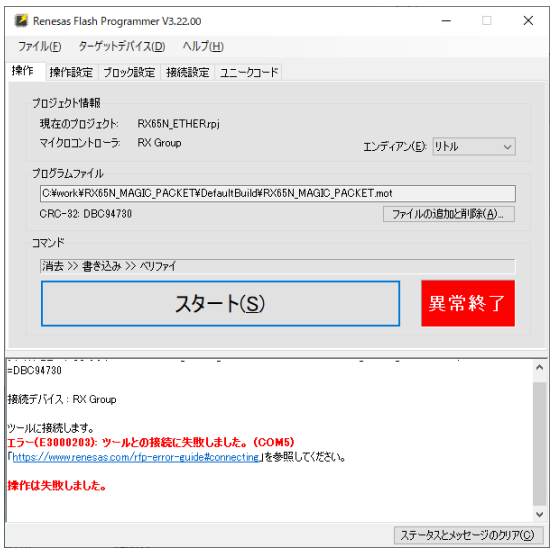


「OK」を押す



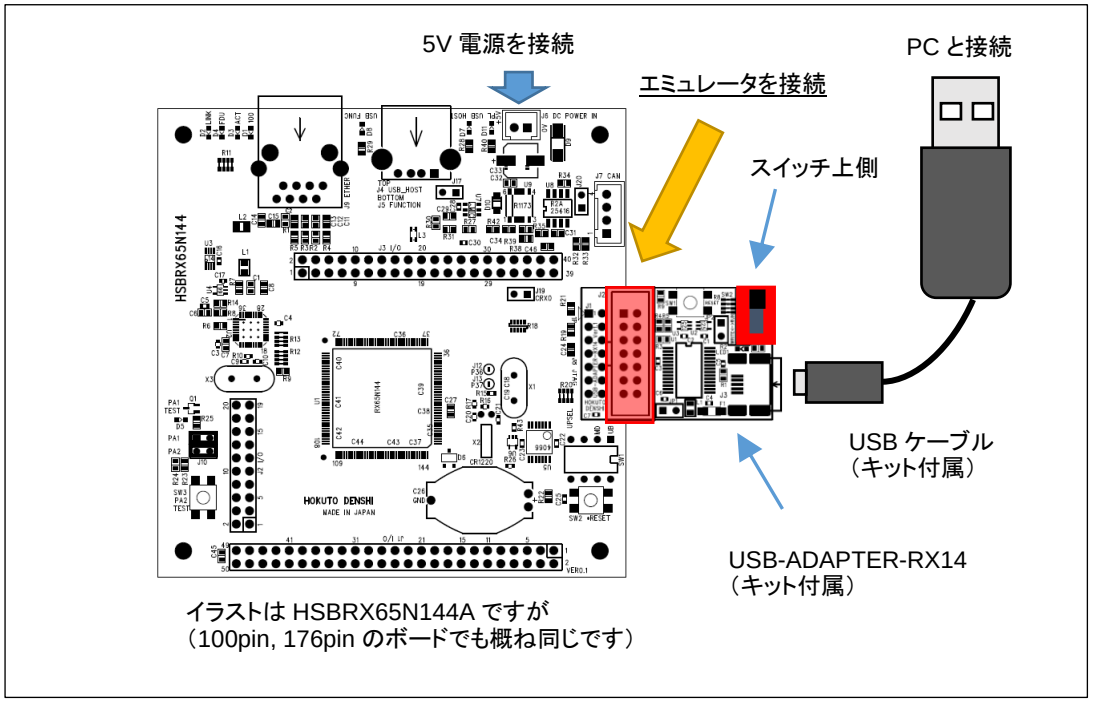
操作が成功しましたのメッセージが表示されれば OK です。

（マイコンボードにプログラムの書き込みが完了しています。）



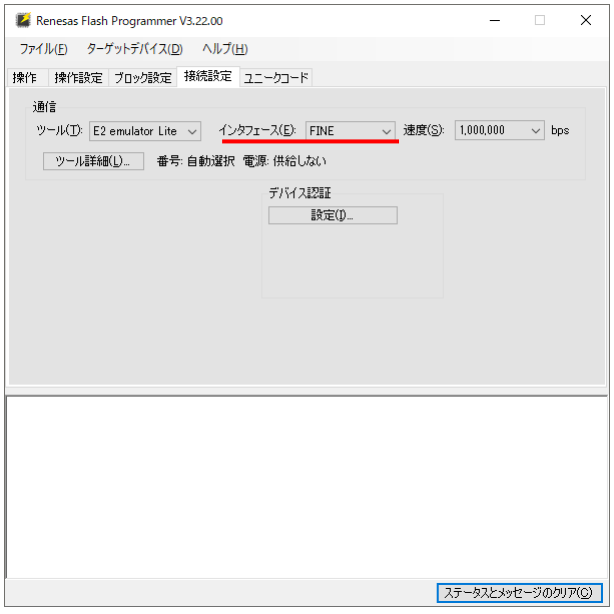
ツールとの接続に失敗しましたというエラーとなった場合は、(この場合は)COM5 で端末が開いていないかを確認してください。

4.2.3. ルネサス製エミュレータを使用する場合(RenesasFlashProgrammer)



マイコンボードに、USB-ADAPTER-RX14 を接続し、USB-ADAPTER-RX14 上の 14P コネクタにエミュレータを接続してください。書き込み手順は、4.2.2 と同じです。

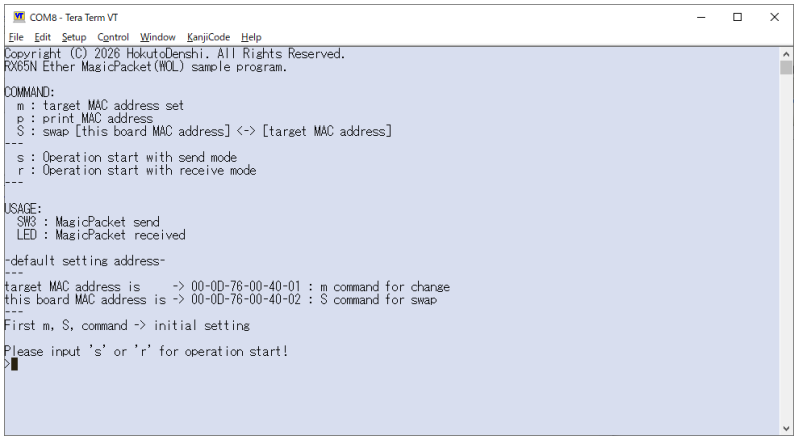
RenesasFlashProgrammer の設定は、



インタフェース　FINE を選択してください。

※インタフェース　2-wire UART を選択した場合

・端末を開いている状態



この状態で書き込みを行うと

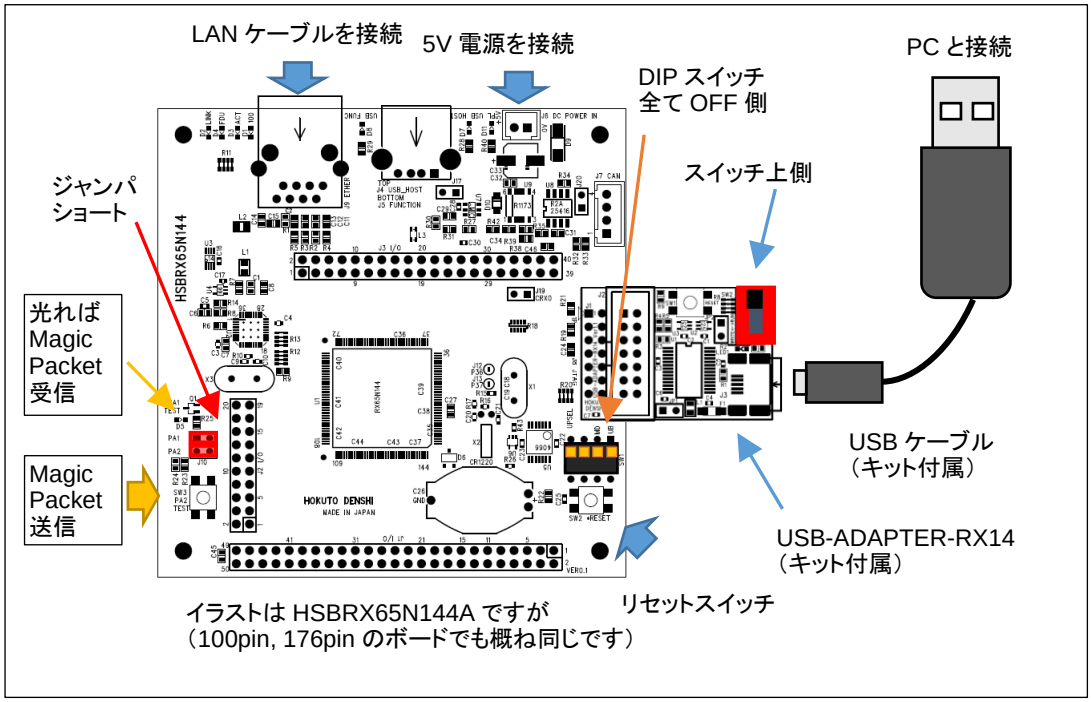
↓

端末が文字化けしてしまいます。(teraterm の場合は、Control-Reset Terminal, Edit-Clear Buffer で文字化けは解消します。)

インタフェースで FINE を選んだ場合、端末を開いた状態で書き込みを行っても、端末表示には影響しません。

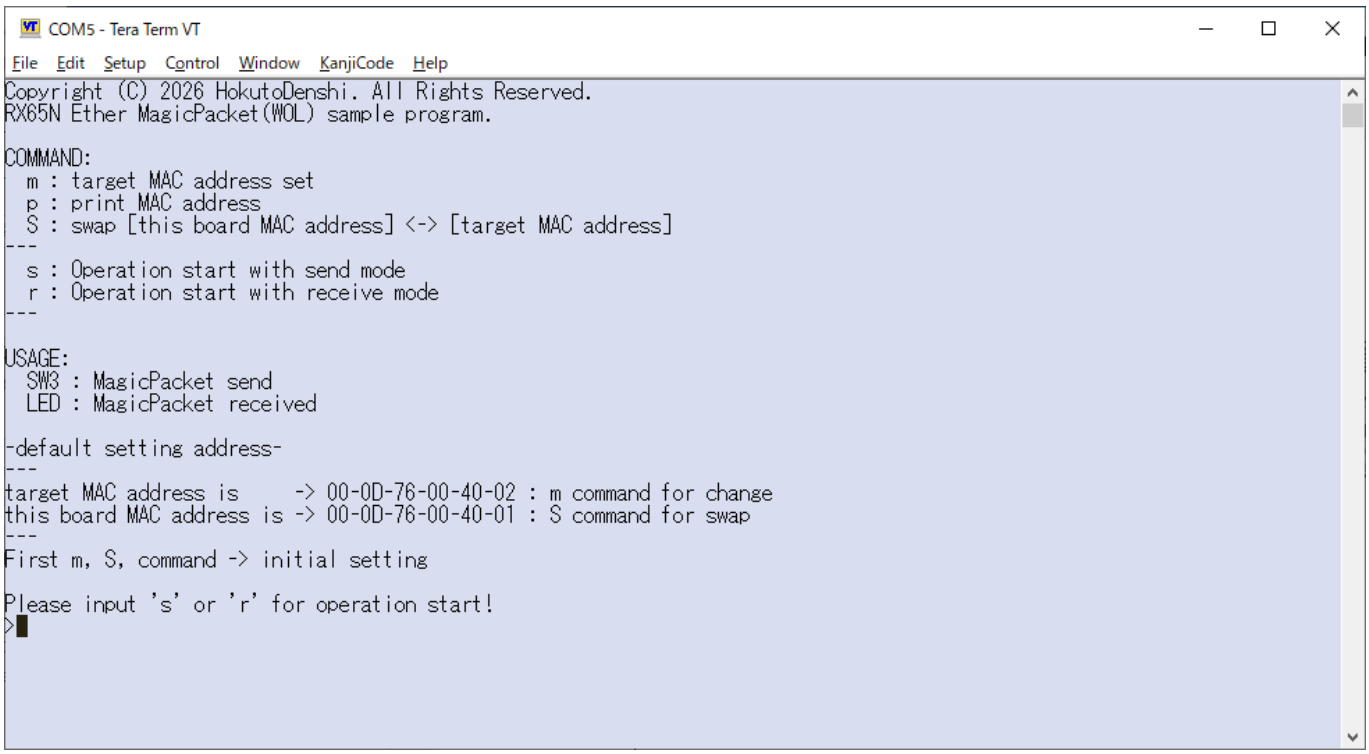
4.3. MagicPacket(WOL)の送受信

ここから、マイコンボードを使用して Ethernet フレームの送受信を伴う実際の動作を行わせてみます。



USB-ADAPTER-RX14 上のスイッチを上側（基板エッジ側）に設定してください。PC 上で、端末ソフト（teraterm 等）を使い、端末を開いてください。

端末ソフトは、115200bps, 8bit, ストップビット 1、パリティなしの設定としてください（速度以外は通常デフォルト）。



電源投入やマイコンボードのリセットで、端末には上記の表示が出るはずです。

－MagicPacket とは－

MagicPacket は、WOL(WakeOnLan)とも呼ばれるデータで、PC の電源投入に用いられます。WOL に対応した PC で、BIOS 上で WOL 機能を有効化しておくと、PC の電源がオフの状態で、ネットワーク経由で MagicPacket（WOL パケット）を送信すると、遠隔で PC の電源を入れる事ができるというものです。

MagicPaket は以下の様な、単純な Ethernet フレームとなっています。

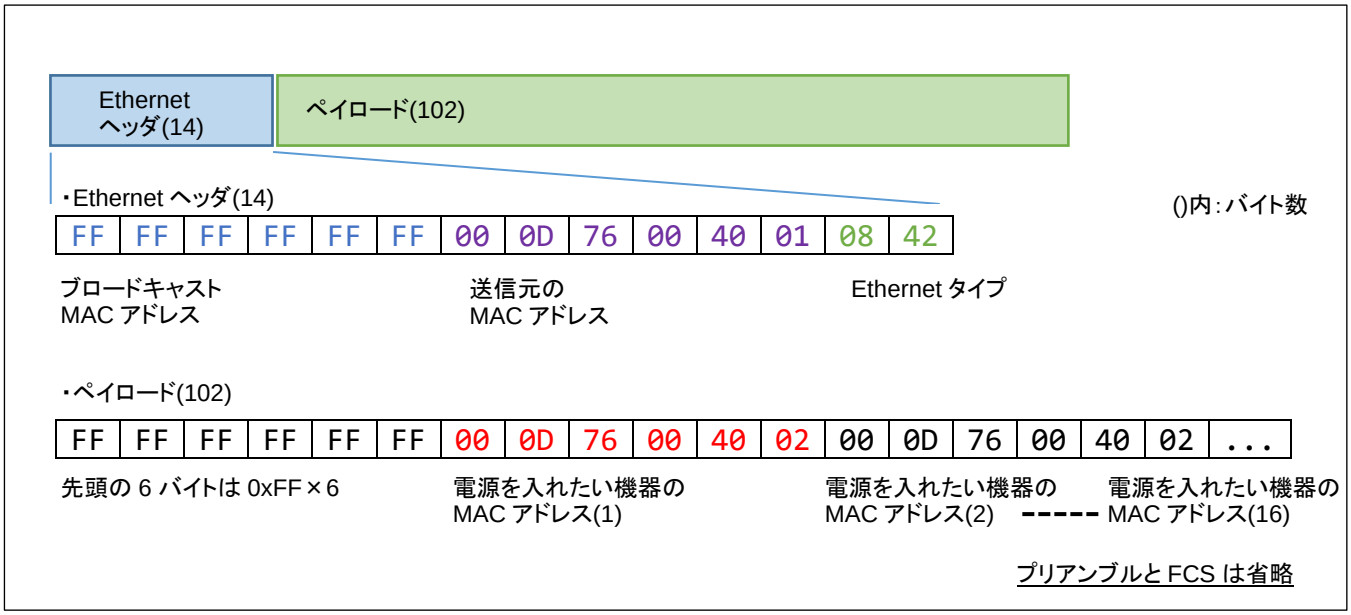


図 3-1 MagicPacket の構造

Ethernet ヘッダの宛先 MAC アドレスは、FF-FF-FF-FF-FF-FF となります。これは、ブロードキャストアドレスと呼ばれ、特定のターゲットに送る訳ではなく、全ての機器に送るためのアドレスとなります。

送信元の MAC アドレスは、RX65N のマイコンボード自身のアドレス（ここでは、00-0D-76-00-40-01 に設定しています）です。

その後の、Ethernet タイプは、どのようなデータを送るかを示した番号となり、MagicPacket の場合は、0x0842 となります。

ペイロードに格納されるデータは、合計 102 バイトで先頭の 6 バイトは 0xFF。7~12 バイト目は、電源を入れたい機器の MAC アドレスが入ります。13 バイト目以降は、電源を入れたい機器の MAC アドレスの繰り返しとなります。合計、16 回 MAC アドレスが繰り返される形となります。

4.3.1. MagicPacket の受信

まずは、MagicPacket の受信を試してみます。

RX65N マイコンボードで、MagicPacket を受信する場合、端末からキーボードで r を入力してください。

```
>command=r

Operation start with MagicPacket receive mode.
this board MAC address = 00-0D-76-00-40-01
Ether0: LINK-UP
-> MagicPacket waiting ...
```

LAN ケーブルが接続されており物理的に LAN に接続した場合、上記の様な表示となります。
（Ether0: LINK-UP の表示が出ない場合は、LAN ケーブルの接続を確認してください。）

この状態で MagicPacket の受信待ちとなっていますので、外部から MagicPacket を送信してみます。
MagicPakcet を送信可能なフリーソフトもありますが、CD 内の
PC_APPLI¥MagicPacketSend.exe
を起動してください。

ーPC ソフトの実行に必要なランタイムに関してー
CD に格納されている、PC_APPLI 以下のソフトは、C#, .NET10 で作成されており、実行するためには、.NET10 のランタイムライブラリが PC にインストールされている必要があります。（OS は、Windows11 か Windows10 に対応）

プログラムの起動時にランタイムエラーが出た場合は、
「.NET10 ランタイム」で検索し、Microsoft の Web ページより「.NET デスクトップランタイム」(x64)をダウンロードしてインストールしてください。

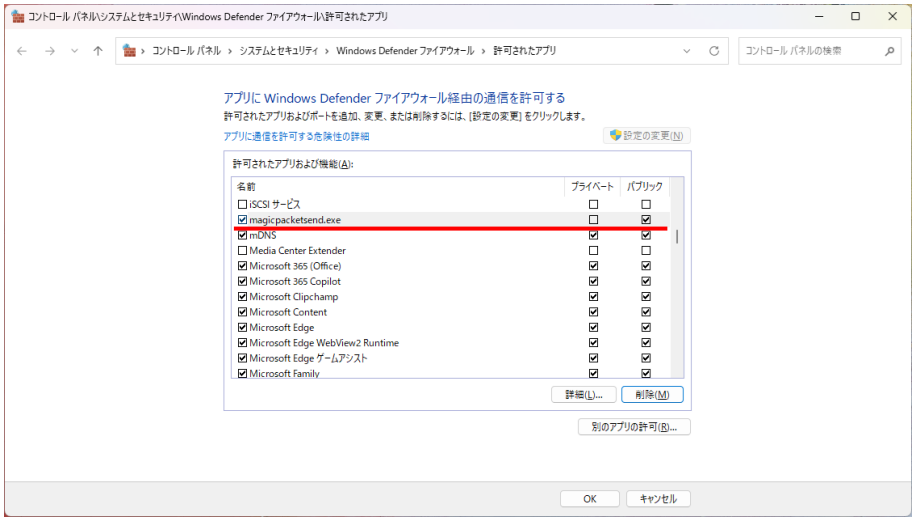
－Windows ファイヤウォールの設定－

MagicPacket 送信アプリケーションを実行すると、

Windows セキュリティの設定画面が出ます。ここでは、許可を押してください。

なお、ここで設定した項目は、

Windows セキュリティ設定の、「ファイヤウォールによるアプリケーションの許可」



上記のリストに追加されますので、許可前の状態に戻したい場合は、「設定の変更」ボタンを押して、magicpacketsend.exe を選択して「削除」してください。または、この画面で許可設定の変更が可能です。

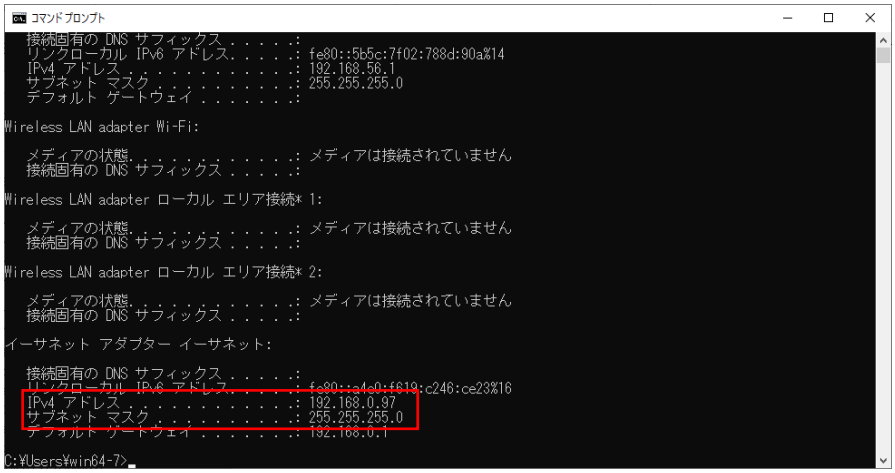
MagicPacketSend アプリケーションを起動すると、以下の様な画面が出ます。

