はじめに

昨年度（平成 26 年度）に開発したテキスト「次世代自動車基礎教材」の続編を作成することとなりました。今年度は、「演習編」を作りたいという要望があり、その中にスマート家電の標準規格となっている ECHONET Lite の学習ができる演習を盛り込んだものを、実習教材キットも含めて開発するという、かなり大変な仕事を引き受けてしまいましたが、何とかまとまったテキストの原稿は、昨年度のものよりも 70 ページくらい多くなりました。規格や省庁の公開文書の掲載があるので、それほどのボリュームとは思っていないのですが、東北電子専門学校のご担当の先生方や、プロジェクトの担当者の方々、事務局の方々の理解に支えられて、このようなテキストを 2 冊も作成できたことは、感慨深いものがあります。

このテキストは先にも述べたとおり、スマート家電を学ぶ方々に対して提供されるものですが、既に社会人となられた方々やあるいは大学生にも、十分興味をもっていただけるのではないかと思います。ECHONET Lite を学ぶためには、Ethernet の基本を理解していることが必要なので、第 1 章はネットワークの基礎としていますが、これを編集することで私自身の復習にもなりました。2 章の ECHONET Lite の規格書説明は、学生のみなさんが経験したことのない、実社会に受け入れられる仕様書というものを読み込む良い機会となるのではないでしょうか。3 章以後では、様々な開発ツールとそれを利用した開発例を、実習キットを活用して学習していただければ、6 章を終えた段階でスマート家電の開発が習得できていると思います。

7 章は集大成ということで、次世代自動車に期待される V2H の話しとモデル開発を解説しました。V2H のモデルで使用するパーツがほぼ海外からのもので、この前書きを書いている時点で、まだすべてのものが揃っていませんが、実証授業や成果報告などを行うときには、皆さんに DEMO できる「モデル V2H カー」が出来上がっていると思います。このモデルをいろいろと使い、また改造などをしているうちに、組込マイコン制御システムの開発スキルが身に付き、スマート家電だけに留まることなく、多方面に活躍できるエンジニアが育つことを期待しています。

プロジェクト開発分科会
開発担当：　有限会社ワイズマン　原田賢一

目次

１章　ネットワークの基礎

現代の自動車には、多くのコンピュータが使われていて、CAN、LIN、FlexRay などでネットワーク化されています。次世代自動車は、HEMS でスマートハウスに接続され、そのスマートハウスも様々な家電や住宅設備との接続が行われています。そのようなネットワークの基本をここでもう一度振り返ってみましょう。読み物としても面白くて、いつの間にかネットワークの基礎が入ってくる・・・ハズです。

１章１節　IP（Internet Protocol）

IP は、とても重要なネットワーク技術です。Web やメールの通信ができるのも、IP に依存しています。IP の全体像をつかみましょう。

１．１．１．さまざまな伝送技術とアプリを取り持つ中核のプロトコル

〔1〕IP の全体像

IP は「internet protocol」の略で、プロトコルの一種です。その目的は、いくつものネットワークを通じてつながっているコンピュータの間で、アプリケーションのデータをやりとりすることです。

IP の仕様は「IETF」という標準化団体が決めています。1978 年に作られた「IPv4」と 1995 年に作られた「IPv6」があり、今回の話題の中心は、現在主に使われている IPv4 です。