MagicPacket 送信先はデフォルトで、RX65N のマイコンボードの MAC アドレスに設定されていますが、変更は可能です。

送信先 IP アドレス範囲は、PC の IP アドレスに応じて変更してください。

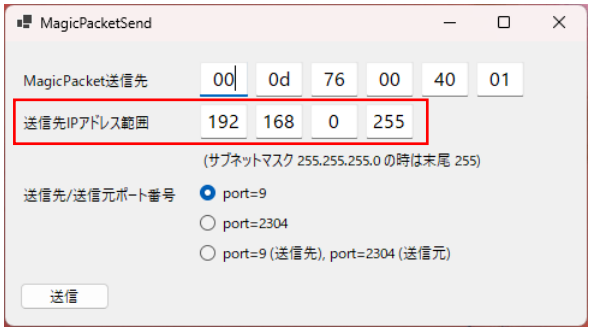
PC のコマンドプロンプト（検索ボックスで、cmd と入力してリターン）を開きます。

コマンドプロンプトから、ipconfig と入力してリターン。



この PC は、LAN ケーブルでネットワークに接続しています。
WiFi でネットワークに接続している場合、Wireless LAN adapter Wi-Fi:のところの IPv4 アドレスとサブネットマスクを参照してください。

イーサネットアダプタの、IPv4 アドレスが、PC に設定されている IP アドレスとなります。
この場合は、192.168.0.97 になります。また、ネットマスクは、255.255.255.0 です。



IP アドレスに関しては、5 章で詳細に説明しています。

IP アドレスに関して？？？となった場合は、4 章の後半は飛ばして、先に 5 章を読んでください。

IPv4 アドレス	192	168	0	97
サブネットマスク	255	255	255	0
送信先 IP アドレス範囲	192	168	0	255

送信先 IP アドレス範囲は、サブネットマスクが 255 の時は、IPv4 アドレスを書く。サブネットマスクが 0 の時は、255 を書くというルールです。IP アドレスに関しては、後で詳しく説明します。

送信先/送信元ポート番号は変更は、9 のままで問題ありませんが、2304 番も選べるようになっています。ポート番号に関しても、後で詳しく説明します。

「送信」ボタンを押すと、
RX65N マイコンボードの LED(D5)が点灯し、端末に

```
-> MagicPacket waiting ...
Ether0: receive abort detected
Ether0: MagicPacket received.
Ether0: LINK-UP
-> MagicPacket waiting ...
```

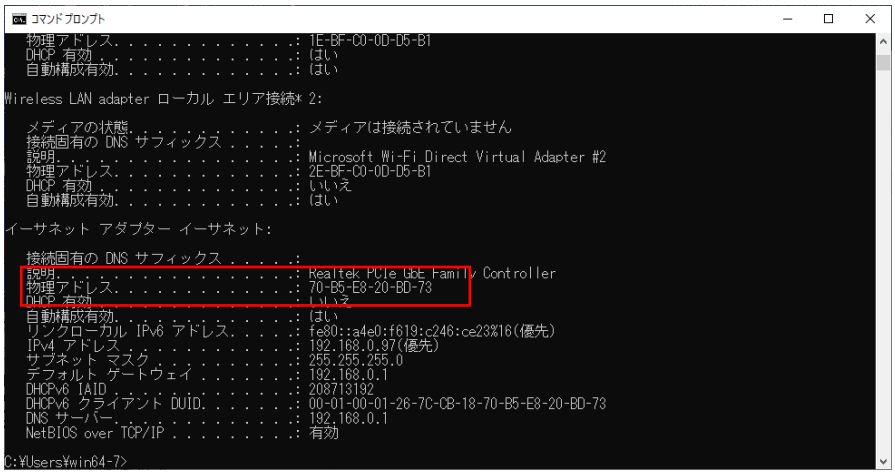
MagicPacket received
の表示が出ます。(PC のアプリケーションで送った Ethernet フレームがマイコンボードに届いた場合)
(その後、MagicPacket waiting... 表示で、再度 MagicPacket 受信待ちとなります)

上記の様な動作となった場合は、PC のアプリケーションから送った Ethernet フレーム（データ）が、マイコンボード側で受信できたという事です。

4.3.2. MagicPacket の送信

次に、MagicPacket の送信を試してみます。

PC 側では、ネットワークインタフェースの MAC アドレスを調べます。コマンドプロンプトを開き、
ipconfig /all
と入力します。



ここで、物理アドレスとなっているのが、PC の MAC アドレスです。（ネットワークインタフェースが複数表示される場合は、RX65N マイコンボードをつないでいるネットワークのインタフェースの物理アドレスを見る必要があります。Wifi 接続なのか、LAN ケーブル接続なのかを確認してください。）このケースでは PC 本体の LAN コネクタを使用していますので、「イーサネットアダプタ」の物理アドレスの表示を見ます。

マイコンボードのリセットボタンを押すか、電源を再投入してください。

```
-default setting address-
---
target MAC address is        -> 00-0D-76-00-40-02 : m command for change
this board MAC address is -> 00-0D-76-00-40-01 : S command for swap
---
First m, S, command -> initial setting
```

初期値は、00-0D-76-00-40-02 に対して MagicPacket を送信する様になっています。ここでは、70-B5-E8-20-BD-73 の PC に対して MagicPacket を送信したいので、キーボードから m を入力して MAC アドレスの設定を行います。

```
>command=m
target MAC address set(please input 2 letters(0-9,a-f,A-F) hex x 6)

MAC[0] >
Please input 's' or 'r' for operation start!
>
```

上記の様な MAC アドレス入力画面となりますので、2 桁ずつ PC の MAC アドレスを入力してください。

```
>command=m
target MAC address set(please input 2 letters(0-9,a-f,A-F) hex x 6)

MAC[0] >70
MAC[1] >b5
MAC[2] >e8
MAC[3] >20
MAC[4] >bd
MAC[5] >73
MAC address set to -> 70-B5-E8-20-BD-73
```

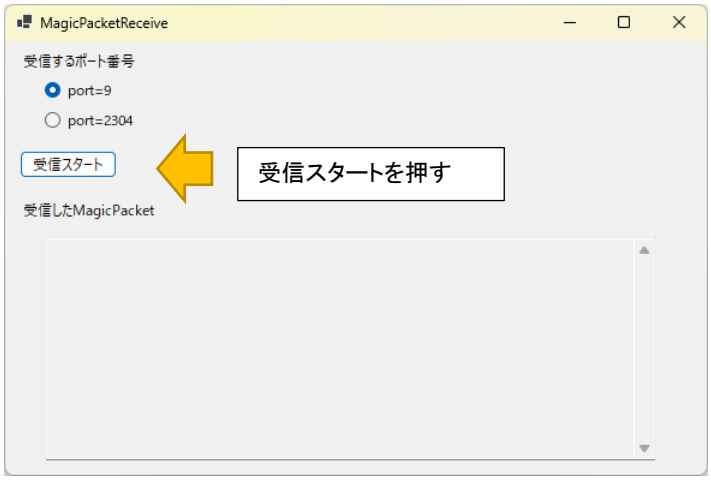
a~f の入力は大文字でも小文字でも
どちらでも構いません

RX65N マイコンボードで、MagicPacket を送信する場合、ソフトの起動後、送信先 MAC アドレスの設定後に端末からキーボードで s を入力してください。

```
>command=s

Operation start with MagicPacket send mode.
PUSH SW3 -> send MagicPacket
Ether0: LINK-UP
```

PC の方では、
PC_APPLI¥MagicPacketReceive.exe
を実行してください。

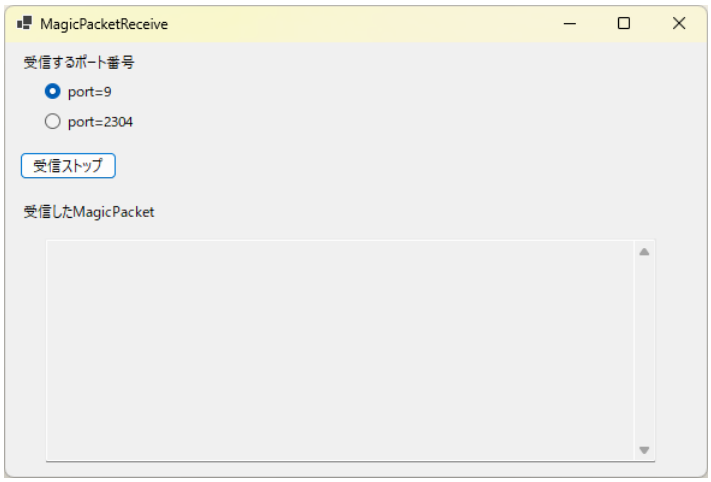


初回起動時、ファイヤウォールの許可の画面が出ますので、「許可」設定としてください。
受信スタートのボタンを押すと、MagicPacket の受信を開始します。

次に、マイコンボードの SW3 を押してください。（SW3 を押したタイミングで MagicPacket 送信）

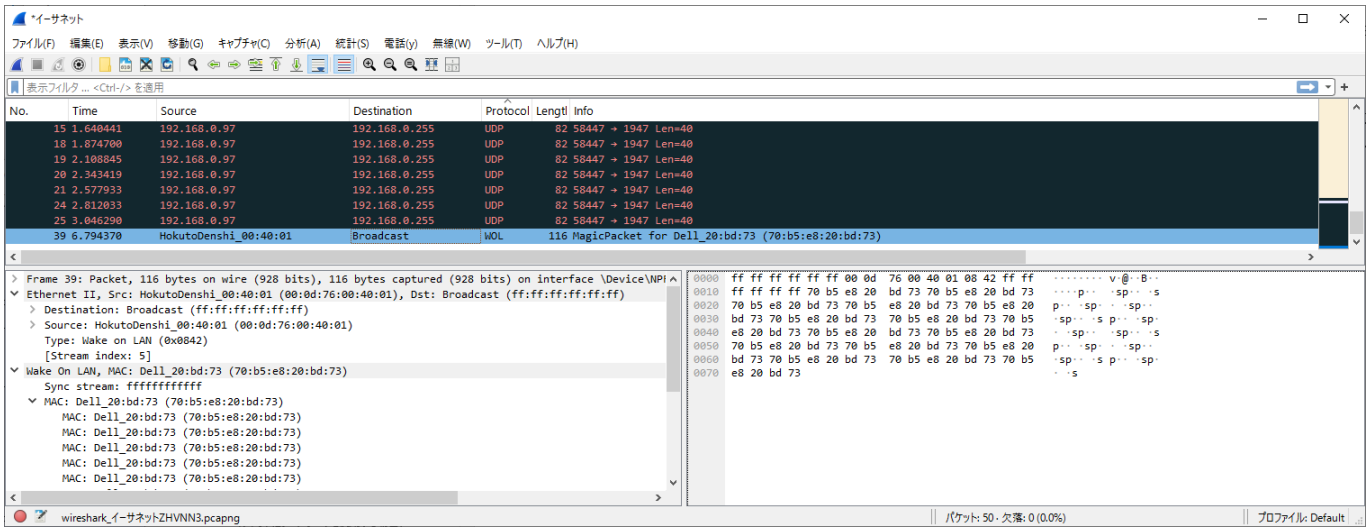
```
MagicPacket data send (70-B5-E8-20-BD-73)
```

上記表示となれば、MagicPacket は送信されました。

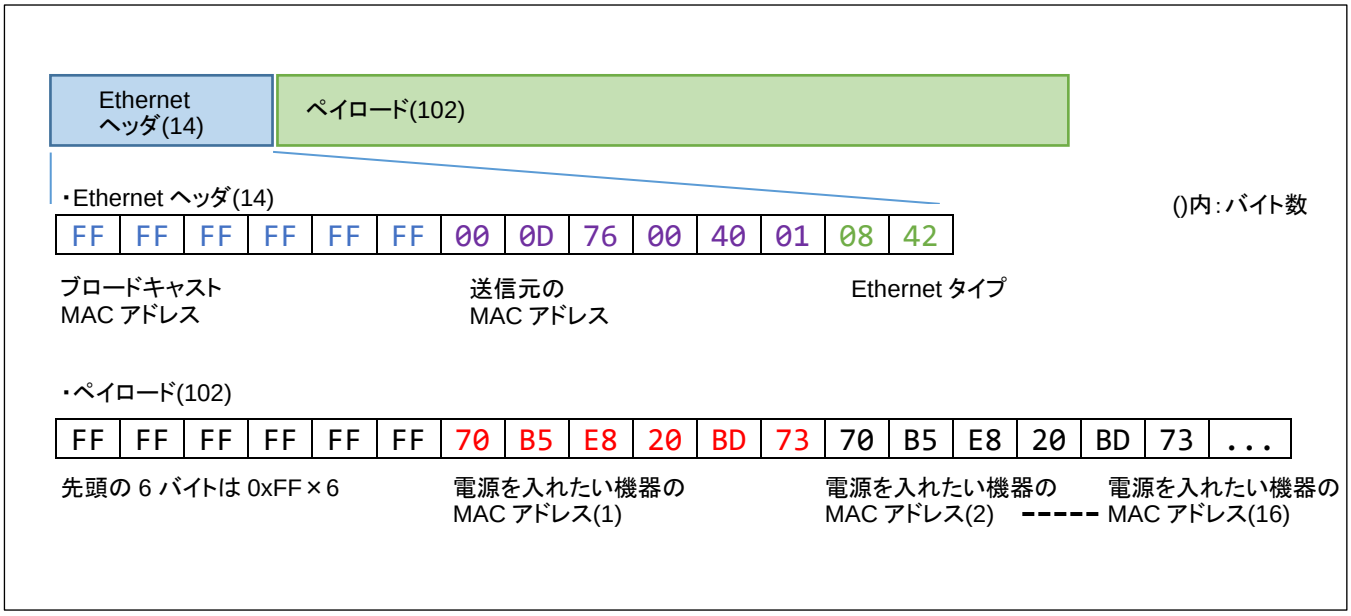


PC の画面に何か変化はありましたでしょうか。実は、PC 側のアプリケーションでは MagicPacket は受信しません。（この時点では、画面に何も表示されないのが期待値です。）

ここで、何が起こっているのか（送信できていないのか、受信に失敗しているのか）を、Wireshark で確認してみます。（Wireshark はフリーのネットワークのパケットダンプソフトです。別途インストールしてください。Wireshark を使用して良いかは、3.2 章に記載していますが、社内のシステム管理部署やネットワーク管理部署に確認を取ってください。）



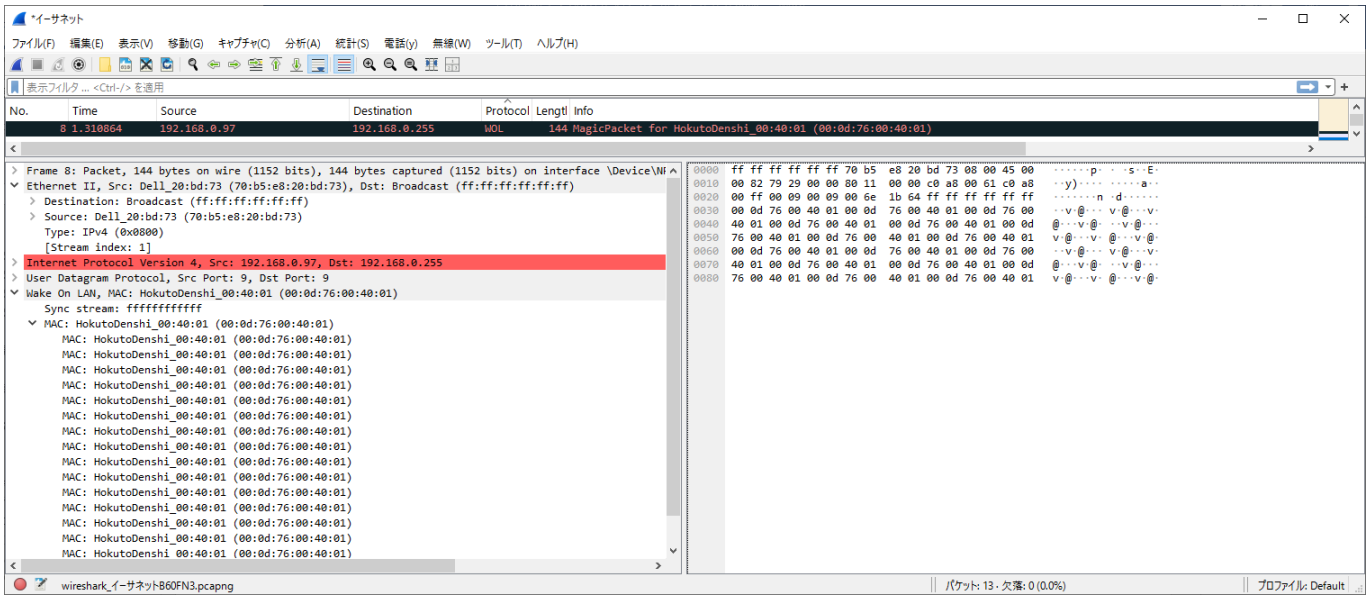
Wireshark で確認すると、RX65N マイコンボードで SW3 を押したタイミングで、



上記のデータが送られている事が確認できます。つまり、RX65N マイコンボード側は送信しているが、PC 側で受信できていない事となります。

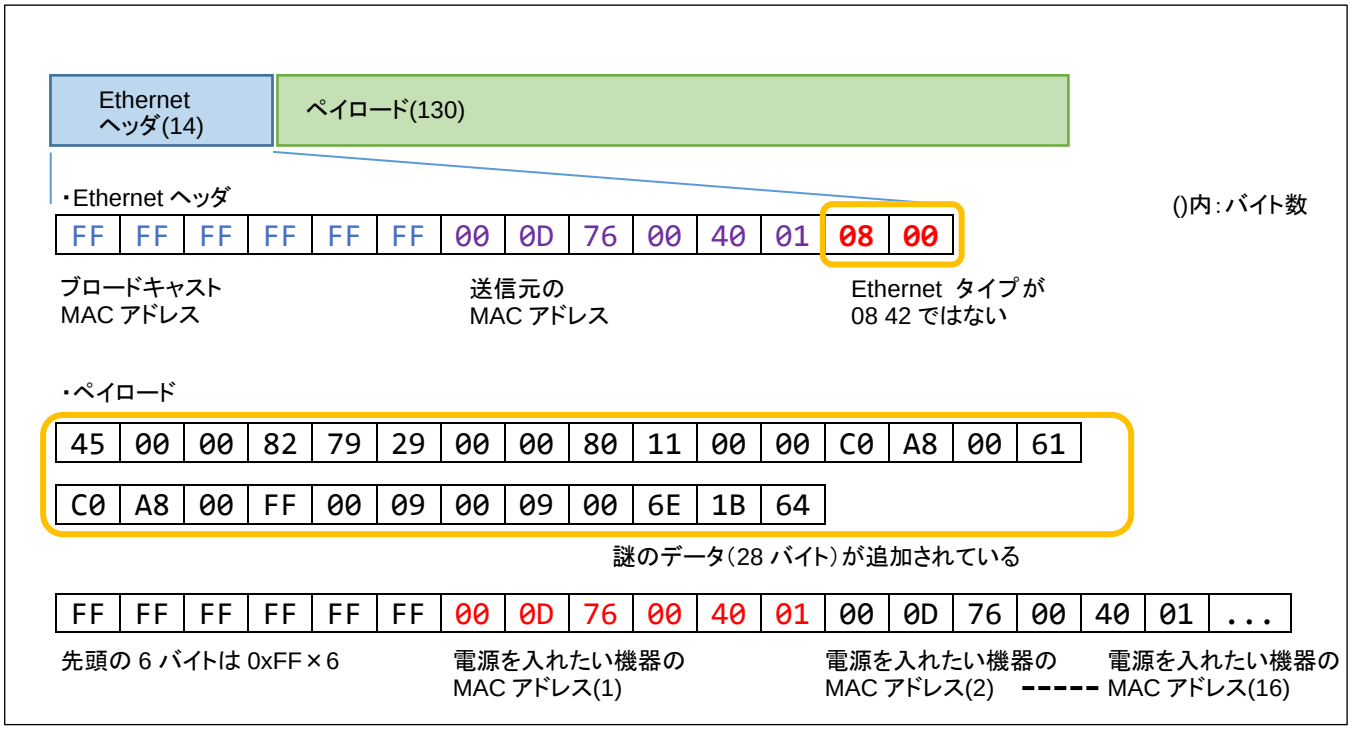
（Wireshark では、Protocol でソートして、WOL のところを見れば MagicPacket の確認ができます。）

今度は、PC 側で、MagicPacketSend.exe で MagicPacket を送信した場合どのように観測されるを見てみます。
（MagicPacketReceive.exe は、受信ストップか、アプリケーションを終了させてください。）



RX65N から MagicPacket を送った場合は、トータルで 116 バイトのデータでした。Ethernet ヘッダが 14 バイト、ペイロードが 0xFF×6 + ターゲットの MAC アドレス（6 バイト）×16 で 102 バイト。合計で、116 バイトです。

それに対し、PC から MagicPacket を送った場合は、トータルで 144 バイトのデータとなっています。



データの中身を見ると、一見 MagicPacket のデータと似ていますが、Ethertnet ヘッダのデータタイプを示すコードが 0x0800 になっていて、かつペイロードの先頭に 28 バイトの謎のデータが挟まっています。ペイロードの後半は、0xFF×6 + MAC アドレス×16 で MagicPacket となっています。

PC から送信した MagicPacket は純粋な MagicPacket ではなく、何か付与されているデータとなっている事が判ります。この謎のデータが何なのかは、後で説明します。

4.3.3. RX65N マイコンボードを 2 台使用した MagicPacket の送受信

RX65N マイコンボードをもう 1 台お持ちであれば、2 台のマイコンボードに同じプログラム(RX65N_MAGIC_PACKET.mot)を書き込み、2 台のマイコンボードを同じハブ（同じハブでなくても問題ありませんが、同一のネットワーク）に接続してください。

○1 台目（送信側とします）

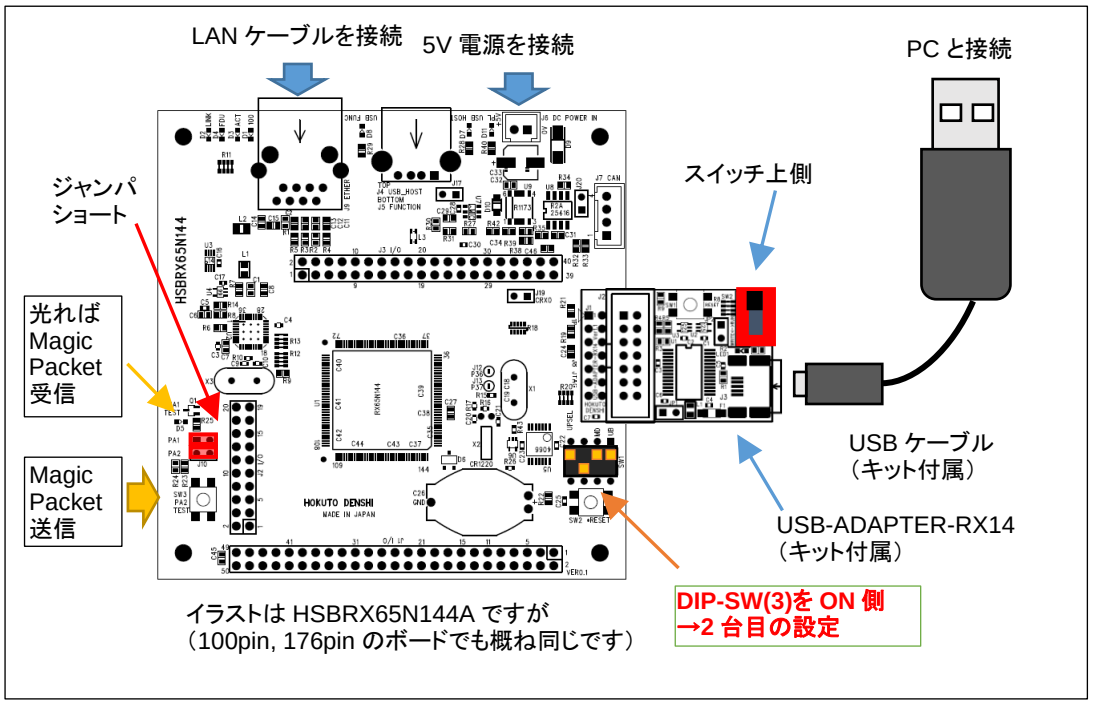
```
-default setting address-
---
target MAC address is       -> 00-0D-76-00-40-02 : m command for change
this board MAC address is -> 00-0D-76-00-40-01 : S command for swap
---
First m, S, command -> initial setting

Please input 's' or 'r' for operation start!
>
>command=s
Operation start with MagicPacket send mode.
PUSH SW3 -> send MagicPacket
Ether0: LINK-UP
```

送信先は
00-0D-76-00-40-02

キーボードから s を入力して、送信待ち状態としてください。

〇2 台目（受信側とします）



※同一ネットワークに RX65N マイコンボードを 2 台接続する場合は、片方の DIP-SW(3)を ON にしてください
その場合は、DIP-SW(3)を ON にした側のボードの MAC アドレス等が 2 台目の設定にセットされます

枠内は DIP-SW(3)を ON にして起動すると設定不要です。

```
>command=S
MAC address swap(this board <-> target)
```

キーボードから S（大文字 S）を入力して、送信 MAC アドレスと受信 MAC アドレスの入れ替えを行ってください。

```
>command=p
target MAC address is       -> 00-0D-76-00-40-01
this board MAC address is -> 00-0D-76-00-40-02
```

キーボードから p を入力すると、MAC アドレスの設定が確認できます。

送信側のデフォルトの送信先は、00-0D-76-00-40-02 なので、2 台目のボードの MAC アドレスを 00-0D-76-00-40-02 に変更します。（DIP-SW(3)を ON にして起動する、または S コマンドを入力する）

```
>command=r

Operation start with MagicPacket receive mode.
this board MAC address = 00-0D-76-00-40-02
Ether0: LINK-UP
-> MagicPacket waiting ...
```

キーボードから r を入力して受信状態とします。

この状態で 1 台目のマイコンボードの SW3 を押してください。

〇1 台目

```
MagicPacket data send (00-0D-76-00-40-02)
```

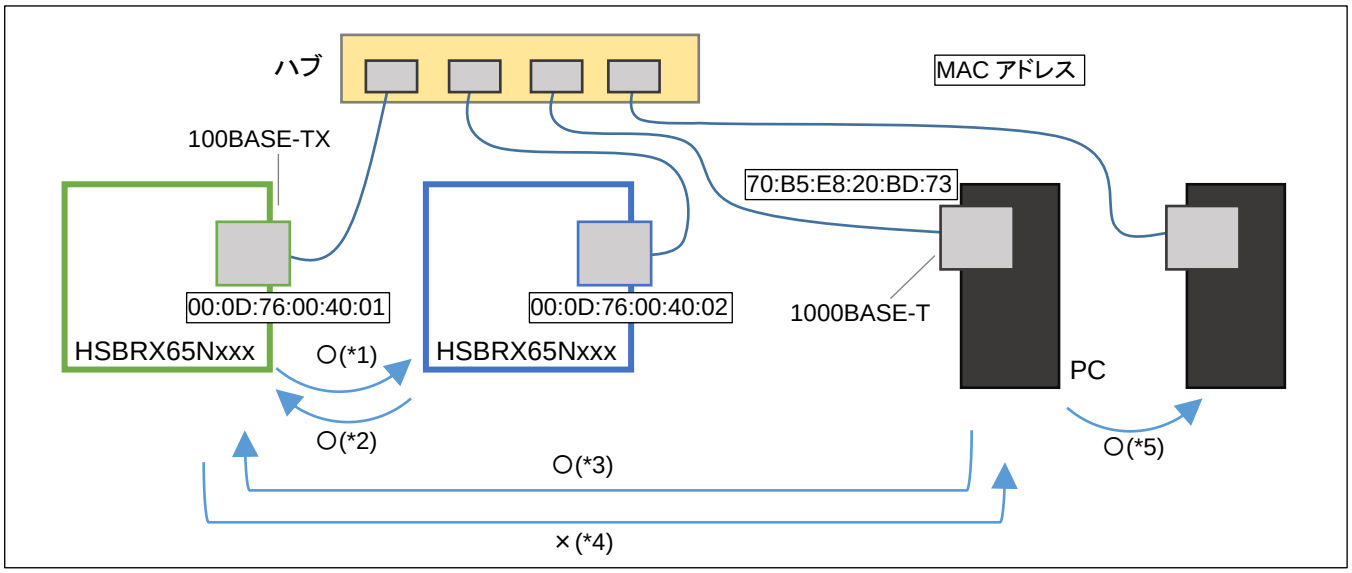
〇2 台目

```
Ether0: MagicPacket received.
Ether0: LINK-UP
-> MagicPacket waiting ...
```

2 台目のボードの LED(D5)が点灯し、MagicPacket received.のメッセージが表示されれば、1 台目→2 台目の MagicPacket の送信・受信は成功しています。

ここでは、RX65N マイコンボード同士であれば、MagicPacket の送受信が問題なく行える事が確認できました。

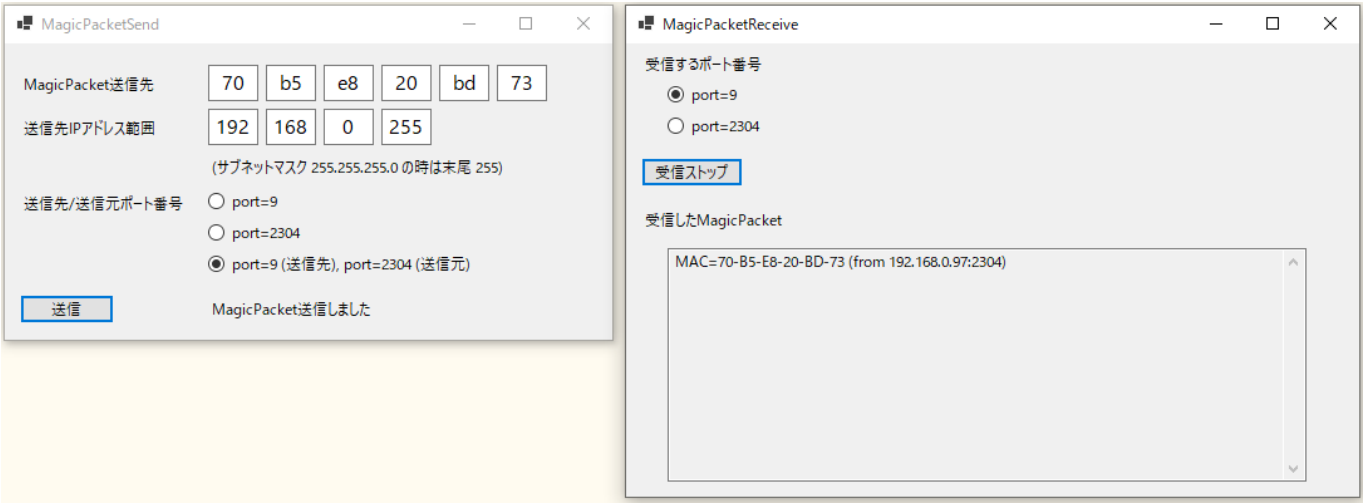
4.3.4. RX65N マイコンボードと PC の MagicPacket の相違点



送受信ができるケースが〇できないケースが×です。

	送信側	受信側	送受信	送信データ
(*1)	マイコンボード	マイコンボード	〇	116 バイト
(*2)	マイコンボード	マイコンボード	〇	116 バイト
(*3)	PC	マイコンボード	〇	144 バイト
(*4)	マイコンボード	PC	×	116 バイト
(*5)	PC	PC	〇	144 バイト

(*5)の実行例 （※この例では、1 台の PC で MagicPacketSend.exe/MagicPacketReceive.exe を動かしてます）



(*5)は、2 台の PC で、MagicPacketSend.exe と MagicPacketReceive.exe を動かして通信させたケースです。この場合は、通信が通ります。

※2 台の PC 間で MagicPacket のやり取りを行う場合はポート番号はデフォルト（送信側 port=9，受信側 port=9）で問題ありません。1 台の PC で実行する場合は、送信元ポート番号と受信するポート番号が重複しない様に設定してください。）

この結果から、

・マイコンボードが送信するのは 116 バイトのデータ

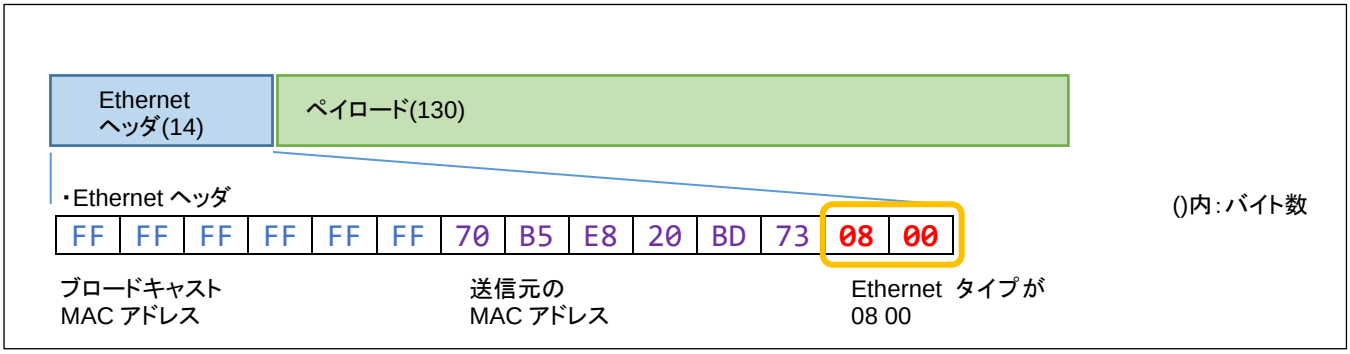
・PC が送信するのは 144 バイトのデータ

・マイコンボードは、116 バイト、144 バイトどちらのデータも受信する

・PC は 144 バイトのデータのみ受信する（116 バイトのデータは受信しない）

という事となります。

－PC が送信する Ethernet フレーム－

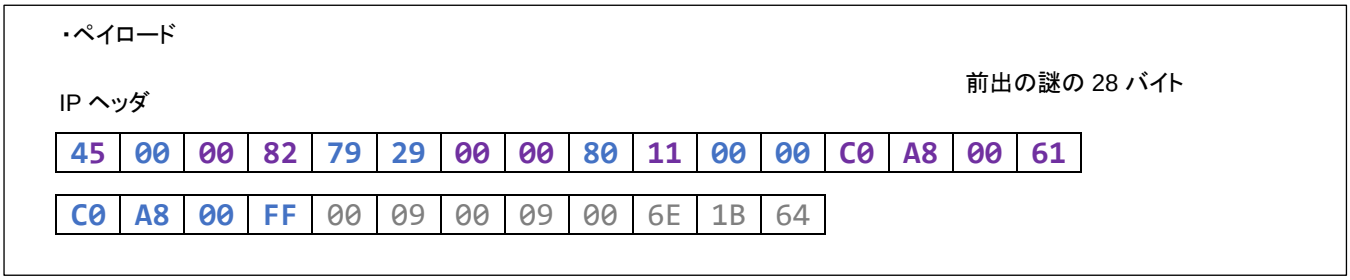


〇Ethernet ヘッダ（14 バイト）

フィールド	bit 数	内容	データ例
宛先 MAC アドレス	48	通信先の MAC アドレス (FF-FF-FF-FF-FF-FF の場合はブロードキャスト →全てのネットワークインタフェースに対する通信)	FF FF FF FF FF FF
送信元 MAC アドレス	48	データを送信したインタフェースの MAC アドレス	00 0D 76 00 40 01
Ethernet タイプ	16	データ種別を示すコード	08 00

ここで、PC が送信している Ethernet フレームについて詳しく見てみる事とします。

Ethernet タイプが 0x0800 となっていますが、これは IP パケットである事を示しています。



上図の太文字の部分が IP ヘッダと呼ばれるデータで 20 バイトのデータとなります。

IP に関しては 5 章で詳細に説明します。ここでは、こんな感じのデータであるという認識で問題ありません。

〇IP ヘッダ（20 バイト）

フィールド	bit 数	内容	データ例
Version	4	Internet protocol version 4(IPv4)	45
Length	4	IP ヘッダのサイズ（4×5 バイト）	45
Service Type	8	パケットの優先度	00
Total Length	16	IP データグラムの長さ	00 82 （=130 バイト）
Identifier	16	パケットの識別子（固有の ID）	79 29 （都度変わる）
Flags	3	パケット分割有無	00 00 =0b000 0b0000000000000
Flagment Offset	13	分割されたパケットの位置	
TTL(Time-to-Live)	8	ルータ中継回数上限	80 （=127 回）
Protocol	8	上位プロトコルの種類	11 （=10 進数で 17, UDP）

Checksum	16	チェックサム	00 00 （チェックサムは省略）
Source IP Address	32	送信元 IP アドレス	C0 A8 00 61 (=192.168.0.97)
Destination IP Address	32	送信先 IP アドレス	C0 A8 00 FF (=192.168.0.255) (*1)

(*1)の部分が MagicPacketSend の赤線部に対応します

純粋な MagicPacket は、IP アドレスとは無関係ですが PC が送信する MagicPacket は IP アドレスと関係しています

28 バイトの謎のデータの後半部分ですが、これは UDP ヘッダと呼ばれるデータとなります。(IP ヘッダの Protocol のところが、0x11(=17)になっているので、IP ヘッダに続くのは UDP のデータである事を示しています。)

・ペイロード

前出の謎の 28 バイト

IP ヘッダ

45	00	00	82	79	29	00	00	80	11	00	00	C0	A8	00	61
C0	A8	00	FF	**00**	**09**	**00**	**09**	**00**	**6E**	**1B**	**64**				

UDP ヘッダ

上図の太文字の部分が UDP ヘッダと呼ばれるデータで 8 バイトのデータとなります。

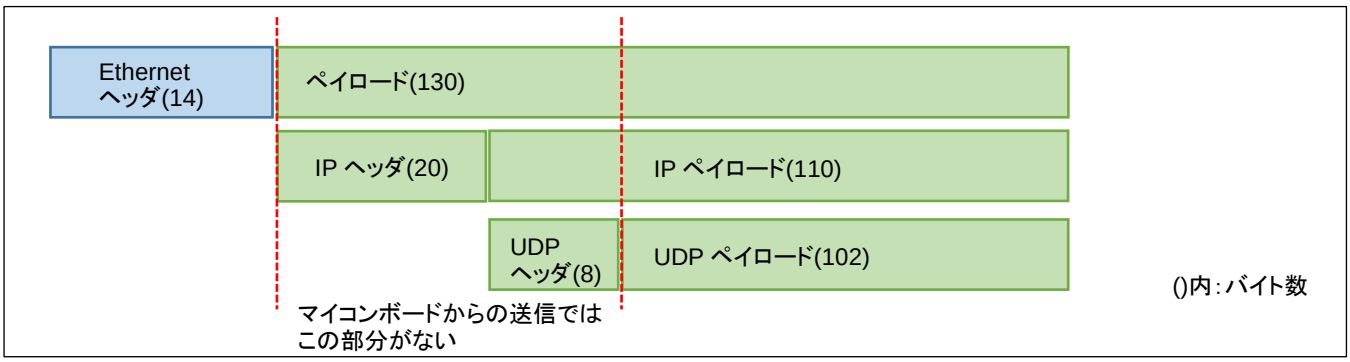
UDP に関しては 6 章で詳細に説明します。ここでは、こんな感じのデータであるという認識で問題ありません。

○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	00 09 （9 番ポートから送信）
Destination Port	16	送信先ポート	00 09 （9 番ポートに対して送信）
Length	16	データ長	00 6E （=110 バイト）
Checksum	16	チェックサム	1B 64

※Length は UDP ヘッダと UDP データ本体（102 バイト）を合わせた値です

－PC から送信した MagicPacket の構造－



PC から送信した MagicPacket は、純粋な MagicPacket に対し、IP ヘッダと UDP ヘッダが追加されており、MagicPacket のデータが UDP ペイロードに格納されて送信されているという事が、Wireshark のパケットダンプから見て取れます。

この PC から送信した MagicPacket は、IP(InternetProtocol)の UDP パケットと呼ばれる形式で送信されており、PC 上でネットワークを利用するプログラムを作成する場合は、純粋な MagicPacket を送信する（＝生の Ethernet フレームを取り扱う）のは簡単ではなく、IP/UDP といったプロトコルに MagicPacket のデータを乗せて送信する方が楽なので、この様な形になっています。

（PC から純粋な MagicPacket を送信する方法に関しては後述します。）

ここでは、
・PC では IP, UDP 等のプロトコルを使った通信方法が一般的である
・マイコンボードは IP や UDP などのプロトコルに囚われずに自由な Ethernet パケットを取り扱う事ができる
という理解で良いと思います。

※ここでは、IP, UDP に関しての詳細な説明は行いません。後に、IP や UDP を使用した通信があるので、その際に IP, UDP に関して詳細に説明します。ここでは、純粋な MagicPacket（116 バイトのデータ）に、28 バイトのデータを付加して、144 バイトのデータにすると、PC で受信できるようになるという認識で OK です。

ープロトコルとはー
決まった通信手順の事を示します。通信の世界のお約束の様なものです。通信は、2 者間で行うものですので、送信側と受信側で好き勝手にはできません。予め決められたルールに則って送信、受信を行う必要があります。

ここで、PC 側は、144 バイト（Ethernet フレーム上に IP ヘッダと UDP ヘッダが追加され、UDP のプロトコル）のデータは受信できる事が判りましたので、マイコン側のプログラムを修正して 144 バイトのデータを送信できる様にしてみましょう。

なお、MagicPacket(WOL)は元々遠隔で PC 等の装置の電源を投入する仕組みです。

PC が通常動作（Windows が動いている）場合は、純粋な MagicPacket（116 バイト）は受信しませんが、
・BIOS で WOL の設定を有効化する
・PC の MAC アドレスを予めメモしておく
・PC の電源を切っておく
・RX65N マイコンボードから PC の MAC アドレスに対して MagicPacket を送信する
という手順で、PC の電源投入が行えるはずです。
（但し、PC 側が BIOS，ネットワークインタフェースレベルで WOL に対応している必要があり、PC が Wifi ではなく、LAN ケーブルでネットワークに接続されている必要があります。）

4.4. PC に対しての MagicPacket(WOL)の送信

RX65N マイコン向けのプロジェクト、RX65N_MAGIC_PACKET2 をマイコンボードに書き込んでください。書き込みの手順は、4.2 章と同じです。（CS+, e2studio でのダウンロードか、RFP を使った書き込み）
（書き込むファイルは、RX65N_MAGIC_PACKET2¥DefaultBuild¥RX65N_MAGIC_PACKET2.mot です）

マイコンボードを起動すると、端末には下記の表示が出ます。

```
RX65N Ether MagicPacket(WOL) sample program2.

COMMAND:
   m : target MAC address set
   i : target broadcast IP address set
   j : this board IP address set
   p : print MAC/IP address
   S : swap [this board MAC address] <-> [target MAC address]
   W : send packet type is WOL.
   I : send packet type is IP.(default)
---
   s : Operation start with send/receive mode
   r : Operation start with receive only mode
---

USAGE:
   SW3 : MagicPacket send
   LED : MagicPacket received

-default setting address-
---
target MAC address is         -> 00-0D-76-00-40-02 : m command for change
this board MAC address is -> 00-0D-76-00-40-01 : S command for swap
target IP address is          -> 192.168.0.255 : i command for change
this board IP address is    -> 192.168.0.80 : j command for change
---
First m, i, j, p, S, W, I command -> initial setting

Please input 's' or 'r' for operation start!
```

赤字の部分が変更点

初期設定が、m, i, j, p, S, W, I コマンドです。
動作開始が、s と r コマンドです。

まずは、m コマンドで MagicPacket の送信先である PC の MAC アドレスを指定します。

```
>command=m
target MAC address set(please input 2 letters(0-9,a-f,A-F) hex x 6)

MAC[0] >70
MAC[1] >b5
MAC[2] >e8
MAC[3] >20
MAC[4] >bd
MAC[5] >73
MAC address set to -> 70-B5-E8-29-BD-73
```

次に、PC の IP アドレスが、192.168.0.???ではない場合は、j コマンドで送信先のブロードキャストアドレスを設定します。

（例 PC の IP アドレスが 192.168.100.3 の場合）

```
>command=j
this board IP address 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)  >100
IP[3](b7-0)   >255
IP address set to -> 192.168.100.255
```