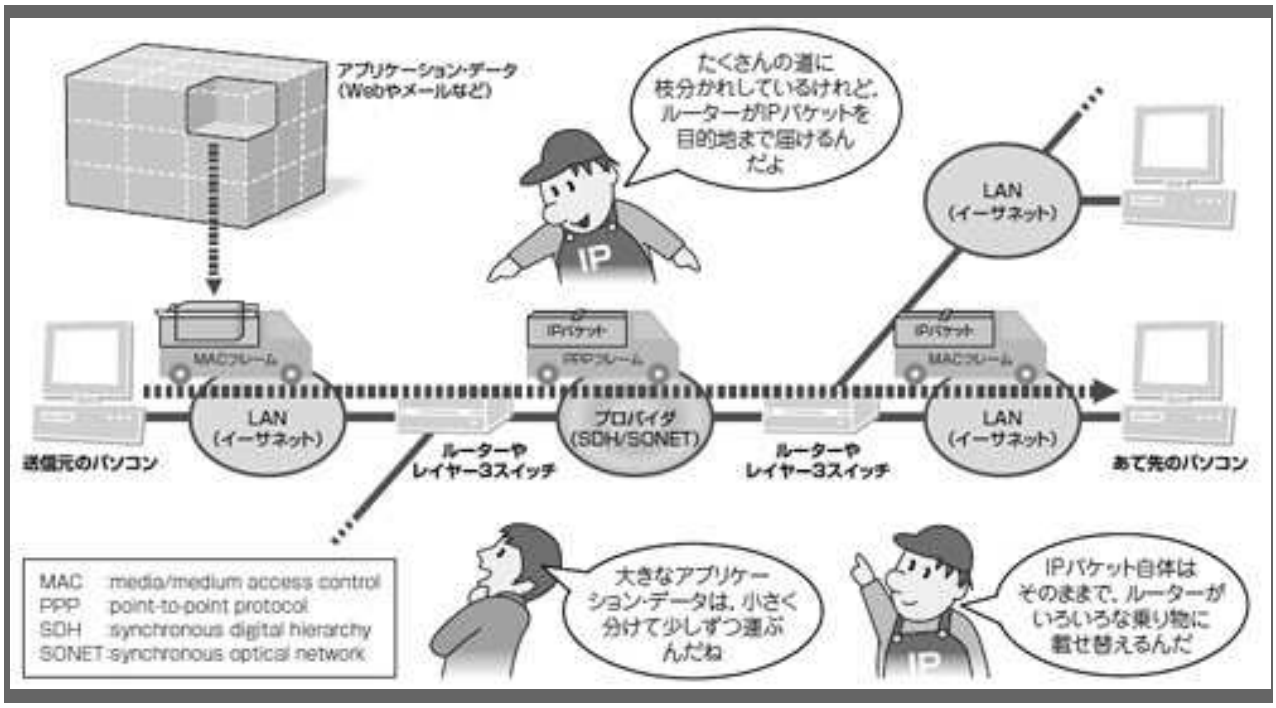
IP は、場所やアプリケーションの種類を問わず、あらゆる場面で活躍しています。

場所を見ると、インターネットをはじめとして、家庭や企業の LAN、IP 電話事業者、通信事業者のネットワークなど、いたるところで IP が使われています。アプリケーションに目を移すと、一般になじみのある Web や電子メールのほか、IP 電話や映像配信など、従来は別のプロトコルで実現していたアプリケーションも、今では IP を利用しているところが多くなっています。いわば IP は「プロトコルの万能選手」です。

〔2〕IP パケットを中継するルーター

IP は、「IP パケット」という単位でデータをやりとりします。パソコンやサーバー上で動作するアプリケーションが IP を使って通信する場合、どんなアプリケーションのデータでもいったん IP パケットに詰め込んで、送り出すことになります（図 1-1）。

こうすることで、アプリケーションの種類を問わず、共通の方法でデータを運べるのです。



（図 1-1：データを入れた IP パケットをルーターが届ける）

IP の基本的な役割は、アプリケーションのデータを IP パケットに入れて、コンピュータからほかのコンピュータへ届けることです。途中でさまざまな種類の伝送路が使われていますが、ルーターが IP パケットをそのまま載せ替えることで、エンド・ツー・エンドのアプリケーション通信を可能にしています。

パケットは小包という意味を持っています。その名の通り、IP パケットを小包に例えると、アプリケーションのデータはその中に詰める荷物ということになります。

IP パケットを運ぶ役割を果たすのは、「ルーター」という装置です。最近では、このルーターと LAN スイッチの機能を兼ね備えた「レイヤー3 スイッチ」というネットワーク機器も、IP パケットを運ぶ装置として数多く使われています。

IP パケットを受け取ったルーターは、適切な転送先を自身で判断して IP パケットを中継します。いくつものルーターがこうした処理を繰り返すことで、IP パケットはバケツ・リレーのように転送され、最終的な宛先に届けられます。

ルーターは、さまざまなネットワークで使われている伝送技術の違いを吸収する役割も果たしています。例えば、個人宅の LAN ではイーサネットや無線 LAN が使われているし、プロバイダのネットワークでは光ファイバによる高速伝送技術が活躍しています。ルーターは、これらの異なった伝送技術を使うネットワークを相互接続して、透過的に IP パケットを転送します。