PC の IP アドレスが、192.168.100.3 の場合は、ブロードキャストアドレスは 192.168.100.255 となります。

また、この場合、RX65N マイコンボードの IP アドレスの変更も必要になります（i コマンドで変更）。

```
>command=i
target broadcast IP address 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)  >100
IP[3](b7-0)   >80
IP address set to -> 192.168.100.80
```

入力が 3 桁に満たない場合は、80[リターン]と入力

PC と同じネットワークに属するアドレス、かつ、他の機器で使用されていないアドレスを指定してください。

p コマンドで設定したアドレスを確認します。

```
>command=p
target MAC address is       -> 70-B5-E8-20-BD-73
this board MAC address is -> 00-0D-76-00-40-01
target IP address is        -> 192.168.100.80
this board IP address is    -> 192.168.100.255
```

なお、以下では 192.168.0.97 に対して MagicPacket を送るので、

```
>command=p
target MAC address is       -> 70-B5-E8-20-BD-73
this board MAC address is -> 00-0D-76-00-40-01
target IP address is        -> 192.168.0.80
this board IP address is    -> 192.168.0.255
```

上記設定とします。

上記の他に、送信するデータ形式として、純粋な MagicPacket（116 バイト、RX65N_MAGIC_PACKET プロジェクトと同じ形式）（W コマンドで選択）と、IP/UDP パケットで送信する MagicPacket（144 バイト、PC と同じ形式）（I コマンドで選択）が選択できますが、今回は IP/UDP パケットで送信するのでデフォルトのままなので、設定は不要です。

（あえて設定する場合は、I（大文字の i）コマンドです。）

```
>command=W
Send MagicPacket is WOL(ethernet type=0824).

>command=I
Send MagicPacket is IP(UDP)(ethernet type=0800).
```

今回はこちらを使用（起動時のデフォルト）

IP アドレスに関しては、5 章で解説します。

p コマンド

```
>command=p
target MAC address is         -> 70-B5-E8-20-BD-73
this board MAC address is -> 00-0D-76-00-40-01
target IP address is           -> 192.168.0.80
this board IP address is    -> 192.168.0.255
```

今回のマイコンボード側の設定は上記です。PC 側では、MagicPacketReceive.exe を起動して、受信スタートボタンを押します。

マイコンボード側では、s コマンドを入力して送信（&受信）をスタートします。

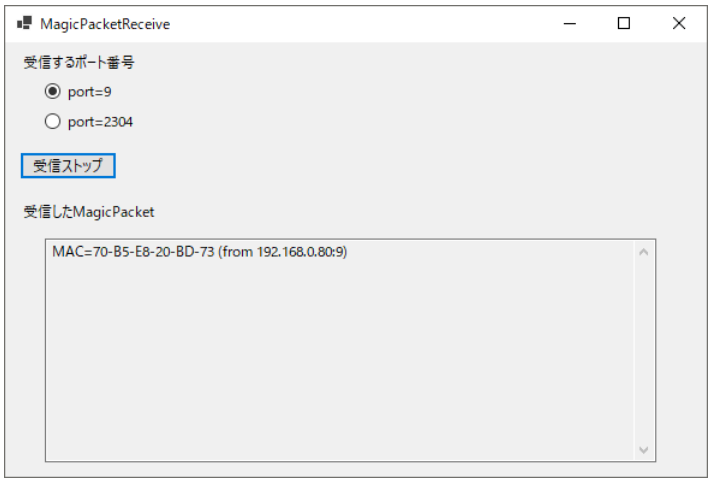
```
>command=s

Operation start with MagicPacket send/receive mode.
PUSH SW3 -> send MagicPacket
this board MAC address = 00-0D-76-00-40-01
Ether0: LINK-UP
```

マイコンボードの SW3 を押します。

```
MagicPacket(IP/UDP) data send (MAC:70-B5-E8-20-BD-73) (IP:192.168.0.255)
```

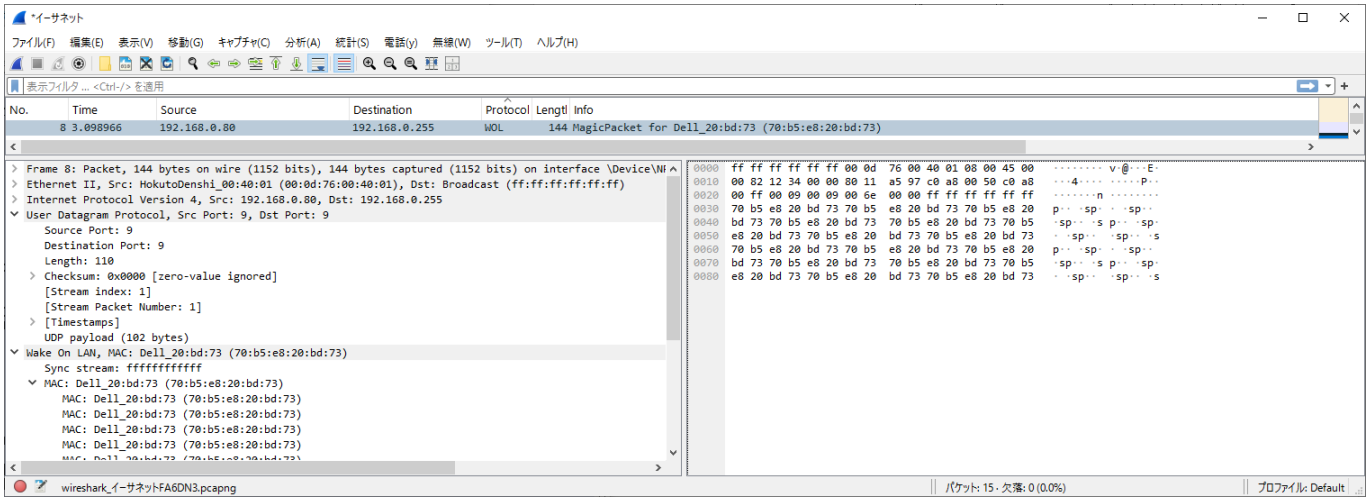
ここでは、Ethernet フレームのペイロードに IP ヘッダ+UDP ヘッダ+MagicPacket データを埋め込んだ形（PC で受信可能な形式）で送信します。



この場合は、PC でも MagicPacket を受信します。
（この例では、送信元は、192.168.0.80:9 となっています。）

※:9 はポート番号を示します。ポート番号に関しては、後の UDP の説明で解説します。

この通信（RX65N マイコンボードが送信した MagicPacket）を、Wireshark で取り込むと、



の様になっており、144 バイトで PC が送信したデータ形式と同じである事が見て取れます。この場合は、PC 側でも MagicPacket の受信が可能です。

なお、RX65N_MAGIC_PACKET2 では、s コマンドで動作開始した場合でも、MagicPacket の受信が可能です。PC から MagicPacketSend.exe で送信すると

マイコンボード側では、MagicPacket(2)を受信します。MagicPacket(2)は、IP/UDP 形式の 144 バイトの MagicPacet です。

```
MagicPacket(2) received.
```

ネットワークにもう 1 台 RX65N マイコンボードを接続した場合、

DIP-SW(3)=ON の時は不要

・2 台目の RX65N マイコンボード

```
>command=S
MAC address swap(this board <-> target)
```

S コマンド
MagicPacket 送信先を
00-0D-76-00-40-01 に変更

```
>command=W
Send MagicPacket is WOL(ethernet type=0824).

>command=s

Operation start with MagicPacket send/receive mode.
PUSH SW3 -> send MagicPacket
this board MAC address = 00-0D-76-00-40-02
Ether0: LINK-UP
MagicPacket(WOL) data send (00-0D-76-00-40-01)
```

W コマンド
MagicPacket 送信フォーマットを純粋な MagicPacket（116 バイト）に変更

s コマンド
動作スタート

SW3 を押して MagicPacket 送信

・1 台目の RX65N マイコンボード

```
MagicPacket(1) received.
```

MagicPacket(1) received.となり、これは純粋な 116 バイトの純粋な MagicPacket を受信したことを示します。

RX65N_MAGIC_PACKET2 プロジェクトでは、起動後の初期設定の段階で、

W : 純粋な MagicPacket（116 バイト）を送信…マイコンボードは受信〇、PC は受信×
I: IP/UDP 形式の MagicPacket（144 バイト）を送信…マイコンボード、PC 共に受信〇 （大文字 i コマンド）

W 及び I コマンドで送信フォーマットを選べます。

RX65N_MAGIC_PACKET2 において、
s コマンド　送信/受信
r コマンド　受信のみ
となっています。s コマンドで受信もできるので、基本的には r コマンドは不要です。
（s コマンドと、r コマンドの受信動作の違いは、ソフトウェア編で説明しています。見た目上の動作は、s コマンドでも r コマンドでも、そう変わりはありません。）

4.5. PC で純粋な MagicPacket を送受信する方法[参考]

4.4 節まででは、
・PC から送信する MagicPacket→144 バイト（IP/UDP 形式）
・PC が受信可能な MagicPacket→144 バイト（IP/UDP 形式）
という説明でした。
PC で純粋な MagicPacket（116 バイト）を送受信する方法ですが、Wireshark をインストール済みであれば、容易に実現できます。
※正確には、Wireshark をインストールした際に、同時にインストールされる winpcap というライブラリが必要です

マイコンボードには、RX65N_MAGIC_PACKET2.mot を書き込んでおきます。
（RX65N_MAGIC_PACKET.mot でも OK です）

マイコンボード起動後に、
・m コマンドで PC の MAC アドレスを設定

```
>command=m
target MAC address set(please input 2 letters(0-9,a-f,A-F) hex x 6)

MAC[0] >70
MAC[1] >b5
MAC[2] >e8
MAC[3] >20
MAC[4] >bd
MAC[5] >73
MAC address set to -> 70-B5-E8-20-BD-73
```

・W コマンドで WOL パケットを送信する様に設定

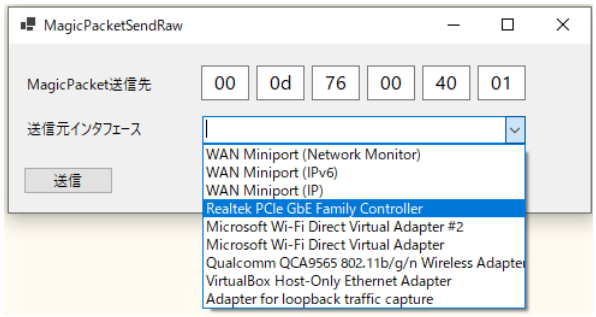
```
>command=W
Send MagicPacket type is WOL(ethernet type=0824).
```

・s コマンドで送受信動作開始

```
>command=s

Operation start with MagicPacket send/receive mode.
PUSH SW3 -> send MagicPacket
this board MAC address = 00-0D-76-00-40-01
Ether0: LINK-UP
```

PC 側では、
MagicPacketSendRaw.exe
を起動。



送信元インタフェースとして、マイコンボードが接続されているネットワークインタフェース（この場合は、LAN ケーブルがささっている有線のアダプタ）を選択。
（Intel xxxx とか Realtek xxxx などが、有線の LAN アダプタです）

MagicPacket 送信先は、マイコンボードの MAC アドレスを変更している場合は変更。

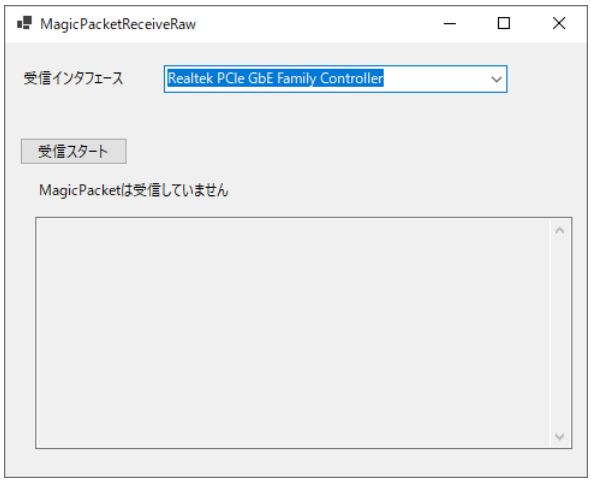
送信ボタンを押す。

マイコンボード側で、

```
MagicPacket(1) received.
```

"MagicPacket(1) received."となれば、116 バイトの純粋な MagicPacket を受信しています。
（144 バイトの MagicPacket を受信した場合は、"MagicPacket(2) received."となります。）

PC で、
MagicPacketReceiveRaw.exe
を起動します。

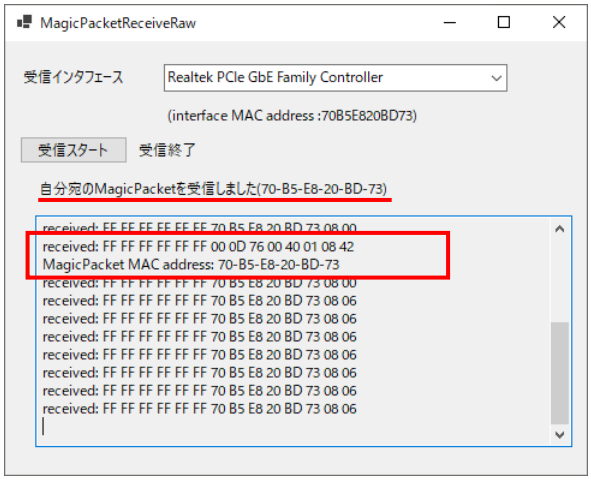


受信インタフェースを、マイコンボードが接続されているネットワークインタフェースを選択して、受信スタートを押してください。

マイコンボードの SW3 を押してください。

```
MagicPacket(WOL) data send (70-B5-E8-20-BD-73)
```

受信スタートを押してから、10 秒間受信データをキャプチャします。10 秒以内に、マイコンボードの SW3 を押してください。10 秒経過すると、受信動作は終了します。



このアプリでは、受信したデータの、Ethernet ヘッダ（14 バイト）を表示します。受信するデータは、ブロードキャストアドレス(FF-FF-FF-FF-FF-FF)か自分宛のデータです。

Ethernet ヘッダの 13~14 バイト目(Ethernet type)が、0x0842 の場合、MagicPacket であると判断して、
Magic Packet MAC address: xx-xx-xx-xx-xx-xx
の表示が出ます。（自分宛の MagicPacket でなくても、Ethernet type が 0842 の場合に表示されます。）
MagicPacket のデータが自分宛の場合は、「自分宛の MagicPacket を受信しました」と表示されます。

本節で使用している PC アプリ、
MagicPacketSendRaw.exe
MagicPacketReceiveRaw.exe
には、IP アドレス（という概念）が登場しません。純粋に、Ethernet フレームのやり取りを行うアプリです。

※本アプリは、winpcap がインストールされている事が前提となっています

PC の世界では、純粋な MagicPacket（単純な Ethernet フレーム）を取り使うのは多少手間が掛かり、逆に IP アドレスを使用した通信（Ethernet フレームとしては複雑）は容易となっています。OS レベルで IP 通信をサポートしているために、IP 通信は C#(.NET)の API 関数を呼び出すだけで実現できます。それに対し、Ethernet フレームを直接操作するような処理となると、.NET の標準機能外の使い方となります。

4.6. RX65N_ETHER_NOAPI プロジェクトに関して

RX65N_ETHER_NOAIP プロジェクトの動作に関して説明します。
（書き込むファイルは、RX65N_ETHER_NOAPI¥DefaultBuild¥RX65N_ETHER_NOAPI.mot です）

プログラムの動作としては、RX65N_MAGIC_PACKET2 プロジェクトとほぼ同等です。

マイコンボードを起動すると、端末には下記の表示が出ます。

```
RX65N Ether MagicPacket(WOL) sample program no API version.

COMMAND:
   m : target MAC address set
   i : target broadcast IP address set
   j : this board IP address set
   p : print MAC/IP address
   S : swap [this board MAC address] <-> [target MAC address]
   W : send packet type is WOL.
   I : send packet type is IP.(default)
---
   s : Operation start with send/receive mode
   r : Operation start with receive only mode
---

USAGE:
   SW3 : MagicPacket send
   LED : MagicPacket received

-default setting address-
---
target MAC address is         -> 00-0D-76-00-40-02 : m command for change
this board MAC address is -> 00-0D-76-00-40-01 : S command for swap
target IP address is          -> 192.168.0.255 : i command for change
this board IP address is    -> 192.168.0.80 : j command for change
---
First m, i, j, p, S, W, I command -> initial setting

Please input 's' or 'r' for operation start!
>
```

初期設定が、m, i, j, p, S, W, I コマンドです。動作開始が、s と r コマンドです。
→RX65N_MAGIC_PACKET2 と同じ。

RX65N_MAGIC_PACKET2 との相違点は、プログラムの作り方にあります。RX65N_MAGIC_PACKET2 は、Ethernet を動かす部分はスマート・コンフィグレータの追加コンポーネント(r_ether_rx)を使い、Ether のデータ送受信は r_ether_rx の API 関数を使用していました。

それに対し、本プロジェクトは r_ether_rx は使わず、Ethernet の機能を自前のコードで動かそうという目的です。

r コマンドでの動作開始

```
>command=r

Operation start with MagicPacket receive only mode.
this board MAC address = 00-0D-76-00-40-01
Ether0: LINK-DOWN
Ether0: LINK-UP
-> MagicPacket waiting ...
```

MagicPacket 受信待ちとなります。この状態で本ボードに対する MagicPacket を受信した場合は、

```
Ether0: MagicPacket received.

-- MagicPacket receive operation is stopped.
   (This program is not reboot ETHERC Interface.)
```

となります。MAGIC_PACKET2 では、この時点で Ethernet インタフェースを一旦ダウンさせて再度 MagicPacket 受信待ちとなりますが、本プログラムでは動作を停止します。（再度動作させるためには、リセットか電源再投入を行ってください。）

s コマンドでの動作開始

```
>command=s

Operation start with MagicPacket send/receive mode.
PUSH SW3 -> send MagicPacket
target MAC address        = 00-0D-76-00-40-02
target IP address         = 192.168.0.255
this board MAC address = 00-0D-76-00-40-01
Ether0: LINK-DOWN
Ether0: LINK-UP
```

送受信可能なモードです。SW3 で MagicPacket の送信。

```
MagicPacket(IP/UDP) data send (MAC:00-0D-76-00-40-02) (IP:192.168.0.255)
```

MagicPacket を受信した場合は、

```
MagicPacket(1) received.
```

となります。

また、本プログラムでは、30 秒毎に、

```
read_data / receive_data = 119 / 119
read_data / receive_data = 245 / 245
read_data / receive_data = 372 / 372
read_data / receive_data = 452 / 452
```

受信データ処理ルーチンで処理したデータの個数(read data)
受信割り込みに飛んできた回数(receive data)

の個数を表示する様になっています。この 2 つの数値が合っていない場合は、受信データの処理（＝受信データを見て、自分宛の MagicPacket であるかどうか判断）が間に合っていないという事を意味します。

これは、データを受信した際の割り込みでの処理や、受信に使用する受信ディスクリプタの取り扱い。受信バッファの読み出しと解放などを自前でやっているので、目安として表示させています。

ソフトウェア編マニュアルで、どのような処理を行っているかは記載しています。

スマート・コンフィグレータ（正確には RX Driver Package）で用意されている、Ethernet ドライバ(r_ether_rx)を使用せずにプログラムを書くと、どのような処理が必要となるかのイメージを持つためのプロジェクトです。

既存の API 関数を使用して、上手く動作している場合は API 関数が裏でどのような処理を行っているかはあまり理解していなくても（望ましくはないかもしれませんが）問題はないかと思います。

プログラムを作成して、デバッグする過程で、大抵は何らかの不具合に遭遇するのではないかと思います。その際に、API 関数（＝他人が作成したプログラム）を使用していると、不具合の原因がなかなか判らず苦労する事が多い様に思います。

他人が作成したプログラムを使わず、一度自前でプログラムを書いた事があれば、デバッグの際に見るべきポイントが判ってくるはずです。（ディスクリプタ内の値がおかしい。ディスクリプタ内に書かれている受信バッファのアドレスにデータが格納されるところまでは動いている。割り込みフラグは立っているが、割り込み許可の設定が抜けている等）

ともかく、Ethernet の処理を自前で書いてみる、というのが目的のプロジェクトです。ネットワーク的には新規要素はないので、必要が無ければプロジェクトの中身は見ずに先へ進んで頂いて問題ありません。

5. ping の応答

5.1. IP(InternetProtocol)ネットワークに関して

純粋な MagicPacket の送信では、送信元や送信先の MAC アドレスが Ethernet ヘッダに含まれる形でした。IPアドレスという概念が存在しない世界です。

それに対し、PC がやり取りをする MagicPacket では、IP ヘッダや UDP ヘッダが含まれる形で、IP アドレスという概念が存在しました。

現在、純粋な Ethernet 通信が使われるのは MagicPacket 位で、一般的にイーサネットといえば IP アドレスを使って通信を行う事を意味しています。

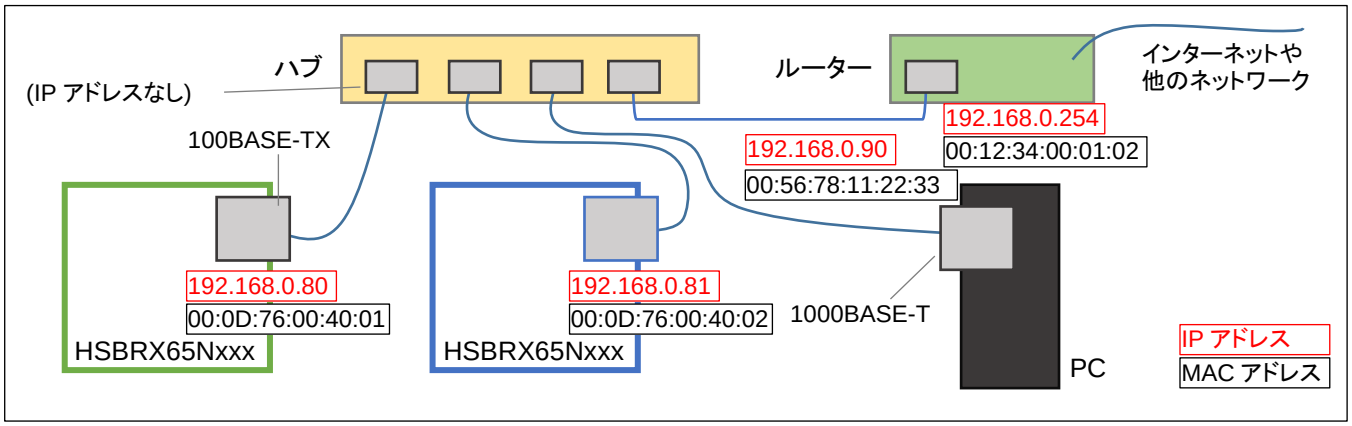


図 5-1 IP ネットワーク接続形態

IP ネットワークは、各インタフェースに IP アドレスが割り振られた形になります。同一ネットワーク内に同一の IP アドレスが複数存在するのは NG です。

※上図では、1 つのネットワークインタフェースに 1 つの IP アドレスが割り振られた形の図としていますが、1 つのネットワークインタフェースに複数の IP アドレスが割り振られるのは OK です

ーIP アドレスに関してー

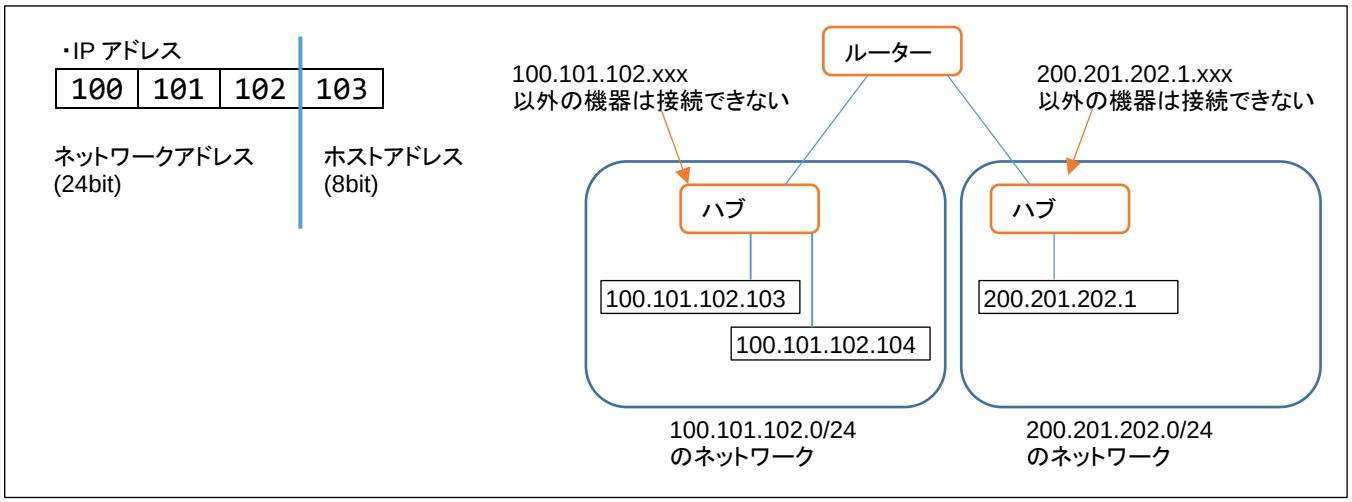
IP アドレスは、192.168.0.1 の様に表記されますが、これは IPv4(InternetProtocolVersion4)のアドレス表記です。IPv4 ではアドレスを 32bit で表現します。32bit は、2^{32}=約 42 億台の機器に別々のアドレスを割り振って識別可能という事です。ネットワークやインターネットが普及し始めた時は、インターネットに接続される機器の台数が 2^{32} で足りるという目論見でしたが、現在ではネットワークに接続される機器は 42 億台を軽く超えていますので、IPv4 のアドレス空間では不足しています。そのため、もっと新しいバージョンのネットワークである IPv6(InternetProtocolVersion6)に移行しつつあります。IPv6 では、2^{128}=3.4×10^{38} 台の機器（ちょっと天文学的な台数）に個別のアドレスを割り振って認識する事が可能です。

本キットでは、IP アドレスは IPv4 を使う事とします。

MAC アドレスは、00:0D:76:00:40:01（16 進数表記）でしたが、何故か IP アドレスは 10 進数で表記するのが一般的です（IPv4 の場合）（なお、IPv6 の場合は 16 進数表記です）。IP アドレスの範囲は、0.0.0.0～255.255.255.255 までです。8bit ずつ.（ドット）で区切って表記します。各桁は 0~255 までで、300.300.300.300 の様なアドレスは存在しません。

（0b11111111 = 0xFF = 255, 2 進数表記, 16 進数表記, 10 進数表記）

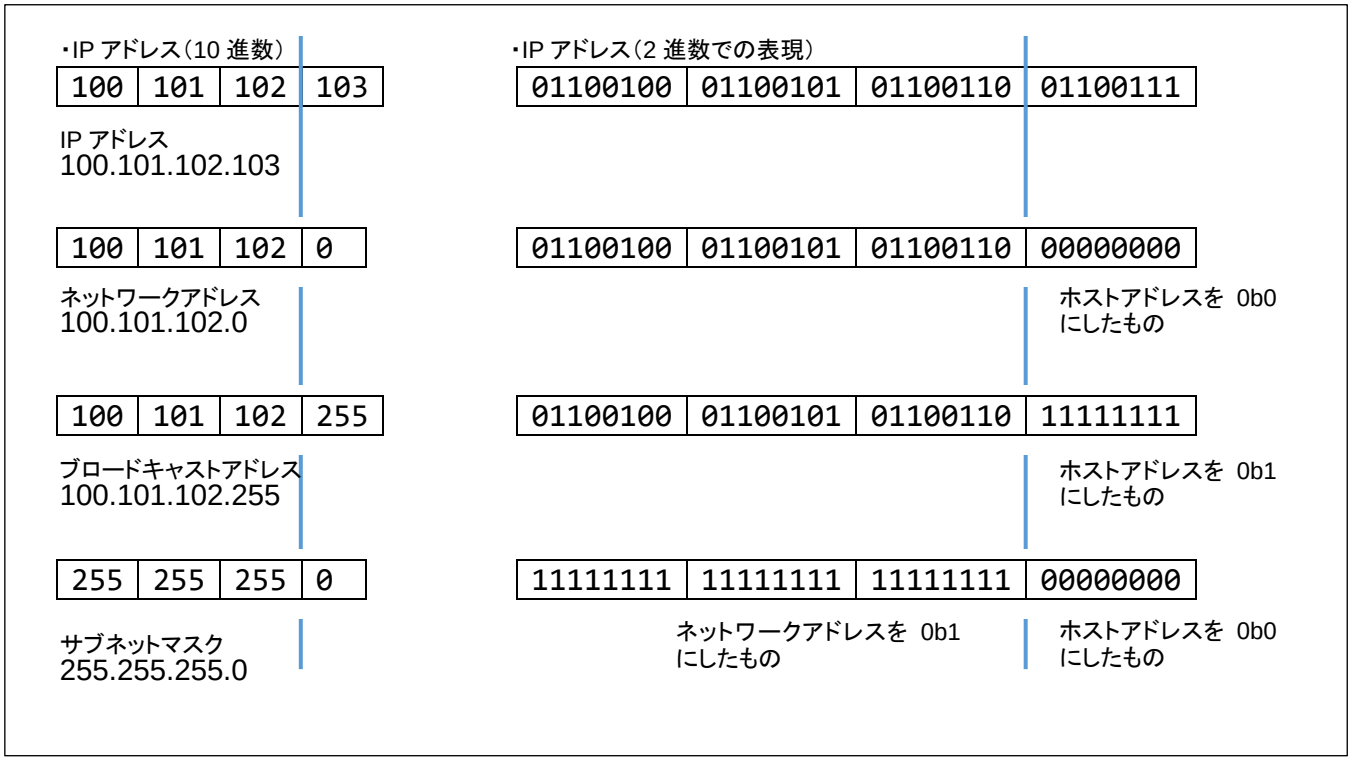
IP アドレスは 32bit ですが、前半がネットワークアドレス、後半がホストアドレスといった様に 2 つに分解されます。



ネットワークアドレスとホストアドレスの区切り位置は任意ですが、ネットワークアドレスが 24bit, ホストアドレスが 8bit の場合、

100.101.102.103/24

の様に、/24 でネットワークアドレスが 24bit であることを表現します。



ここで、ホストアドレス部を 2 進数の 0(0b0)としたアドレスは特別な意味を持ちネットワークアドレスと呼びます。この場合は、100.101.102.0 というアドレスです。同じく、ホストアドレス部を 2 進数の 1(0b1)としたアドレスは、ブロードキャストアドレスと呼び、この場合は、100.101.102.255 です。この 2 つのアドレス

100.101.102.0

100.101.102.255

は、機器の IP アドレスとしては使用できません。機器に設定可能なのは、100.101.102.1～100.101.102.254 までとなります（ネットワークアドレス 24bit, ホストアドレス 8bit で分けた場合は同一ネットワークに 254 台までの機器を接続可能）。

100.101.102.0 は、ネットワーク全体を表し、100.101.102.255 はブロードキャストアドレスと言い、100.101.102.0 に属する全ての機器に対する通信の宛先アドレスとなります。

また、ネットワークアドレス部を 2 進数の 1(0b1)、ホストアドレス部を 2 進数の 0(0b0)で表現したアドレスをサブネットマスクと言い、この場合は 255.255.255.0 です。

ネットワークアドレスとホストアドレスの区切りがどこであるかを表現するのに、

・100.101.102.103/24

・100.101.102.103, サブネットマスク 255.255.255.0

どちらかの表現が用いられます。どちらも同じ意味合いです。IP アドレスを使った通信においては、各機器に固有の IP アドレスが割り振られる。ホストアドレスとネットワークアドレスが何ビットで区切られるかといった情報が必要になるという事となります。

ホストアドレスを 24bit とした場合は、同一ネットワーク内に xxx.xxx.xxx.1～xxx.xxx.xxx.254 までの 254 台の機器を接続可能。それ以上の機器を接続したい場合は、ネットワークを分けてネットワーク間はルータ（複数のネットワーク間の通信を取り持つ機器）で接続する必要があります。ホストアドレスを 16bit とした場合は、同一ネットワーク内に 65534 台(2^{16}-2)（ホストアドレスとブロードキャストアドレスを除外）までの機器を接続可能です。

－プライベートアドレスとクラス－

自宅で使用している PC の IP アドレスが、192.168 で始まっているというケースが多いのではないかと思います。

インターネットに直接接続する IP アドレスは、自由に決める事は出来ず、IP アドレスを管理している団体から割り振られてアドレスを設定する必要があります。(*1)

それに対し、インターネットに直接接続していないローカルネットワークの機器やルータを介してインターネットにつながっている機器の場合はルータがグローバルアドレス（インターネットで一意の IP アドレス）とローカルアドレスの変換を行うので、自由に IP アドレスを設定する事が可能です。

ローカルなネットワークで自由に IP アドレスを設定する場合ですが、このアドレスの範囲を利用するのが推奨というアドレスが決められています。ローカルなネットワークの場合は、IP アドレスの制約はありませんが、推奨はあるという事になっています。この、ローカルなネットワークで推奨されているアドレスをプライベートアドレスと言います。

プライベートアドレスは、ホストアドレスが 24bit の場合 192.168.0.0/24～192.168.0.255/24 です。
市販の家庭向けルータでは、192.168.0.0/24 や 192.168.10.0/24, 192.168.100.0/24 などがデフォルトで設定されている事が多いと思います。
ホストアドレスが 16bit の場合のプライベートアドレスは、172.16.0.0/16～172.31.0.0/16。ホストアドレスが 8bit の場合は、10.0.0.0/8 が使われます。

ここで、ホストアドレスが 8bit の場合、クラス A 接続といいます（2^{24} -2→約 1677 万台の機器が接続可能）。16bit の場合、クラス B（2^{16}-2=65534 台の機器が接続可能）。24bit の場合は、クラス C です（2^{8}-2=254 台の機器が接続可能）。（昔は、クラス C の/24 が最小単位であったりもしましたが、最近では/26, /28 の様にクラス C よりも小さいネットワークに分割されるケースもあります。）

(*1)すでに IPv4 の IP アドレスは枯渇しており、新規に割り当てられる事は基本的にはありません。

－IP アドレスの自動割り振り－
IP アドレスは、ネットワーク内で重複してはいけないというルールがありますが、何番をどの機器に割り振ったかを覚えておくのも面倒です。そこで、IP アドレスを自動的に割り振る仕組みが用意されており、DHCP(DinamicallyHostConfigureProtocol)という手順で IP アドレスの割り振りを行うケースが多いです。DHCP では、ルータなどが DHCP サーバ（IP アドレスの割り振り役）となり、PC などが DHCP クライアント（IP アドレスをもらう側）となります。
※DHCP では、IP アドレス以外にも、ネットワーク接続に必要な情報を自動でサーバからクライアントに伝える仕組みです。
企業においては、情報管理部門が IP アドレスの初期設定を行った後で利用者に PC を引き渡すか、DHCP による自動設定のどちらかで、ユーザ自身が IP アドレスの設定を行う事はまずないと思われます。一般家庭においても、ルータが DHCP で IP アドレスの設定を行うケースが圧倒的に多いと思います。

そのため、利用者が使用している PC の IP アドレスがどのようになっているかあまり気にせずに使っている（IP アドレスを意識しなくても、ネットワークに接続して使えてしまっている）ケースが多いとは思いますが、IP ネットワークにおいては、重複なく IP アドレスを割り振るという事が重要です。

本キットでは、基本的には DHCP には非対応ですので、DHCP サーバが存在しない様に構築されたネットワーク（ルータが存在せず、ハブのみで構成されたネットワーク）か、DHCP サーバで自動割り振りに設定されている IP アドレスの範囲外の IP アドレスを設定する様にしてください。

（例えば 192.168.0.1~192.168.0.199 までが DHCP の自動割り振り対象であれば、マイコンボードの IP アドレスを 192.168.0.200 などに設定する。）

（DHCP の割り振り対象アドレスは、ルータのマニュアルに記載されています。）

（本キットで実験する際は、DHCP サーバが存在しない構成が判り易いと思います。）

5.2. ARP(AddressResolutionProtocol)とは？

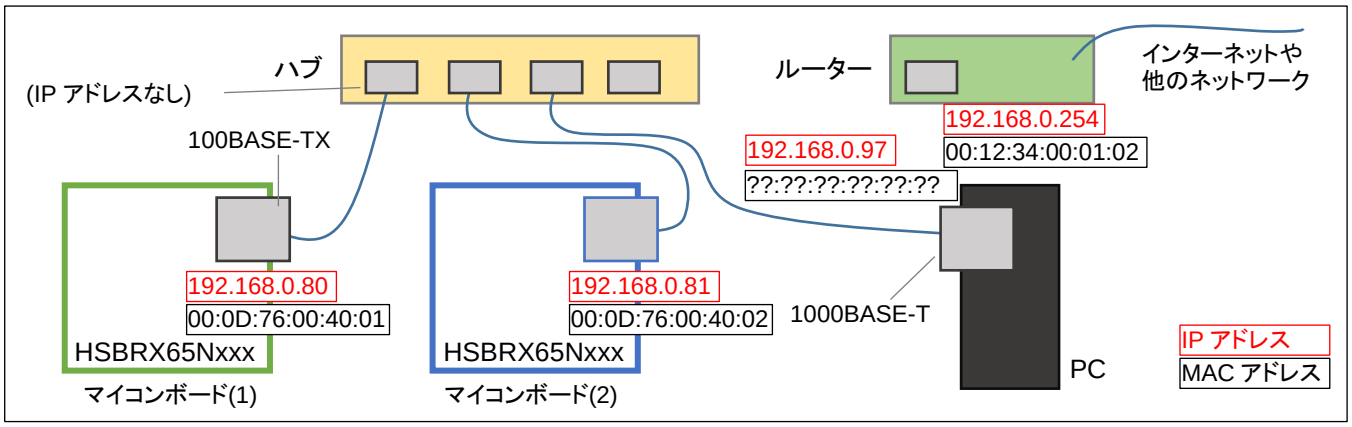
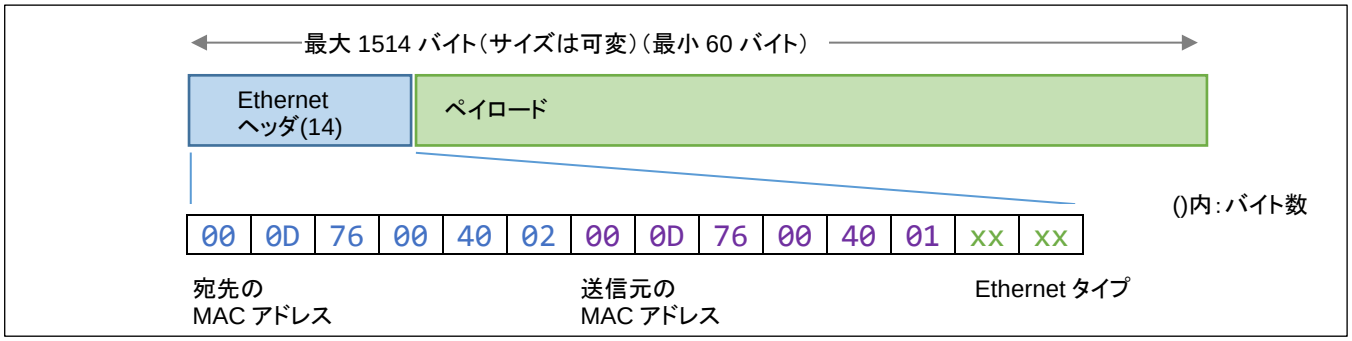


図 5-2 IP ネットワーク接続形態

マイコンボード(1)からマイコンボード(2)に対して、Ethernet でデータを送る際、



Ethernet ヘッダは上記の様になります。マイコンボード(2)は、ネットワーク上を流れているデータで
・宛先 MAC アドレスが自分のアドレス(00-0D-76-00-40-02)
の場合、受信を行います。自分宛ではないデータの場合は、受信を行いません。
（Ethernet ヘッダのレベルで自分宛でない場合は、ペイロード（中身）を見に行かない）

ここで、マイコンボード(1)が IP アドレスが 192.168.0.97 の PC に対してデータを送ろうと思った際に、
192.168.0.97 に対応する MAC アドレスが判らないと送る事ができないという事となります。

192.168.0.97 の IP アドレスを使っている人の MAC アドレスを問い合わせるのが、
ARP(AddressResolutionProtocol)です。アドレス解決を行うためのプロトコル（通信手順）です。

MAC アドレスの問い合わせが、ARP リクエスト。それに対する応答を、ARP リプライと言います。

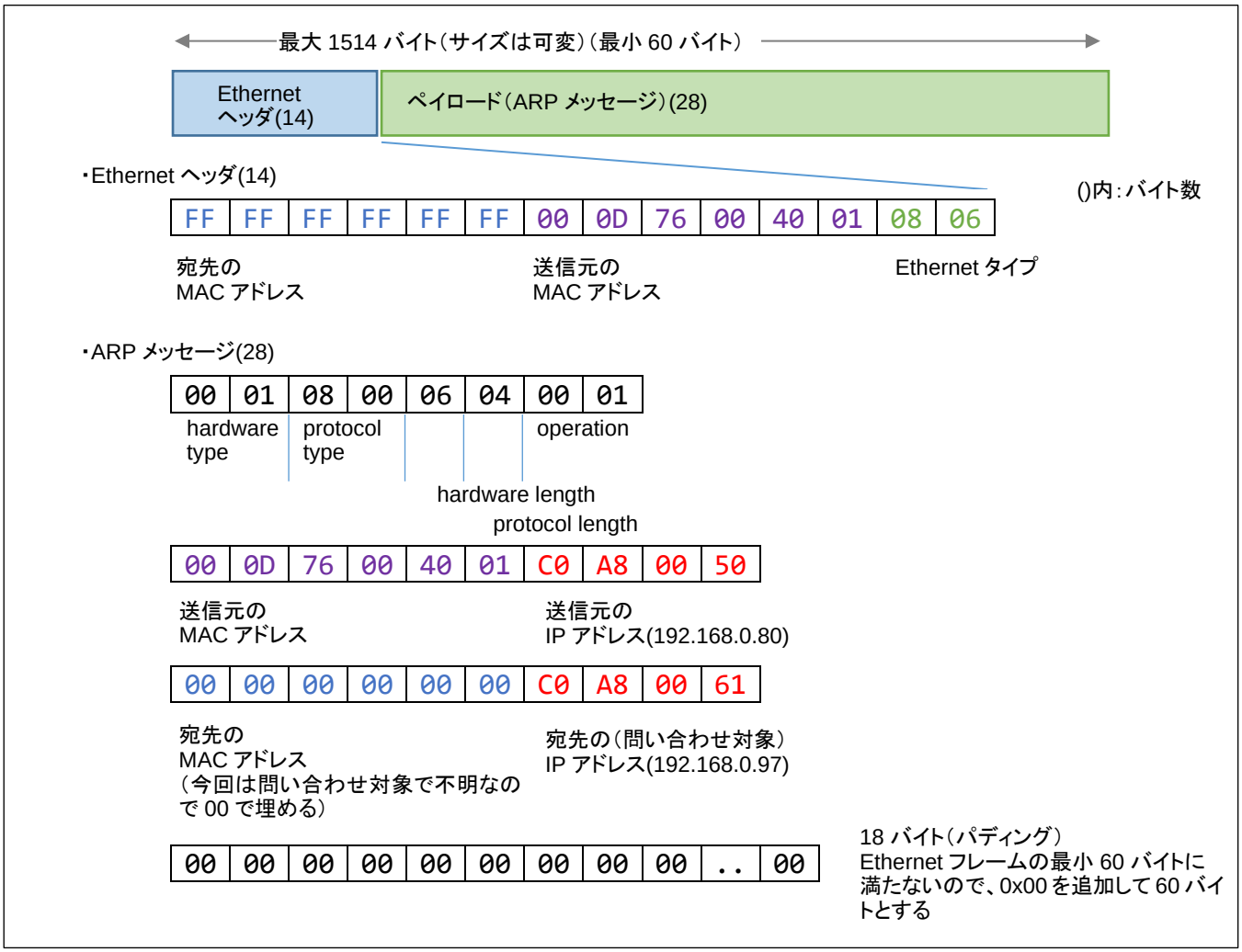


図 5-3 ARP リクエスト

○Ethernet ヘッダ（14 バイト）

フィールド	bit 数	内容	データ例
宛先 MAC アドレス	48	通信先の MAC アドレス (FF-FF-FF-FF-FF-FF の場合はブロードキャスト →全てのネットワークインタフェースに対する通信）	FF FF FF FF FF FF
送信元 MAC アドレス	48	データを送信したインタフェースの MAC アドレス	00 0D 76 00 40 01
Ethernet タイプ	16	データ種別を示すコード	08 06