IP パケットを小包になぞらえると、伝送技術はトラックに当たります。小包を運ぶトラックが途中で変わっても、小包自体は変わらずに、最終目的地に届きます。

〔3〕IP はプロトコルの中心

今度は、別の角度から IP の全体像を眺めてみましょう（図 1-2）。

（図 1-2：プロトコルの中心となる IP）

プロトコルの階層構造で見た IP の位置付けです。下位のプロトコルの違いを吸収し、上位のプロトコルを統一的に扱うのが IP の役割です。

通信はいくつものプロトコルを組み合わせて実現しますが、それらのプロトコルは分担する機能ごとにレイヤー（階層）状になっています。

IP のレイヤーは、複数のネットワークを介してつながっているコンピュータ間で、IP パケットを受け渡すという機能を受け持っています。IP の上位にあるレイヤーと下位にあるレイヤーもそれぞれの役割があります。

IP をレイヤーとしてみると、下位の伝送技術の違いを吸収して、上位のアプリケーションに共通したデータ転送の手段を提供しています。IP のおかげで、さまざまな伝送技術を使い、さまざまなアプリケーションの通信を実現することができるのです。

1．1．2．IP ヘッダーとデータ部で構成される IP パケット

〔1〕IP パケットの構造

IP パケットは「IP ヘッダー」と「データ部」という二つの部分で成り立っています。IP ヘッダーが前に、データ部が後ろになっています。

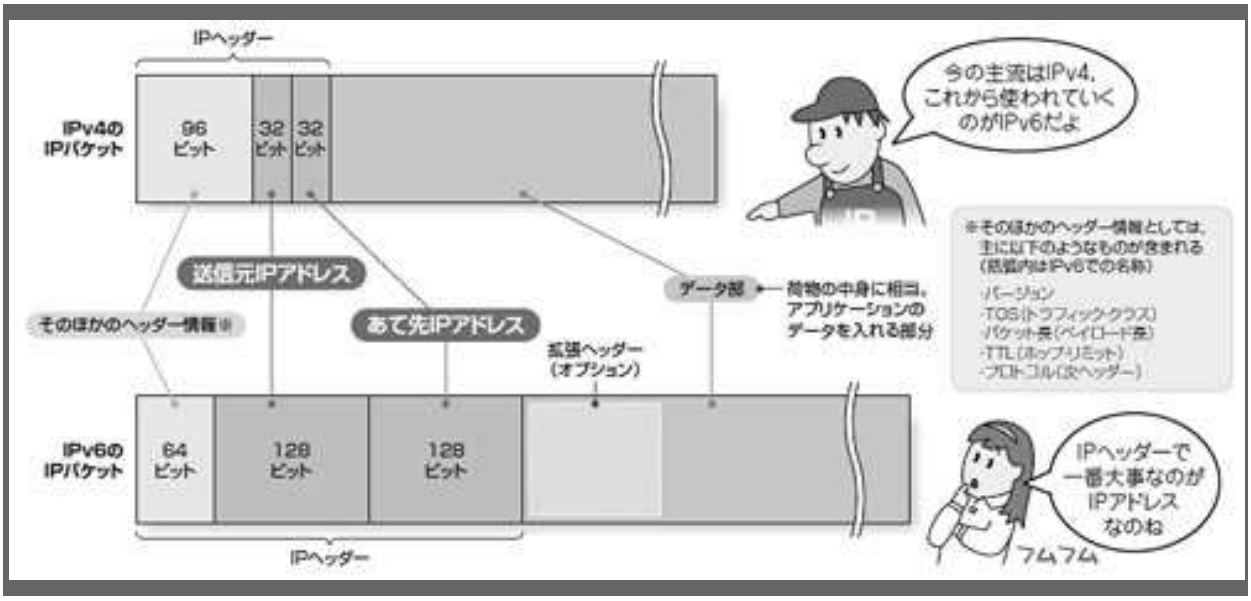
IP ヘッダーは、IP パケットを届けるために使われるさまざまな情報が入っています。データ部には、アプリケーションのデータが入れられます。【1.1.1】で説明した

「荷物の中身」に相当するのが、このデータ部です。

〔2〕IP アドレスで場所を示す

では、IP ヘッダーにはどんな情報が詰まっているのでしょうか。

最も重要な情報は、二つの IP アドレスです。その中でも、あて先 IP アドレスが重要です（図 2-1）。



（図 2-1：IP パケットの中身）

IP パケットは、IP ヘッダーとデータ部から成り立っています。IP ヘッダーには、IP アドレスなどの情報があります。データ部には、アプリケーションのデータが入れられています。