○ARP メッセージ（28 バイト）

フィールド	bit 数	内容	データ例
Hardware Type	16	ネットワークの物理層の種類 （0x0001 は Ethernet である事を示す）	00 01
Protocol Type	16	上位プロトコルの種類 （0x0800 は IP である事を示す）	08 00
Hardware Length	8	ネットワークの物理層のアドレス長 （MAC アドレス 6 バイト）	06
Protocol Length	8	上位プロトコルのアドレス長 （IPv4 アドレス 4 バイト）	04
Operation	16	ARP の動作 （ARP リクエスト:1, ARP リプライ:2）	00 01
Source MACAddress	48	送信元 MAC アドレス	00 0D 76 00 40 01
Source IP Address	32	送信元 IP アドレス	C0 A8 00 50 (=192.168.0.80)

Destination MAC Address	48	送信先 MAC アドレス （この時点では MAC アドレスが判明していないので、0x00×6）	00 00 00 00 00 00
Destination IP Address	32	送信先 IP アドレス	C0 A8 00 61 (=192.168.0.97)

Ethernet ヘッダ(14)＋ARP メッセージ(28)では、Ethernet フレームの最小 60 バイトに満たないので、隙間を埋めるパディング（0x00×18 バイト）を付与して、ARP リクエストを送信します。

先程、宛先 MAC アドレスが自分宛ではないデータは受信しないと言いましたが、例外があり、宛先 MAC アドレスが FF-FF-FF-FF-FF-FF の場合は、ブロードキャストアドレスと言い全てのインタフェースに向けた通信の場合の特別なアドレスとなります。

よって正しくは、
・宛先 MAC アドレスが自分のアドレス
・宛先 MAC アドレスがブロードキャストアドレス(FF-FF-FF-FF-FF-FF)
の場合、受信が行われます。

ARP リクエストに対応するのが、ARP リプライです。問い合わせがあった IP アドレスに対する、MAC アドレスを返信する通信です。この場合は、応答を返すのは、

・宛先 MAC アドレスがブロードキャストアドレス(FF-FF-FF-FF-FF-FF)
・ARP リクエストのパケット内の宛先 IP アドレスが自分の IP アドレス

の機器、図 5-2 の PC が ARP リプライを送信します。

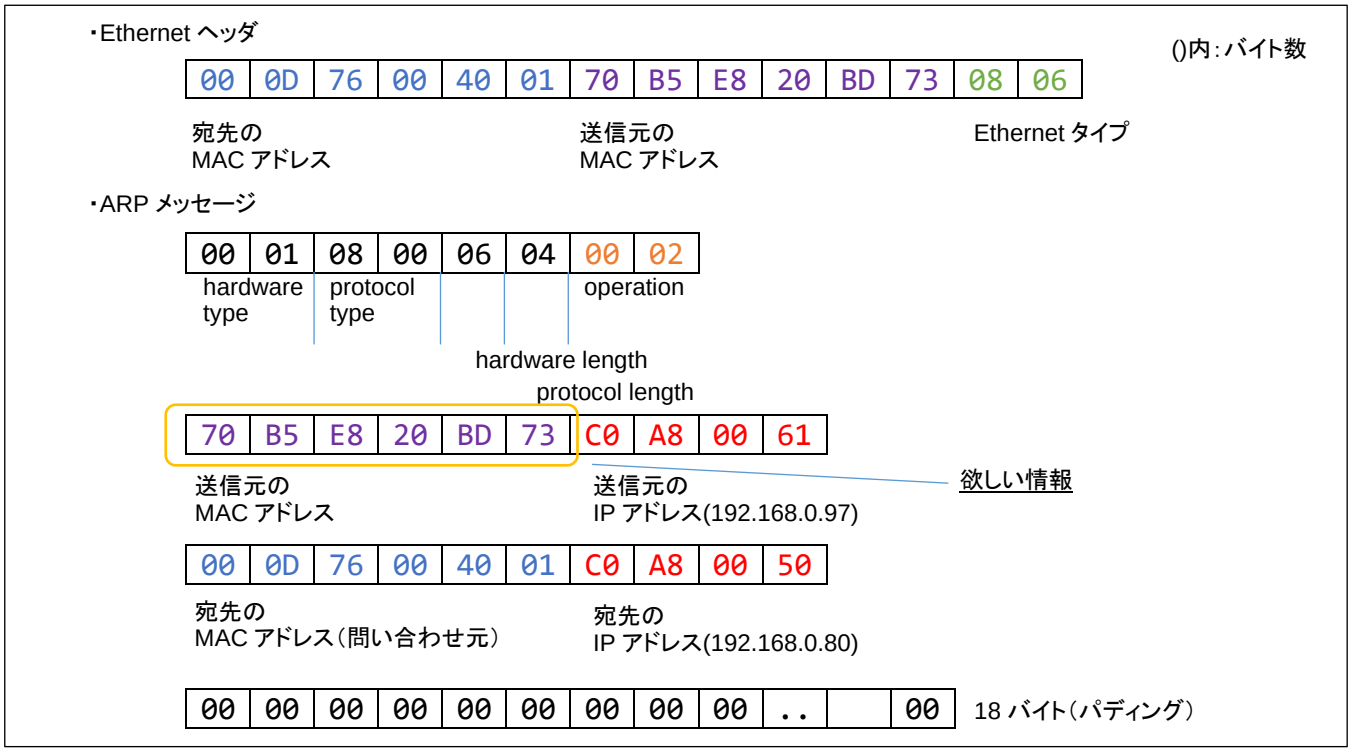


図 5-4 ARP リプライ

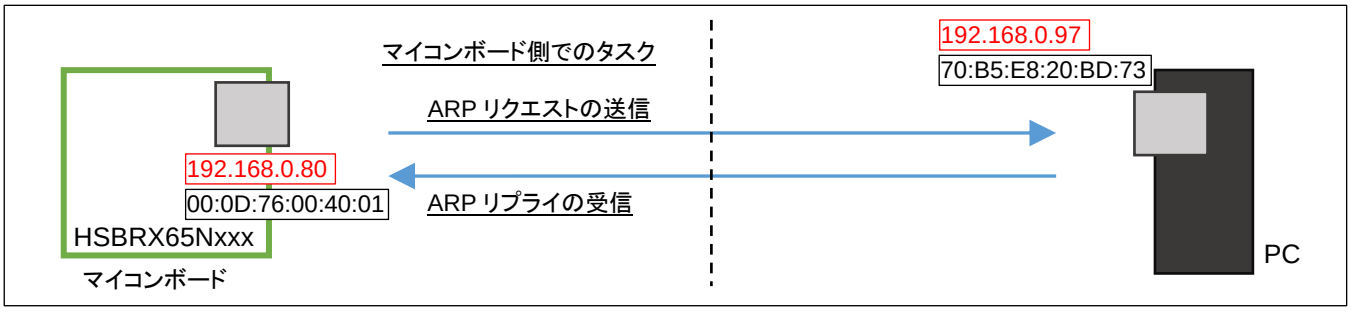
ARP リプライの ARP メッセージ内に PC（IP アドレス=192.168.0.97）に対応する MAC アドレスの情報が含まれますので、この値を記憶しておいて、次に PC（IP アドレス=192.168.0.97）にデータを送る場合は、Ethernet ヘッダ内の宛先 MAC アドレスにこの値を埋め込めば良いという事となります。

ARP は、IP で通信する準備、IP アドレスに対する物理的な通信先である Ethernet のアドレス（MAC アドレス）を調べるプロトコルとなります。

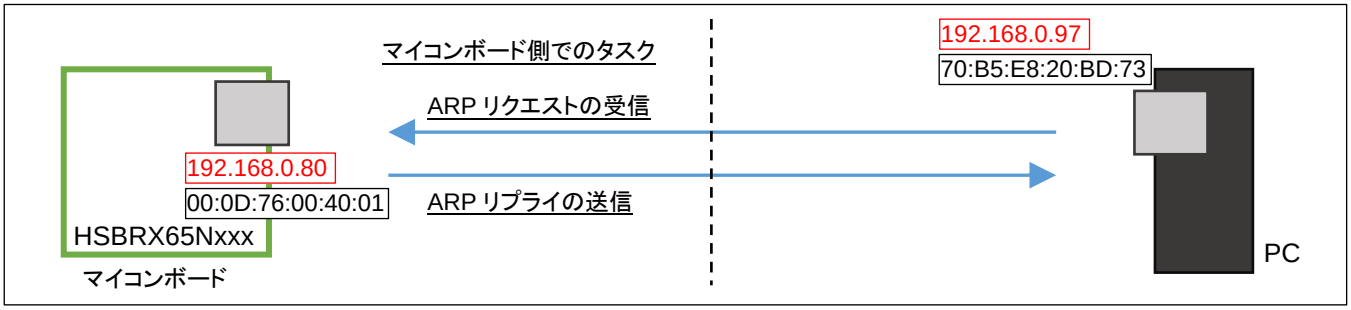
本章の目的は ping を送る事ですが、ping を送る前に ping の送り先の MAC アドレスを調べておく必要があり、その際に使用されるのが ARP リクエストと ARP リプライであるという事です。

なお、上記で説明したのは
・マイコンボードから PC に対しての ARP リクエスト
・PC からマイコンボードに対しての ARP リプライ
ですが、PC からマイコンボードに対して、ARP リクエストがあった場合は、マイコンボードが ARP リプライを返す必要があります。

・PC（や他のネットワーク機器）の MAC アドレス解決



・マイコンボードに対する MAC アドレスの解決



マイコンボード側の処理としては、上記の 4 つのタスクをこなす必要があります。（プログラムの処理に関しては「ソフトウェア編」のマニュアルを参照してください。）

5.3. ping がどうやって実現されているか

ネットワークの世界で ping が通ると、とりあえず安心すると思います。

ping が通るという事は、
・相手側の機器が反応した（生きている）
・自分と相手がつながっている経路に問題がない（LAN ケーブルがちゃんと差さっていて、ハブやルータも仕事をしている）
という確認になるからです。

ping を返す条件は何でしょうか。
ping 192.168.0.80
等の様に実行するので、IP アドレスが設定されている必要はあります。PC からマイコンボードに ping を打って、ping が通るためには、マイコンボード側ではどのような処理が必要でしょうか？

PC から他の機器（マイコンボードではない）に対して ping を実行してみて、その様子をモニタリングしてみます。

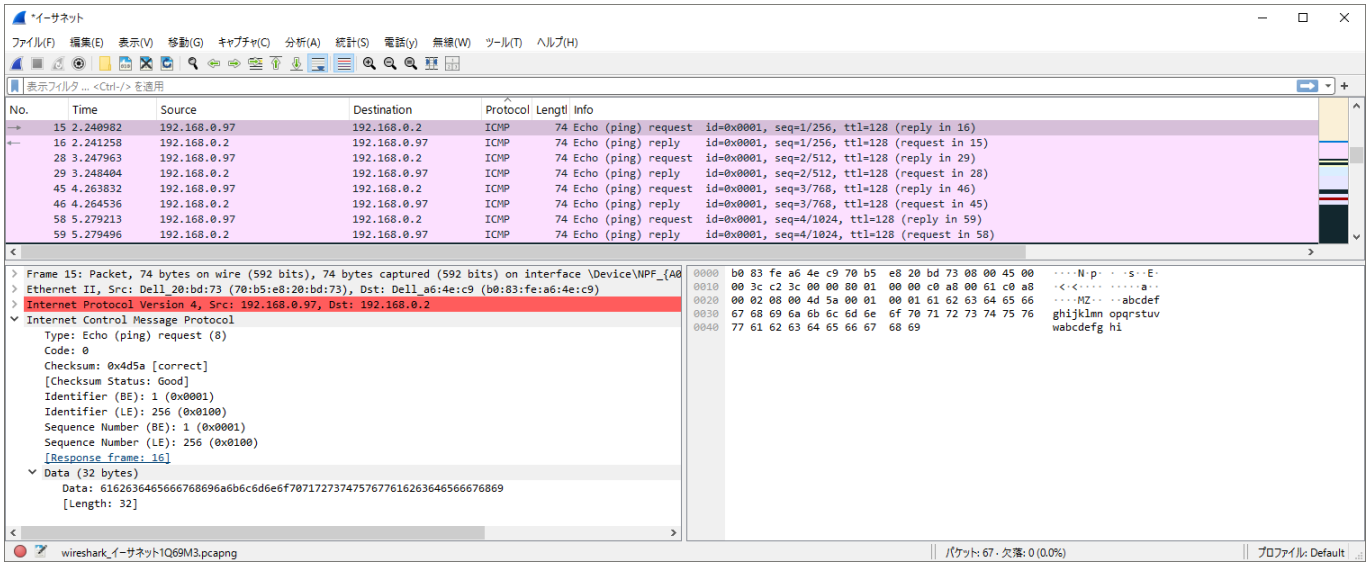
コマンドプロンプトから、ping を実行してみると

```
C:¥Users¥win64-7>ping 192.168.0.2

192.168.0.2 に ping を送信しています 32 バイトのデータ:
192.168.0.2 からの応答: バイト数 =32 時間 <1ms TTL=128
192.168.0.2 からの応答: バイト数 =32 時間 <1ms TTL=128
192.168.0.2 からの応答: バイト数 =32 時間 <1ms TTL=128
192.168.0.2 からの応答: バイト数 =32 時間 <1ms TTL=128

192.168.0.2 の ping 統計:
    パケット数: 送信 = 4、受信 = 4、損失 = 0 (0% の損失)、
ラウンド トリップの概算時間 (ミリ秒):
    最小 = 0ms、最大 = 0ms、平均 = 0ms
```

上記の様になりました。これは、ping が返ってきている場合のメッセージです。



この時の通信を Wireshark で見ると、

Protocol ICMP Echo(ping) request

Protocol ICMP Echo(ping) reply

が 4 回繰り返されている事が見て取れます。データの中身を見てみると以下の様になっています。

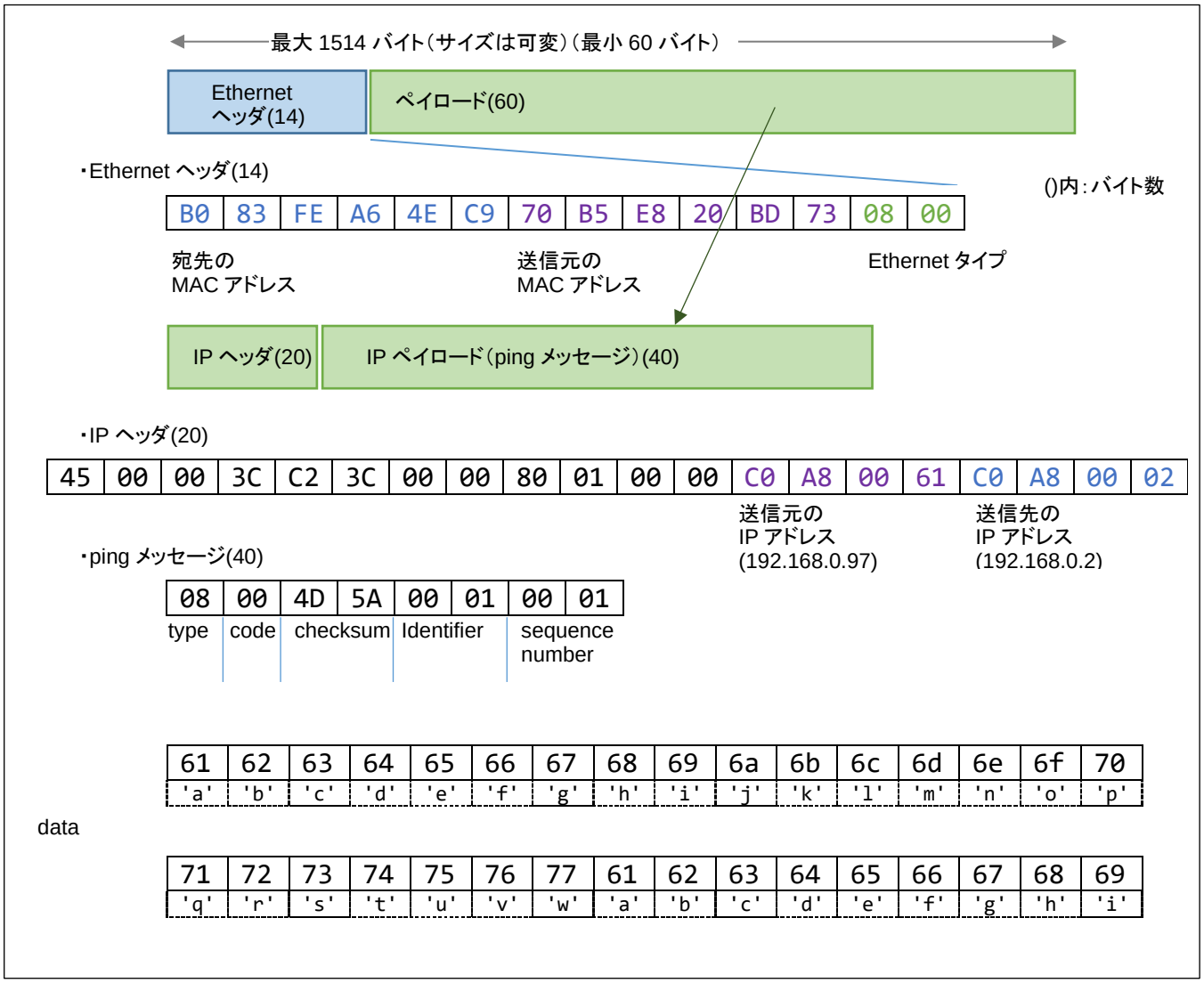


図 5-5 ping リクエスト

ping は、Ethernet ヘッダ内の Ethernet タイプが 0x0800 となっており、IP のデータとなっています。Ethernet フレームのペイロードは、IP ヘッダ＋IP ペイロードで構成されています。IP ペイロードはこの場合は ping メッセージとなっています。

ping は、ICMP(Internet Control Message Protocol)というプロトコルで実現されている通信です。そのため、IP ヘッダ内の Protocol フィールドは 0x01(ICMP)となる通信となります。

ここで IP ヘッダの中身を詳しくみてみます。

〇IP ヘッダ（20 バイト）

フィールド	bit 数	内容	データ例
Version	4	Internet protocol version 4(IPv4)	**4**5
Length	4	IP ヘッダのサイズ（4×5 バイト=20 バイト）	4**5**
Service Type	8	パケットの優先度	00
Total Length	16	IP データグラムの長さ	00 3C （=60 バイト）
Identifier	16	パケットの識別子（固有の ID）	C2 3C （都度変わる）
Flags	3	パケット分割有無 b2 未使用 b1=1 分割禁止, b1=0 分割 OK b0=0 分割された場合途中のパケット, b0=1 分割された最後のパケット	00 00 =0b000 0b0000000000000
Flagment Offset	13	分割されたパケットの位置	
TTL(Time-to-Live)	8	ルータ中継回数上限	80 （=127 回）
Protocol	8	上位プロトコルの種類	01 （=10 進数で 1, ICMP）
Checksum	16	チェックサム	00 00
Source IP Address	32	送信元 IP アドレス	C0 A8 00 61 (=192.168.0.97)
Destination IP Address	32	送信先 IP アドレス	C0 A8 00 02 (=192.168.0.2)

※IP ヘッダは必ず 20 バイトという訳ではなく、オプションが追加されるケースがあります（その場合は、Length のフィールドが 5（=20 バイト）から変わります。IP ヘッダ長は 4 バイトの倍数となります。）

(Version)IP アドレスを、192.168.0.80 の様に 4 バイトで示す IPv4 の場合、IP ヘッダの先頭 4bit は 0x4 となります。本キットでは、IPv4 を取り扱うので、基本的には IP ヘッダの先頭は 0x4 です。

(Length)その次に、IP のヘッダサイズ情報(4bit)が来ます。最短では、0x5 です。この場合は、IP ヘッダの長さは 4×5=20 バイトとなります。IP ヘッダにはオプションが付けられるので、24 バイト以上のヘッダ長となる場合もあります。ヘッダサイズは 4 の倍数に決まっているので、21 バイトなどにはなりません。

(Service Type)はパケットの優先度を制御するデータで、デフォルトは 0x00 です。(Total Length)は、IP ヘッダとペイロードの合計サイズです。(Identifier)は、都度変わる ID 値です。IP のデータは複数の機器を経由して伝送される際に、途中で分割されて届くケースもあり、後でどのデータがひとまとまりか判る様に個別の ID が付加されています。(Flags)(Flagment Offset)は、分割された際に途中のデータなのか最後のデータなのか。何番目のデータなのか等が判る様にするためのものです。

(TTL)は、ルータを経由する度に減算されていき、0 になったらルータを超える事がなくなります。インターネット(*1)は、複数の経路で目的地までデータを届ける設計となっており、場合によっては経路内でループしてしまう事があります。その場合も、無限にループする事が無いように IP パケットには寿命が来るようになっています。

(Protocol)は、プロトコルの種別を示す値で、ping の場合は ICMP(Internet Control Message Protocol)というプロトコル（プロトコル番号 1）となります。

(Checksum)チェックサムは、データ化けが無いかの確認ができる様に設けられたデータフィールドです。IP ヘッダと IP ペイロードのデータから計算される、16bit の計算結果が格納されます。この値が 0x0000 の場合は、チェックサムを省略するといった意味合いとなりますが…。（後述の 5.4 節「IP ヘッダ内のチェックサムフィールドを Wireshark でモニタした場合」を参照してください）

(Source IP Address)送信元の IP アドレス、(Destination IP Address)送信先の IP アドレス、どちらも 4 バイトのデータとなります。

(*1)日本語で書くと「インターネット」ですが、internet→複数のローカルネットワークを接続した広域ネットワーク
Internet→いわゆるショッピングサイトで買い物ができるインターネット、ここでは前者の internet の意味で使用

〇ping メッセージ（40 バイト）

フィールド	bit 数	内容	データ例
Type	8	メッセージの種類 （ping リクエスト:8, ping リプライ:0）	08
Code	8	コード （ping リクエスト:0, ping リプライ:0）	00
Checksum	16	チェックサム	4D 5A
Identifier	16	パケットの識別子（固有の ID）	00 01 （任意）
Sequence Number	16	パケットのシーケンス番号	00 01 （送信毎にインクリメント）
Data	可変	任意のデータ	61 62 ....