IP アドレスは、ネットワーク上のコンピュータの"位置"を示す情報です。各コンピュータにそれぞれ異なる IP アドレスが割り当てられます。IP ヘッダーに含まれるあて先 IP アドレスは、IP パケットを届ける相手のコンピュータを示しています。ルーターは IP パケットの転送先を決めるときにこの情報を利用 するのです。

送信元 IP アドレスは、IP パケットを送り出すコンピュータ自身の IP アドレスです。送信元 IP アドレスはおもに、あて先のパソコンやサーバーが受け取った IP パケットに対して返信するときに使われます。

IP アドレスの長さは、IPv4 と IPv6 で異なります。IPv4 は 32 ビット長、IPv6 は 128 ビット長のアドレスを使います。IPv4 ではアドレス数が足りなくなりそうなので、アドレス長を長くしてアドレス数を増やした IPv6 が開発されています。

〔3〕大切なヘッダー情報

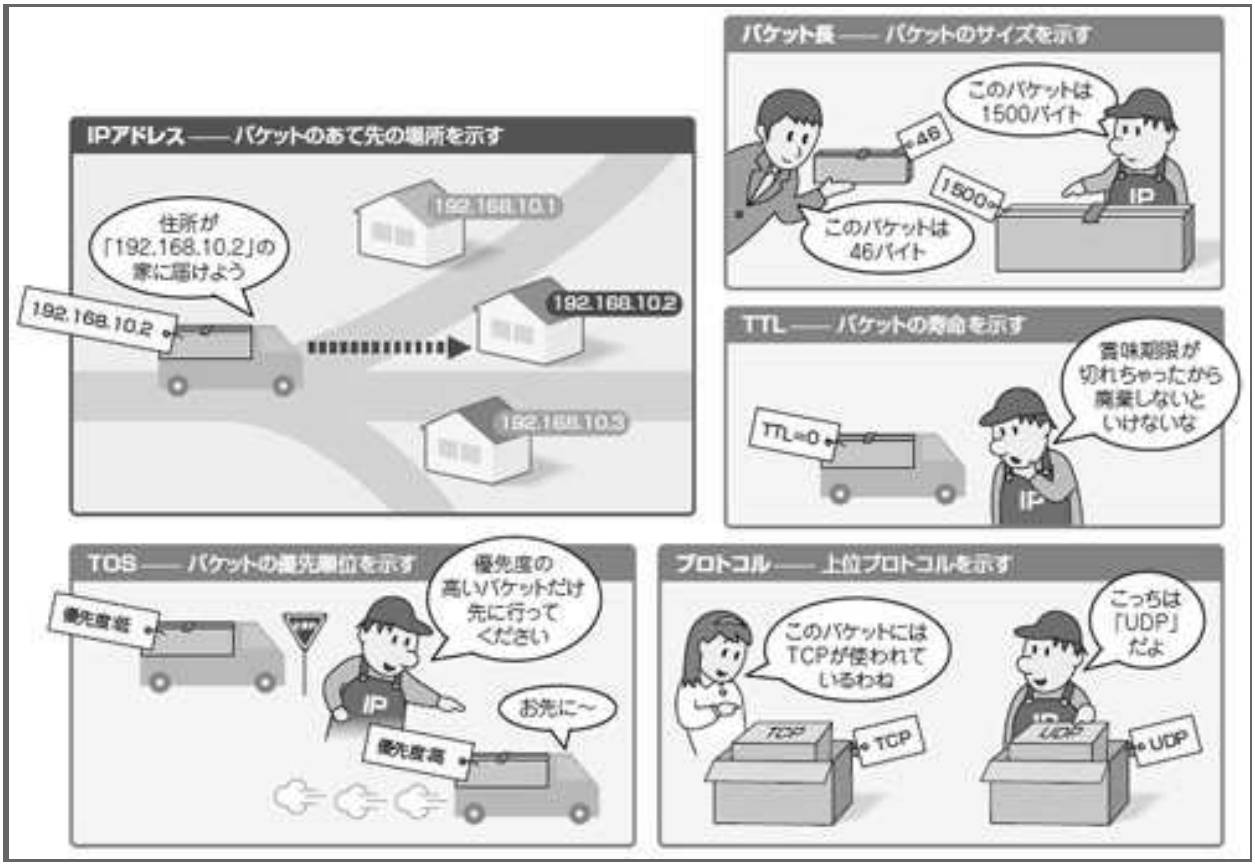
IP ヘッダーには、IP アドレス以外にもさまざまな情報が含まれています。代表例を挙げると、「バージョン」、「TOS」、「パケット長」、「TTL」、「プロトコル」といった