(Type)(Code)メッセージの種類とコードは、ping のリクエスト（ping 送信）と、ping リプライ（ping を受信した事を返送）によって決まっています。(Checksum)チェックサムは ping メッセージ全体を所定の計算方法で計算したチェックサム値。

(Identifier)ID 値は、任意ですが Windows で ping を実行した場合 0x0001 が使われる様です。

(Service Number)シーケンス番号は、送信毎に+1 となる値です。

(Data)は、任意ですが Windows で ping を実行すると ASCII では
abcdefghijklmnopqrstuvwabcdefghi
という、32 バイトのデータで固定されている様です。

上記が ping リクエストの場合で、リクエストに対する ping リプライは、以下の様なデータになっています。

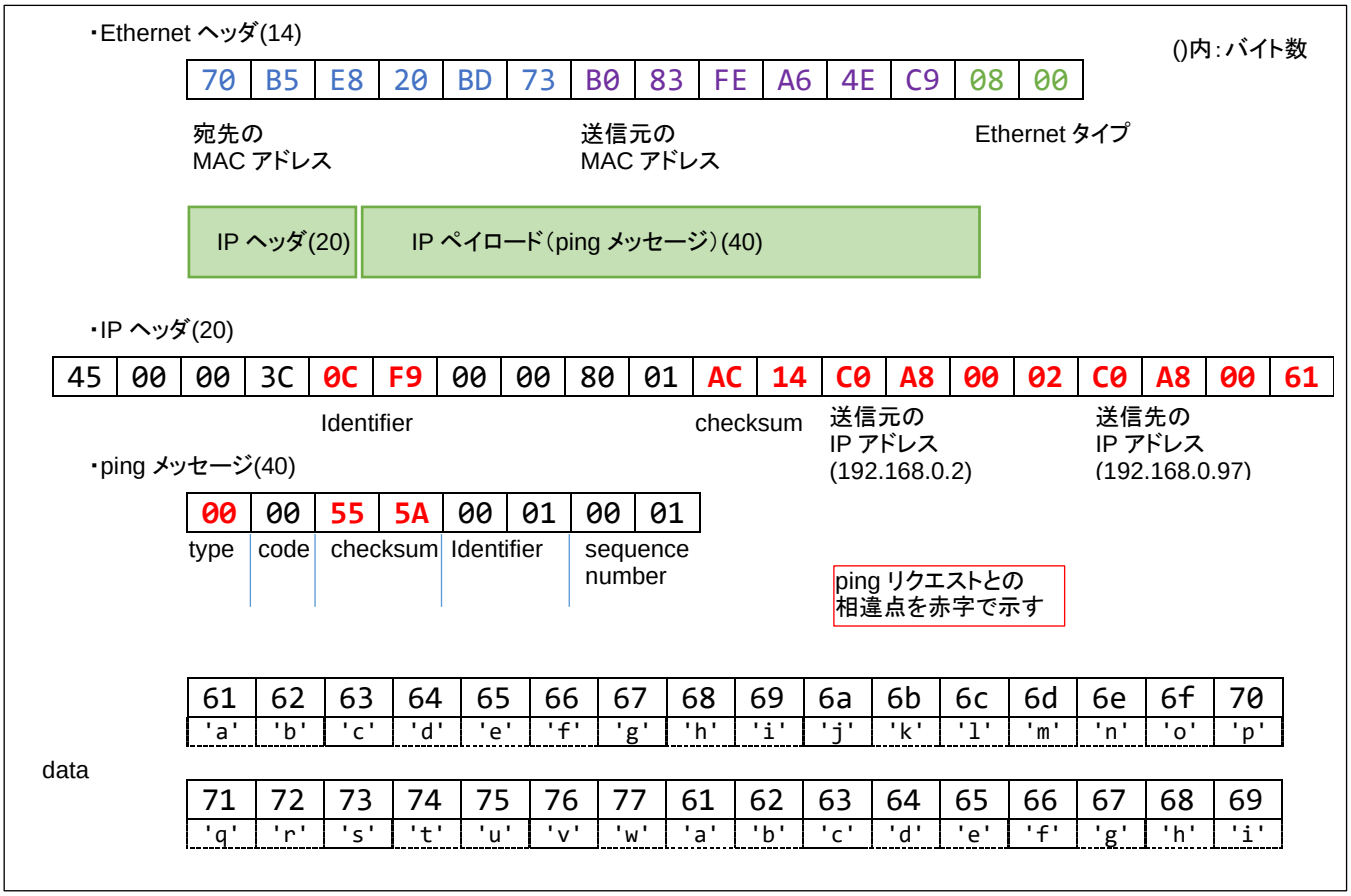


図 5-6 ping リプライ

〇IP ヘッダ（20 バイト）

フィールド	bit 数	内容	データ例
Identifier	16	パケットの識別子（固有の ID）	0C F9 （都度変わる）
Checksum	16	チェックサム	AC 14
Source IP Address	32	送信元 IP アドレス	C0 A8 00 02 (=192.168.0.2)
Destination IP Address	32	送信先 IP アドレス	C0 A8 00 61 (=192.168.0.97)

（ping リクエストと同じケースは記載を省略）

(Idenfifier)ID 値は、都度変わるものなので、ping リクエストと ping リプライには相関関係なし。

(Checksum)は、ping リプライでは計算された値が入っています。

(Source IP Address)(Destination IP Address)送信元と送信先の IP アドレスは、ping リクエストと ping リプライではひっくり返った値が入ります。（これはある意味当たり前の変化です）

〇ping メッセージ（40 バイト）

フィールド	bit 数	内容	データ例
Type	8	メッセージの種類 （ping リクエスト:8, ping リプライ:0）	00
Code	8	コード （ping リクエスト:0, ping リプライ:0）	00
Checksum	16	チェックサム	55 5A
Identifier	16	パケットの識別子（固有の ID） （＝ping リクエストの ID）	00 01

Sequence Number	16	パケットのシーケンス番号 （＝ping リクエストのシーケンス番号）	00 01
Data	256 (32bytes)	データ （＝ping リクエストで受け取ったデータ）	61 62 ...

(Type)メッセージの種類は、ping リプライの場合 0x00 になります。(Checksum)チェックサムは、ping メッセージ全体のチェックサムなので、メッセージの種類のところが変わっているのでチェックサムも変わります。

(Identifier)(Service Number)(Data)ID とシーケンス番号、データに関しては、ping リクエストで受け取ったものを返送するのが流儀です。

Windows でどのように ping を行っているかが分かったので、この動作を RX65N のマイコンに落とし込んでプログラムを作成してみます。

5.4. マイコンボードでの ping 動作

プロジェクト、RX65N_PING（書き込むファイルは、RX65N_PING¥DefaultBuild¥RX65N_PING.mot）をマイコンボードに書き込んで実行してみてください。

```
Copyright (C) 2026 HokutoDenshi. All Rights Reserved.
RX65N Ether ping sample program.

COMMAND:
   i : target IP address set
   j : this board IP address set
   m : this board MAC address set
   p : print IP/MAC address
   S : IP address swap/MAC address increment
---
   s : Operation start
---

USAGE:
   SW3 : ping send
   LED : ping received

-default setting address-
---
target IP address is          -> 192.168.0.81 : i command for change, S command for target<->source swap
this board IP address is    -> 192.168.0.80 : j command for change, S command for source<->target swap
this board MAC address is -> 00-0D-76-00-40-01 : m command for change, S command for increment MAC address
---
First m, i, j, p, S command -> initial setting

Please input 's' for operation start!
```

起動すると、上記の様な表示となります。

項目	規定値	備考
ping 送信先の IP アドレス	192.168.0.81	i コマンドで変更
マイコンボードの IP アドレス	192.168.0.80	j 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	m コマンドで変更

なお、2 枚のマイコンボードをネットワークにつなぐ場合は 2 台目の DIP-SW(3)を ON 側に設定してください。その場合は、起動後のデフォルト設定が

項目	規定値	備考
ping 送信先の IP アドレス	192.168.0.80	i コマンドで変更
マイコンボードの IP アドレス	192.168.0.81	j コマンドで変更
マイコンボードの MAC アドレス	00-0D-76-00-40-02	m コマンドで変更

となります。（基本的には何も変更せずに、2 台のボード間で ping が通る設定です。）
（2 台目のマイコンボード→DIP-SW(3)=ON 設定というのは、今後も同様です。）

ここでは、PC と ping のやり取りを行ってみます。PC の IP アドレスは、192.168.0.97 とします。

・i コマンドを入力

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

IP[0](b31-24) >192
```

IP アドレス入力となりますので、3 桁の数値。3 桁に満たない場合は、数値＋Enter を入力してください。

```
>command=i
target IP address 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
```

上記で ping 送信先の IP アドレスの設定は OK です。マイコンボード側の IP アドレスの変更が必要な場合は、j コマンドで変更してください。m コマンドで、マイコンボードの MAC アドレスの変更ができますが、基本的には不要だと思います。

※MagicPacket の時は MAC アドレスに対してデータを送る形でしたが、ping は IP アドレスに対してのやり取りとなりますので、MAC アドレスは裏方（Ethernet 通信においては必要だが、表には出ない）設定となります

上記以外に S コマンドがあり、これは DIP-SW(3)=ON で起動したときの設定、マイコンボードの 1 台目と 2 台目の設定を入れ替えるコマンドとなります。

その他、初期設定の状態で入力可能なコマンドですが、p コマンドで初期設定の値を確認できます。

```
>command=p
target IP address is          -> 192.168.0.97
this board IP address is    -> 192.168.0.80
this board MAC address is -> 00-0D-76-00-40-01
```

初期設定が終わると、s コマンドを入力してください。動作開始のコマンドです。

```
>command=s

Operation start !
PUSH SW3 -> send ping to target
Ether0: LINK-UP
```

ここで、Ether0: LINK-UP 表示が出ない場合は、LAN ケーブルの接続やハブの電源などを確認してください。

この状態で、ping の送信と、外部から ping を打った際にマイコンボードが ping の応答（ping リプライ）を返すようになります。

5.4.1. ping 応答（ping リプライ）

マイコンボードの IP アドレスは、192.168.0.80（初期値のまま）としているので、PC からコマンドプロンプトを開き、マイコンボードに対して ping を実行してみます。

・PC のコマンドプロンプト

```
C:¥Users¥win64-7>ping 192.168.0.80

192.168.0.80 に ping を送信しています 32 バイトのデータ:
192.168.0.80 からの応答: バイト数 =32 時間 <1ms TTL=128
192.168.0.80 からの応答: バイト数 =32 時間 <1ms TTL=128
192.168.0.80 からの応答: バイト数 =32 時間 <1ms TTL=128
192.168.0.80 からの応答: バイト数 =32 時間 <1ms TTL=128

192.168.0.80 の ping 統計:
    パケット数: 送信 = 4、受信 = 4、損失 = 0 (0% の損失)、
ラウンド トリップの概算時間 (ミリ秒):
    最小 = 0ms、最大 = 0ms、平均 = 0ms
```

上記の様な表示となれば、マイコンボード側は PC からの ping に応答しています。

この時、マイコンボード側では、端末に以下の表示が出ます。

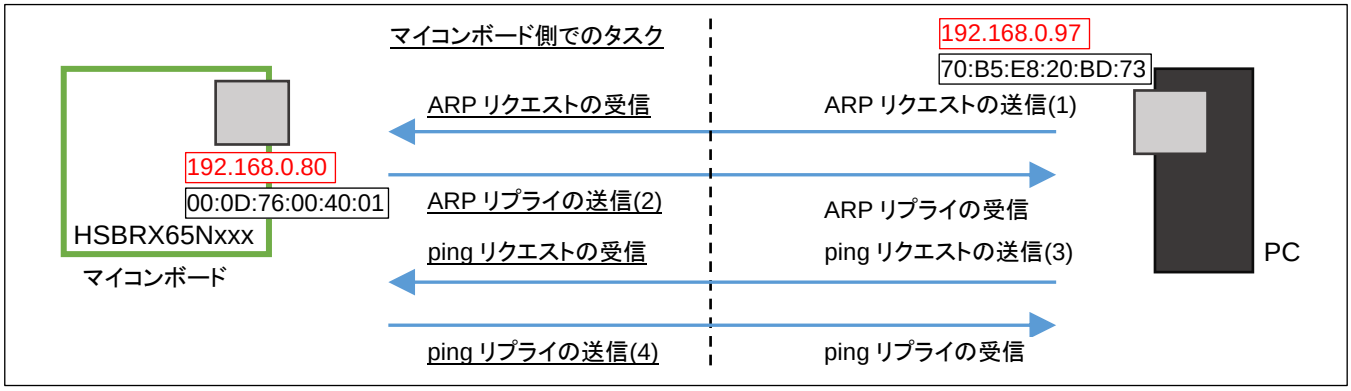
```
ARP reply send (this board MAC address = 00-0D-76-00-40-01) to 70-B5-E8-20-BD-73
ping reply send to 192.168.0.97
ping reply send to 192.168.0.97
ping reply send to 192.168.0.97
ping reply send to 192.168.0.97
```

最初の行は、PC から ARP リクエストを受け取ったので、ARP リプライを返したという表示です。

PC が 192.168.0.80 に ping リクエストを送る際、この段階では PC は 192.168.0.80 の IP アドレスを使っているネットワークインタフェースの MAC アドレスを知りません。どの MAC アドレスに対して、ping リスエストを送ってよいか判らないので、まずは ARP リクエストを送ります。その ARP リクエストに対し、マイコンボードは自分の MAC アドレス(00-0D-76-00-40-01)を返しています。

その後、PC は ping リクエストの送信先が判ったので、MAC アドレス=00-0D-76-00-40-01, IP アドレス =192.168.0.80 の機器に対して ping リクエストを送っています。WindowsPC の ping はデフォルトでは 4 回送る様になっているので、PC 側でも 4 回の応答の表示となっており、マイコンボード側でも ping reply send が 4 回表示されています。

・マイコンボードでの ping 応答



ー通信パケット例ー

(1)PC からの ARP リクエスト

・Ethernet ヘッダ(14)

FF	FF	FF	FF	FF	FF	70	B5	E8	20	BD	73	08	06
宛先 MAC アドレス（ブロードキャスト）						送信元 MAC アドレス						Ethernet type (ARP)	

・ARP メッセージ(28)

00	01	08	00	06	04	00	01	70	B5	E8	20	BD	73	C0	A8
Hardware type (Ethernet)		Protocol type (IPv4)		length (*1)	length (*2)	ARP reqest		送信元 MAC アドレス						送信元 IP アドレス	

00	61	00	00	00	00	00	00	C0	A8	00	50
送信元 IP アドレス (192.168.0.97)		送信先 MAC アドレス （この時点では不明）						送信先 IP アドレス (192.168.0.80)			

(2)APR リプライ

・Ethernet ヘッダ(14)

70	B5	E8	20	BD	73	00	0D	76	00	40	01	08	06
宛先 MAC アドレス						送信元 MAC アドレス						Ethernet type (ARP)	

・ARP メッセージ(28)

00	01	08	00	06	04	00	02	00	0D	76	40	00	01	C0	A8
Hardware type (Ethernet)		Protocol type (IPv4)		length (*1)	length (*2)	ARP reply		送信元 MAC アドレス						送信元 IP アドレス	

00	50	70	B5	E8	20	BD	73	C0	A8	00	61
送信元 IP アドレス (192.168.0.80)		送信先 MAC アドレス						送信先 IP アドレス (192.168.0.97)			

(*1)Hardware length, MAC アドレスのサイズ 6 バイト

(*2)Protocol length, IP アドレスのサイズ 4 バイト

(3)PC からの ping リクエスト

・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	3C	2A	6F	00	00	80	01	00	00	C0	A8	00	61
IPv4 ヘッダ 20B	Service type	Total Length 60 バイト		ID		Fragment		TTL	Protocol ICMP	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)			

・ping メッセージ(40)

08	00	4D	42	00	01	00	19	61	62	63	64	65	66	67	...
ping request	Code	Checksum		ID		Sequence Number		'a'	'b'	'c'	'd'	'e'	'f'	'g'	

(4)ping リプライ

・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	3C	AB	CD	00	00	80	01	0C	F2	C0	A8	00	50
IPv4 ヘッダ 20B	Service type	Total Length 60 バイト		ID (*1)		Fragment		TTL	Protocol ICMP	Checksum (*2)		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)			

・ping メッセージ(40)

00	00	55	42	00	01	00	19	61	62	63	64	65	66	67	...
ping reply	Code	Checksum (*3)		ID (*4)		Sequence Number (*4)		'a'	'b'	'c'	'd'	'e'	'f'	'g'	

※ping メッセージのデータ部分は ASCII で、abcdefg....（32 バイト）ですが記載は後略（5.3 節で全データを記載しています。

ここで、マイコンボード側での処理は(2)(4)になります。

(*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。

(*2)IP ヘッダ内の Checksum はマイコンボード側で計算する必要があります（後述）。

(*3)ping メッセージ内の Checksum はマイコンボード側で計算する必要があります（後述）。

(*4)ping メッセージ内の ID と Sequence Number は、PC が送ってきた ID をそのまま返信しています（仕様）。

－IP ヘッダ内のチェックサムの計算方法－

・IP ヘッダ(20)

45	00	00	3C	AB	CD	00	00	80	01	0C	F2	C0	A8	00	50
IPv4 ヘッダ 20B	Service type	Total Length 60 バイト		ID		Fragment		TTL	Protocol ICMP	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)			

・チェックサムは 2 バイト単位での計算なので 2 バイトずつ区切る

・チェックサムのフィールドは 0x0000 にする

4500	003C	ABCD	0000	8001	0000	C0A8	0050	C0A8	0061

・数値を単純に加算していく

4500	+	003C	=	453C

453C	+	ABCD	=	F109

F109	+	0000	=	F109

F109	+	8001	=	1710A

・0xFFFF を超えた場合は、0xFFFF を超えた部分の最上位桁は消して、1 を加算する(0x1710A-0x10000+0x1)

710B	+	0000	=	710B

710B	+	C0A8	=	131B3

31B4	+	0050	=	3204

3204	+	C0A8	=	F2AC

F2AC	+	0061	=	F30D

・最後にビットごとの反転を取る

~F2FC	=	0CF2

ここで計算した、0x0CF2 が IP ヘッダのチェックサム値となります。

加算する際は、1 の補数和というアルゴリズム（0xFFFF を超えた場合は、最上位桁を消して 1 を加算）で計算します。最後に、ビットごとの反転(Not)を取った値がチェックサム値です。

マイコンボード側では、この様に計算したチェックサム値を IP ヘッダ内の Checksum フィールドに埋め込んで送信します。

[参考]受信側でのチェックサム検証方法

4500	003C	ABCD	0000	8001	0CF2 (Checksum)	C0A8	0050	C0A8	0061

受信側では、チェックサムが埋め込まれているデータを受信します。受信側では送信側と同じ 1 の補数和のアルゴリズムで全データを加算していきます。

4500	+	003C	=	453C	
453C	+	ABCD	=	F109	
F109	+	0000	=	F109	
F109	+	8001	=	1710A	→710B
710B	+	0CF2	=	7DFD	
7DFD	+	C0A8	=	13EA5	→3EA6
3EA6	+	0050	=	3EF6	
3EF6	+	C0A8	=	FF9E	
FF9E	+	0061	=	FFFF	
~FFFF			=	0000	

計算結果が 0 になれば、チェックサムは一致です。
（受信側でチェックサムが一致しなければ、データは伝送途中でデータ化けが生じたと判断されて破棄されます。）

－ping メッセージ内のチェックサムの計算方法－

計算のアルゴリズムは、IP ヘッダと同じです。計算対象は、ping メッセージの 40 バイトの範囲です。

・ping メッセージ(40)

00	00	55	42	00	01	00	19	61	62	63	64	65	66	67	68
ping reply	Code	Checksum (*3)		ID (*4)		Sequence Number (*4)		'a'	'b'	'c'	'd'	'e'	'f'	'g'	'h'

69	6A	6B	6C	6D	6E	6F	70	71	72	73	74	75	76	77	61
'i'	'j'	'k'	'l'	'm'	'n'	'o'	'p'	'q'	'r'	's'	't'	'u'	'v'	'w'	'a'

62	63	64	65	66	67	68	69
'b'	'c'	'd'	'e'	'f'	'g'	'h'	'i'

0000	+	0000	=	0000	Checksum フィールドは 0x0000 で計算する
0000	+	0001	=	0001	
0001	+	0019	=	001A	
001A	+	6162	=	617C	
617C	+	6364	=	C4E0	
C4E0	+	6566	=	12A46	→2A47
2A47	+	6768	=	91AF	
91AF	+	696A	=	FB19	
FB19	+	6B6C	=	16685	→6686
6686	+	6D6E	=	D3F4	
D3F4	+	6F70	=	14364	→4365
4365	+	7172	=	B4D7	
B4D7	+	7374	=	1284B	→284C
284C	+	7576	=	9DC2	
9DC2	+	7761	=	11523	→1524
1524	+	6263	=	7787	
7787	+	6465	=	DBEC	
DBEC	+	6667	=	14253	→4254
4254	+	6869	=	AABD	
~AABD			=	5542	

上記の様な計算方法で算出できます。

－PC 側からの ARP リクエストに関して－

PC からマイコンボードに対して ping を実行すると、初回は必ず ARP リクエストの後に、ping リクエストが送信されます。これは、PC がマイコンボード(192.168.0.80)の MAC アドレスを知らないからです。

それに対し、2 回目以降は、ARP リクエストなしに、ping リクエストが送信される（PC は、192.168.0.80 の MAC アドレスが 00-0D-76-00-40-01 である事を既に知っているので）動作となります。

この時、PC のコマンドプロンプトで、arp -a とコマンドを入力すると、

・PC のコマンドプロンプト(arp -a)