情報です。これらの情報は、IP パケットを適切に転送したり、受信側で IP パケットを処理したりするときに使われます。

これらがどのように利用されているのか、簡単にチェックしましょう。

バージョンは、IP のバージョンを示す情報です。IPv4 なら「4」、IPv6 なら「6」となります。

パケット長は、IP パケットの長さ（サイズ）を示します（図 2-2）。これは IP ヘッダーとデータ部の両方を含む全体のバイト数です。



（図 2-2●IP ヘッダーにある情報の意味や利用方法）

IP ヘッダーにあるさまざまな情報は、IP パケットを宛先まで届けるためや、届けたあとの処理をうまく行うために使われます。

プロトコルは、IP の上位プロトコルを示します。ほとんどの場合、「TCP」と「UDP」のいずれかになります。TCP は、途中で失われたデータを再送することなどで、確実にデータを届ける役割を果たします。Web やメールはこの TCP を利用しています。一方の UDP は、再送などの処理は行わずに、できるだけ遅延を少なくします。主に IP 電話や映像配信などに使われていますが、後の章で述べる、スマート家電の標準プロトコルとして、UDP が使われています。

TTL は「time to live」（生存時間）の略で、IP パケットが越えられるルーターの台数を示す値です。パソコンが IP パケットを送り出すときに 32 などの値に設定され、

ルーターによってIPパケットが中継される度に1ずつ減らされます。TTLが「0」になった IP パケットはルーターで廃棄されます。こうしたしくみによって、行き先が見つからない IP パケットがいつまでもネットワークを徘徊し、余分に帯域を消費することを防いでいます。

TOSは、IPパケットのタイプを区別するための値です。通常はIPパケットを中継するときの優先度を示すために使われます。例えば、相手に届くのが遅れると音声品質を低下させる IP 電話のパケットに高い優先度を付け、ルーターが優先して転送するようにしています。