```
C:¥Users¥win64-7>arp -a

インターフェイス: 192.168.0.97 --- 0x10
  インターネット アドレス  物理アドレス           種類
  192.168.0.2           b0-83-fe-a6-4e-c9     動的
  192.168.0.80          00-0d-76-00-40-01     動的
  192.168.0.255         ff-ff-ff-ff-ff-ff     静的
  224.0.0.22            01-00-5e-00-00-16     静的
```

上記の様な表示となります。

この場合は、PC は 192.168.0.80 の IP アドレスを使用しているネットワークインタフェースの MAC アドレスが、00-0D-76-00-40-01 である事を記憶しているので、ARP リクエストなしに ping リクエストを送信します。

なお、管理者コマンドプロンプトで arp -d と入力すると、IP アドレスと MAC アドレスの対応表は破棄されるので、ping 実行時、初回実行時と同様に ARP リクエストの後 ping リクエストという動作となります。

・PC から 2 回連続で ping を実行した場合のマイコンボード側の端末表示例

```
ARP reply send (this board MAC address = 00-0D-76-00-40-01) to 70-B5-E8-20-BD-73
ping reply send to 192.168.0.97
ping reply send to 192.168.0.97
ping reply send to 192.168.0.97
ping reply send to 192.168.0.97
ping reply send to 192.168.0.97
ping reply send to 192.168.0.97
ping reply send to 192.168.0.97
ping reply send to 192.168.0.97
ARP reply send (this board MAC address = 00-0D-76-00-40-01) to 70-B5-E8-20-BD-73
```

1 回目
（ARP リクエストの後に ping リクエスト 4 回）

2 回目
（ARP リクエストなしに ping リクエスト 4 回の後で
何故か ARP リクエスト）

PC 側では、arp -a で 192.168.0.80 に対応する MAC アドレスが「動的」となっており、常に 192.168.0.80 と 00-0D-76-00-40-01 が対応する訳ではなく、変化する事もあると認識されています。

基本は、MAC アドレスの情報取得から 30 秒（程度）以上経過した場合は、再度 ARP のリクエストを発行する様です。

5.4.2. ping の送信（ping リクエスト）

今度は、マイコンボードから PC に対して ping を実行してみます。

（マイコンボードの起動時に i コマンドで、192.168.0.97 に対して ping を打つように設定済みとします。）

SW3 を押すと、マイコンボード側では、端末に以下の表示が出ます。

```
ARP reply send (this board MAC address = 00-0D-76-00-40-01) to 70-B5-E8-20-BD-73
ping 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
```

もう一度 SW3 を押すと、

```
ping request send to 192.168.0.97
ping reply received from 192.168.0.97
```

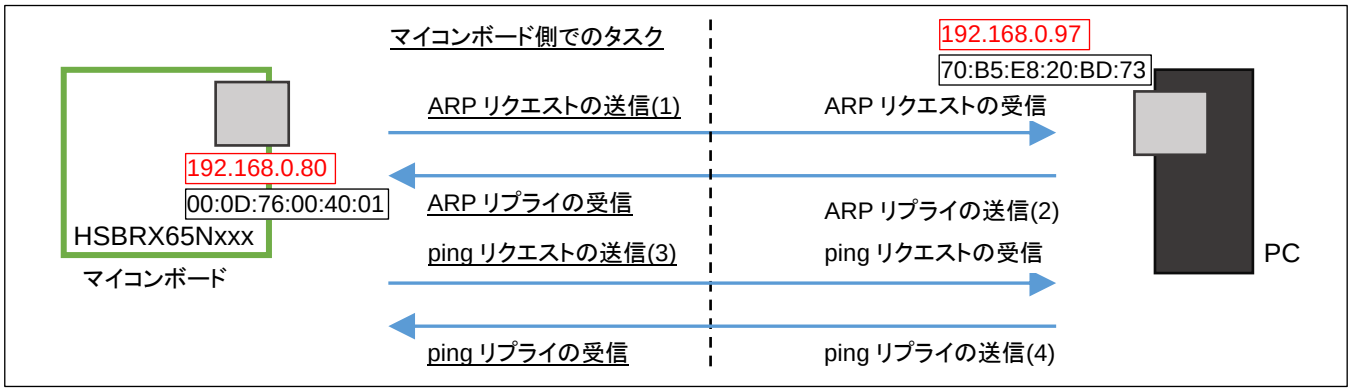
となります。

1 回目に SW3 を押した際には、192.168.0.97 に対応する MAC アドレスが不明なので、ARP リクエストを送って MAC アドレス(70-B5-E8-20-BD-73)の情報を取得します。（本プログラムでは、ここで一連の動作を完了します）

2 回目に SW3 を押した際は、192.168.0.97 に対して、ping リクエストを送ります。ping リプライを受信した場合は、「ping reply received from」が表示されます。PC の ping コマンドとは異なり、1 回しか ping リクエストを送りません。

SW3 を押す動作、3 回目以降、MAC アドレスの解決が済んでいれば、都度 ping リクエストを送ります。

・マイコンボードでの ping 送信



－通信パケット例－

(1)マイコンボードからの ARP リクエスト

・Ethernet ヘッダ(14)

FF	FF	FF	FF	FF	FF	00	0D	76	00	40	01	08	06
宛先 MAC アドレス（ブロードキャスト）						送信元 MAC アドレス						Ethernet type (ARP)	

・ARP メッセージ(46)

00	01	08	00	06	04	00	01	00	0D	76	00	40	01	C0	A8
Hardware type (Ethernet)		Protocol type (IPv4)		length (*1)	length (*2)	ARP reqest		送信元 MAC アドレス						送信元 IP アドレス	

00	50	00	00	00	00	00	00	C0	A8	00	61	00	...	00
送信元 IP アドレス (192.168.0.80)		送信先 MAC アドレス （この時点では不明）						送信先 IP アドレス (192.168.0.97)				パディング(*3) （18 バイト）		

(2)APR リプライ

・Ethernet ヘッダ(14)

00	0D	76	00	40	01	70	B5	E8	20	BD	73	08	06
宛先 MAC アドレス						送信元 MAC アドレス						Ethernet type (ARP)	

・ARP メッセージ(20)

00	01	08	00	06	04	00	02	70	B5	E8	20	BD	73	C0	A8
Hardware type (Ethernet)		Protocol type (IPv4)		length (*1)	length (*2)	ARP reply		送信元 MAC アドレス						送信元 IP アドレス	

00	61	00	0D	76	00	40	01	C0	A8	00	50
送信元 IP アドレス (192.168.0.97)		送信先 MAC アドレス						送信先 IP アドレス (192.168.0.80)			

(*1)Hardware length, MAC アドレスのサイズ 6 バイト

(*2)Protocol length, IP アドレスのサイズ 4 バイト

(*3)Ethernet フレームの最小長 60 バイトとなる様に 0x00 を 18 バイト送信データの末尾に付加

PC が送信する ARP リプライや ARP リクエストのデータを Wireshark でダンプすると 42 バイトになっています。これは、後述の「－IP ヘッダ内のチェックサムフィールドを Wireshark でモニタした場合－」記載の内容とも関連しますが、ソフトウェアのレベルでは 42 バイトのデータをネットワークインタフェースに引き渡す。ネットワークインタフェースでは、Ethernet の規格を満たすように、60 バイトに満たない場合は末尾にパディングデータを付加して送信する動作となるはずです。（ネットワークインタフェース（ハード）が、ある程度都合良く処理してくれる）

それに対し、マイコンボード側で送信するデータに関しては、ソフトウェアで生成したデータに関して「誰か」が Ethernet の規格を考えて適切にパディングデータを追加してくれる様な事はありません。パディングデータの追加も含めて、プログラム作成者の範疇での処理となります。

(3)マイコンボードからの ping リクエスト

・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	3C	AB	CD	00	00	80	01	0C	F2	C0	A8	00	50
IPv4 ヘッダ 20B	Service type	Total Length 60 バイト		ID (*1)		Fragment		TTL	Protocol ICMP	Checksum (*2)		送信元 IP アドレス (192.168.0.80)			

C0	A8	00	61
送信先 IP アドレス (192.168.0.80)			

・ping メッセージ(40)

08	00	3B	27	12	34	00	01	61	62	63	64	65	66	67	...
ping request	Code	Checksum (*2)		ID (*3)		Sequence Number (*4)		'a'	'b'	'c'	'd'	'e'	'f'	'g'	

(4)ping リプライ

・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	3C	8A	1B	00	00	80	01	00	00	C0	A8	00	61
IPv4 ヘッダ 20B	Service type	Total Length 60 バイト		ID		Fragment		TTL	Protocol ICMP	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)			

・ping メッセージ(40)

00	00	43	27	12	34	00	01	61	62	63	64	65	66	67	...
ping reply	Code	Checksum		ID (*5)		Sequence Number (*5)		'a'	'b'	'c'	'd'	'e'	'f'	'g'	

※ping メッセージのデータ部分は ASCII で、abcdefg....（32 バイト）ですが記載は後略（5.3 節で全データを記載しています。

ここで、マイコンボード側での処理は(1)(3)になります。

(*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。

(*2)IP ヘッダ内, ping メッセージ内の Checksum はマイコンボード側で計算する必要があります（5.4.1 参照）

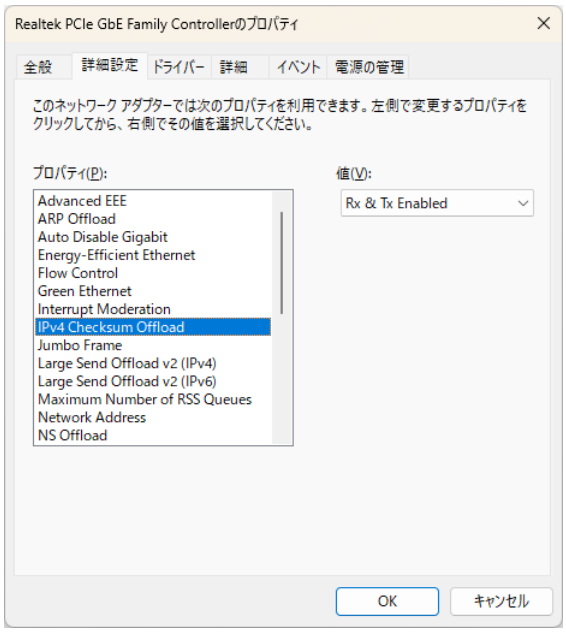
(*3)ping メッセージ内の ID は、、本プログラムでは 0x1234 に固定しています。

(*4)ping メッセージ内の SequenceNumber は起動時 1 から ping を送るごとにインクリメントした数値としています。

(*5)参考ですが、PC から送り返される ping メッセージ内の ID と Sequence Number は、マイコンボードが送ったものをそのまま返信してくれています（仕様）。

ARP と ping のプログラムの中身に関しては、ソフトウェア編マニュアルに書かれていますが、ネットワークの動作では返ってくるのが当たり前の ping ですが、ping を送信、ping に応答するだけでも、中味を詳しく見てみるとそれなりに色々な事をやらなければならない事が見えてきます。

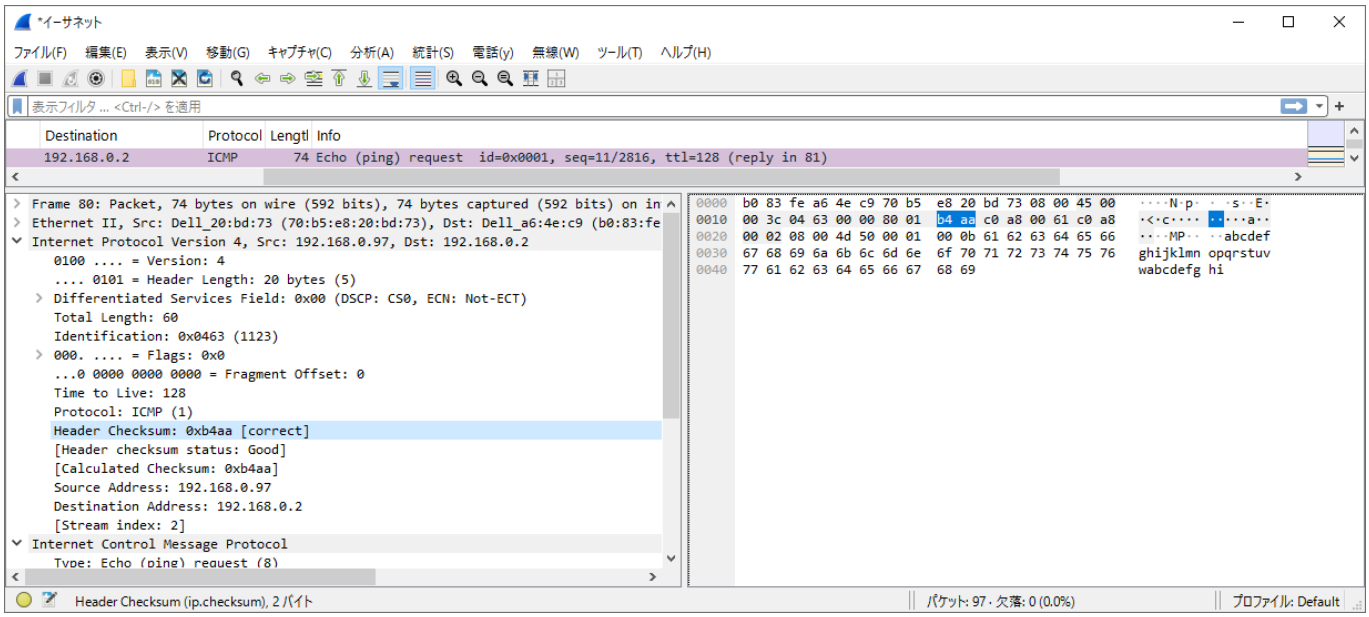
5.5. IP ヘッダ内のチェックサムフィールドを Wireshark でモニタした場合



ネットワークインタフェースのアダプタ設定の項目に、
IPv4 Checksum Offload
という項目があります。この設定は、デフォルトでは「RX & TX Enabled」（送信時も受信時も有効）になっているのではないかと思います。

この部分が有効（デフォルト）の場合は、ハードウェア（ネットワークインタフェース）レベルでチェックサムの処理が行われるため、パケットダンプ時は 0x0000 になっていても、実際の送信データにはネットワークインタフェースカード（現在ではマザーボード、オンボードの LAN インタフェースが主流です）がハードウェアでチェックサムを付与して送信するという動作になると思います。

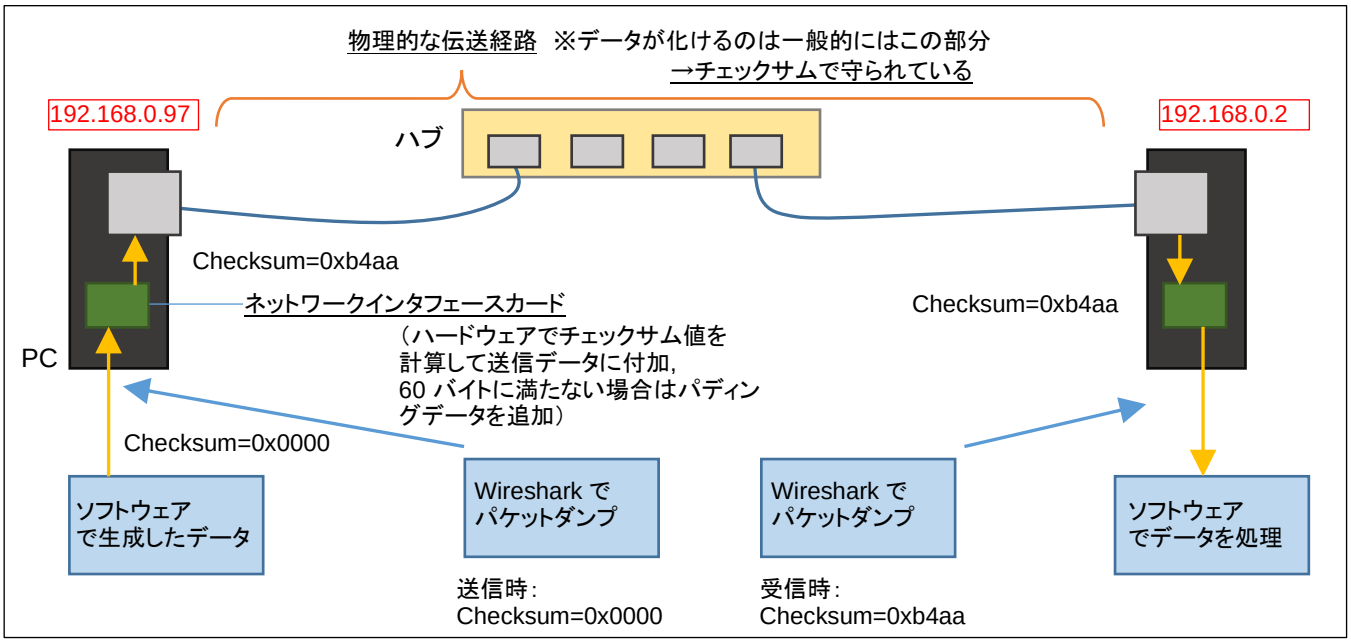
上記設定を Disable として送信した場合、



IP ヘッダ内のチェックサムは、0x0000 ではなく、計算値が入ります。

この場合は、OS（Windows の IP プロトコルスタック）のレベルでソフトウェアでチェックサム値を計算して送信している動作だと思います。

ネットワークインタフェースレベルで、IPv4 のチェックサムを有効にした場合（デフォルト）は、以下の様な動作になると考えられます。



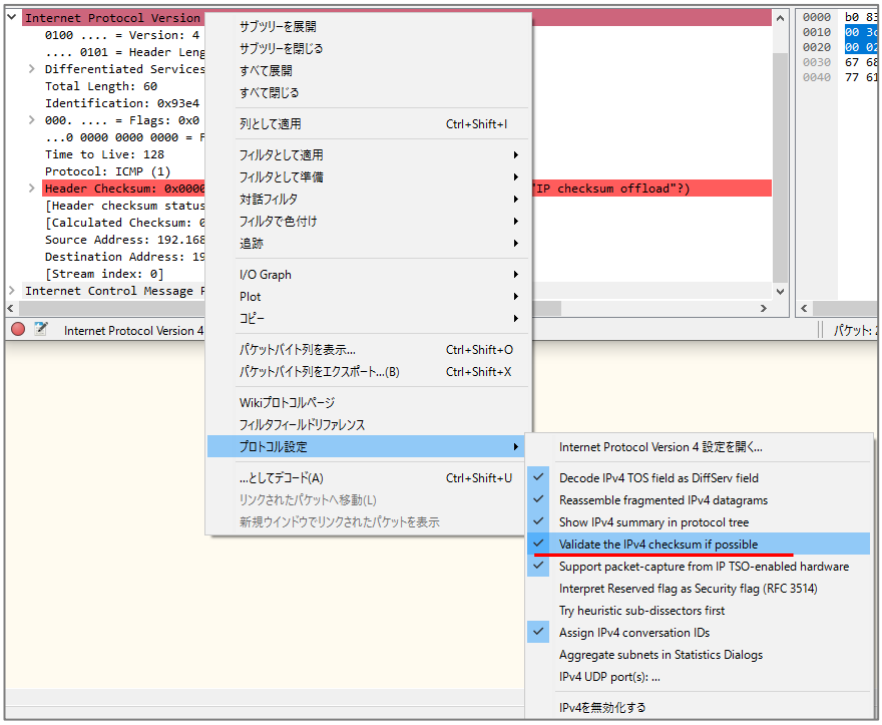
・OS（Windows の IP プロトコルスタック） Checksum=0x0000　で送信
・ネットワークインタフェースがハードウェアで Cheksum を計算、送信するデータ内に Checksum を埋め込み
・LAN ケーブルやハブなどの物理的な伝送経路を通過　★データが化けるとするとこの部分
・通信相手のネットワークインタフェースで受信（RX 側の IPv4 Checksum Offload=Enable の場合はこの部分で Checksum 値の検証）
・通信相手のソフトウェア（OS やアプリケーション）が受信データを処理

Wireshark で観測した際に、Checksum が 0x0000 になっていても、実際の通信においてはチェックサムは有効になっているという認識で良いかと考えます。

また、Wireshark で観測した際に、送信データが 60 バイトに満たない場合でもネットワークインタフェースがパディングデータを追加して、Ethernet ネットワークに流れているデータとしては規格を満たすようになっていると考えます。

→PC から送信するデータに関しては、その PC 上で動作する Wireshark でパケットダンプしても、Ethernet を流れるデータとは一致しない事に注意してください

[参考] Wireshark でのチェックサムの検証



Wireshark では、デフォルトではチェックサムをチェックしない設定になっています。
プロトコル設定－Validate the IPv4 checksum if possible
にチェックを入れた場合、チェックサムの検証が有効となります。
（有効にした場合、Wireshark を実行している PC から送信したデータのチェックサムはエラー判定（赤色）となりますが、マイコンボードから送信したチェックサム値が合っているかの確認のために、チェックサム検証を有効化するのが推奨です。）
（一般的には、チェックサムはハードウェアで付加されるので確認は不要ですが、本キットではマイコンボード側のチェックサムはユーザプログラムで計算していますので、プログラムのデバッグの過程では計算が間違っている事もままあります。その際に、Wireshark 側のチェックサム検証を有効にしておけば間違いに気付けます。）
（上記は、IPv4 の IP ヘッダ内のチェックサム検証ですが、プロトコル毎にチェックサム検証を有効化する設定があります。設定は一括ではなくてプロトコル毎です。扱うプロトコル毎に必要に応じて、チェックサム検証を有効化してください。）

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

商標等の表記について

・　全ての商標及び登録商標はそれぞれの所有者に帰属します。

・　パーソナルコンピュータを PC と称します。

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(1)

株式会社 北斗電子

©2026 北斗電子 Printed in Japan 2026 年 6 月 5 日改訂 REV.1.0.0.0 (260605)