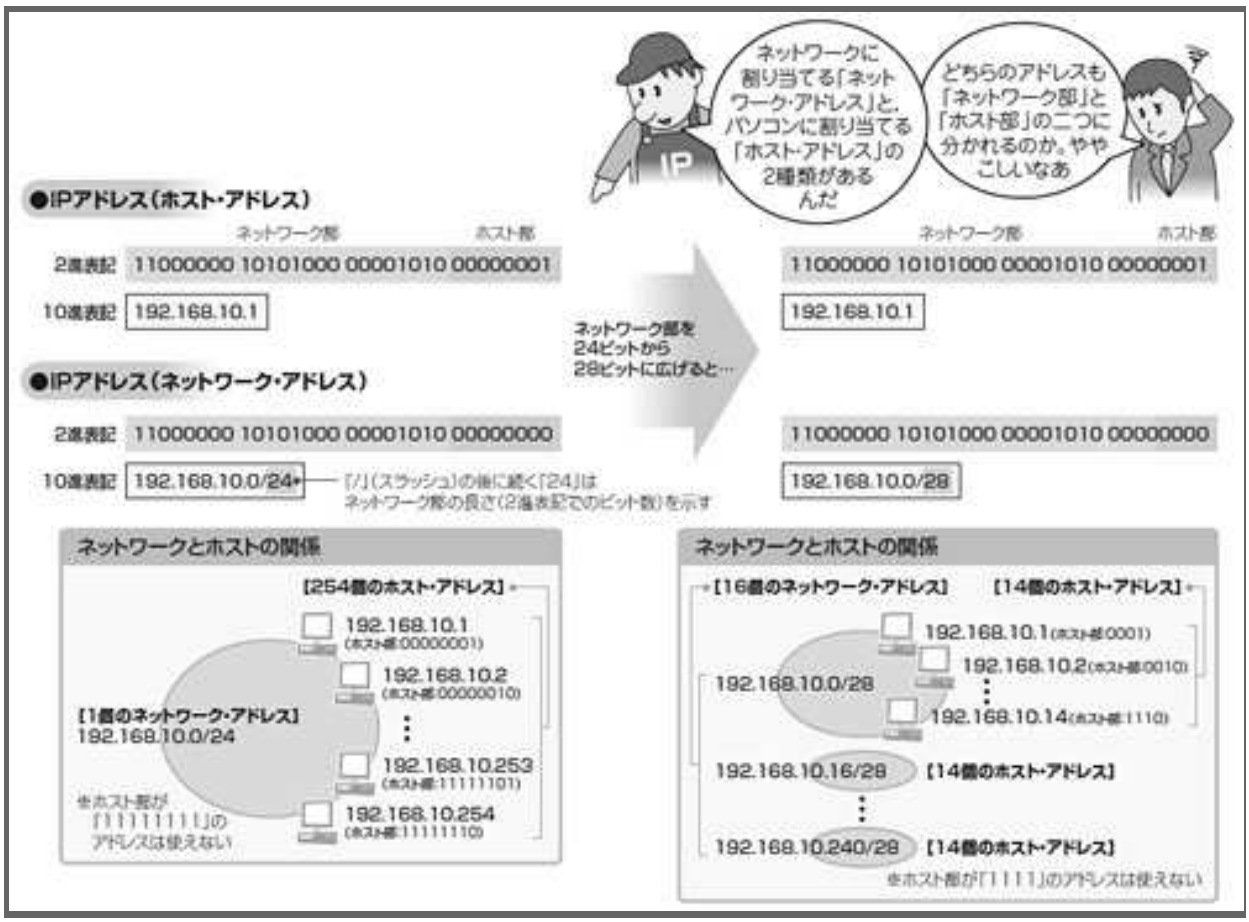
１．１．３．IPアドレスでネットワークとその中のホストの場所を示す

〔1〕IPアドレス

IPアドレスはIPパケットのあて先を示すために使われる情報です。パソコンやサーバーなどのコンピュータには、1台ずつ異なるIPアドレスを割り当てます。また、そのようなコンピュータが所属するネットワークにも、IP アドレスを割り当てます。コンピュータ（ホスト）に割り当てる IP アドレスを 「ホスト・アドレス」、ネットワークに割り当てるIPアドレスを「ネットワーク・アドレス」と呼んでいます。

IPアドレスは基本的には「0」と「1」が32個並んだ2進数のビット列ですが、人が見やすいように四つの10進数を組にして表記するのが一般的です。

IPアドレスは、ネットワークを表す「ネットワーク部」とホストを表す「ホスト部」で構成されています。（図３-1）の 左側に示したのは、「192.168.10.1」というホスト・アドレスと、「192.168.10.0/24」というネットワーク・アドレスです。 ネットワーク・アドレスにある「/24」は、ネットワーク部の長さを示す値です。24 ビットがネットワーク部、残りの8ビットがホスト部になります。この場合、256（2の8乗）から2を引いた254個のホスト・アドレスを作ることができます。



（図 3-1：ネットワークとパソコンに割り当てる IP アドレス）

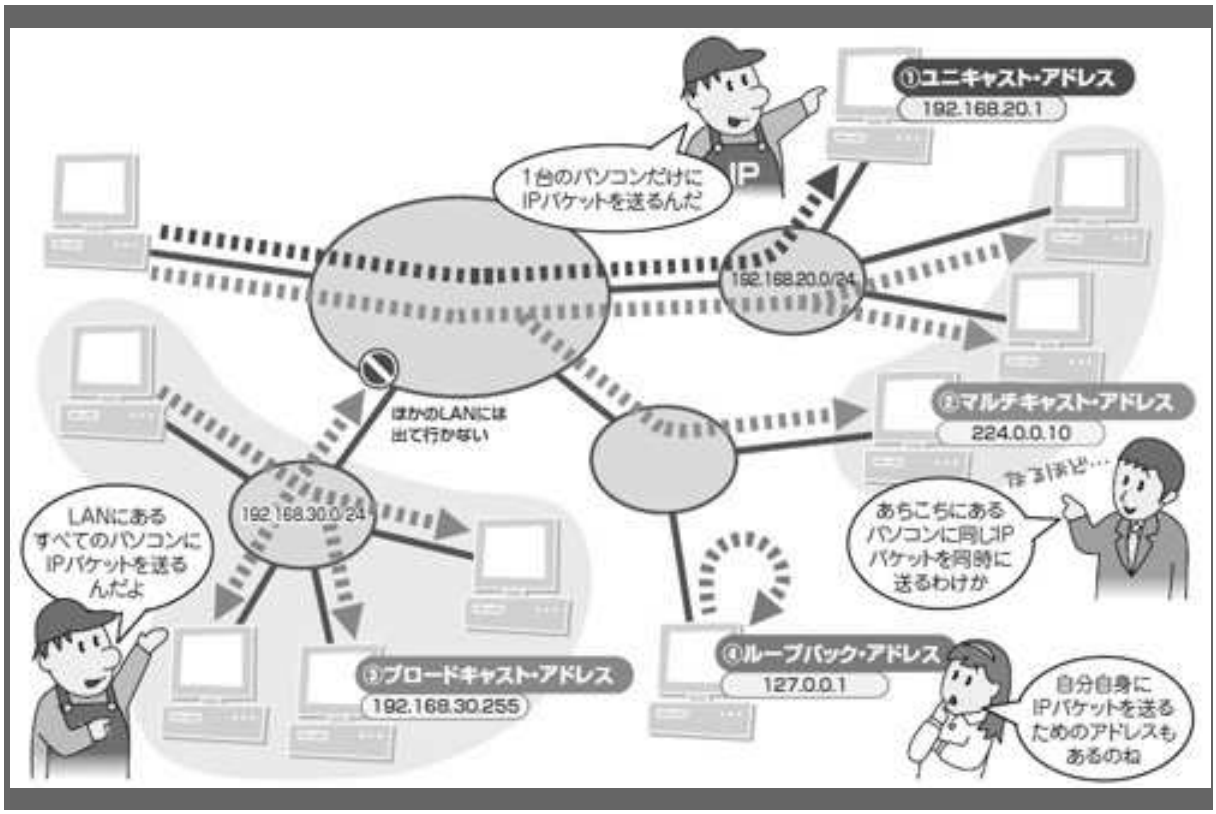
IP アドレスには、ネットワークの場所を示す「ネットワーク・アドレス」と、パソコンなどの端末の場所を示す「ホスト・アドレス」があります。さらに、それぞれ ネットワーク部とホスト部に分かれます。ホスト・アドレスのホスト部は端末ごとに変えます。ネットワーク・アドレスのホスト部は、すべて「0」にします。

ネットワーク部とホスト部の関係を理解するために、「192.168.10.0/24」のネットワーク部の長さを変えてみましょう。アドレスの末尾の 8 ビットを自由に変えられるという条件で、ネットワーク部の長さを 24 ビットから 28 ビットに長くします。

ホスト部の長さは 4 ビットに短くなるため、ホスト・アドレスを割り当てられるパソコンの台数は、16（2 の 4 乗）から 2 を引いた 14 台となります。一方、ネットワーク部の末尾 4 ビットを変えられるので、ネットワーク・アドレスは 16 個設けられます。

〔2〕通信方式ごとに用意されているアドレス

IP には、通信方式に対応して 4 種類のアドレスが用意されています。(1) ユニキャスト・アドレス、(2) マルチキャスト・アドレス、(3) ブロードキャスト・アドレス、そして (4) ループバック・アドレス――です（図 3-2）。



（図 3-2：IP パケットの送り方に合わせて定義された 4 種類の IP アドレス）

IP（IPv4）では、IP パケットの送り方によって、（1）ユニキャスト・アドレス、（2）マルチキャスト・アドレス、（3）ブロードキャスト・アドレス、（4）ループバック・アドレス--という 4 種類の IP アドレスが定義されています。

（1）のユニキャスト・アドレスは、【1.1.3】の前半で説明してきたアドレスのことで、2 台のコンピュータ同士が 1 対 1 で IP パケットを送受信するために使われます。

（2）のマルチキャスト・アドレスは、特定のグループに所属するパソコンだけに、同じデータを同時に送信するために使います。例えば、映像配信をパソコンで受信するような場合、パソコンにマルチキャスト・アドレスを割り当てます。ルーターは、パソコンからの要求に応じて、マルチキャストのパケットを送るようになります。このほか、特定の機器がマルチキャストを利用する場合などは、その機器に決められたマルチキャスト・アドレスを割り当てておきます。

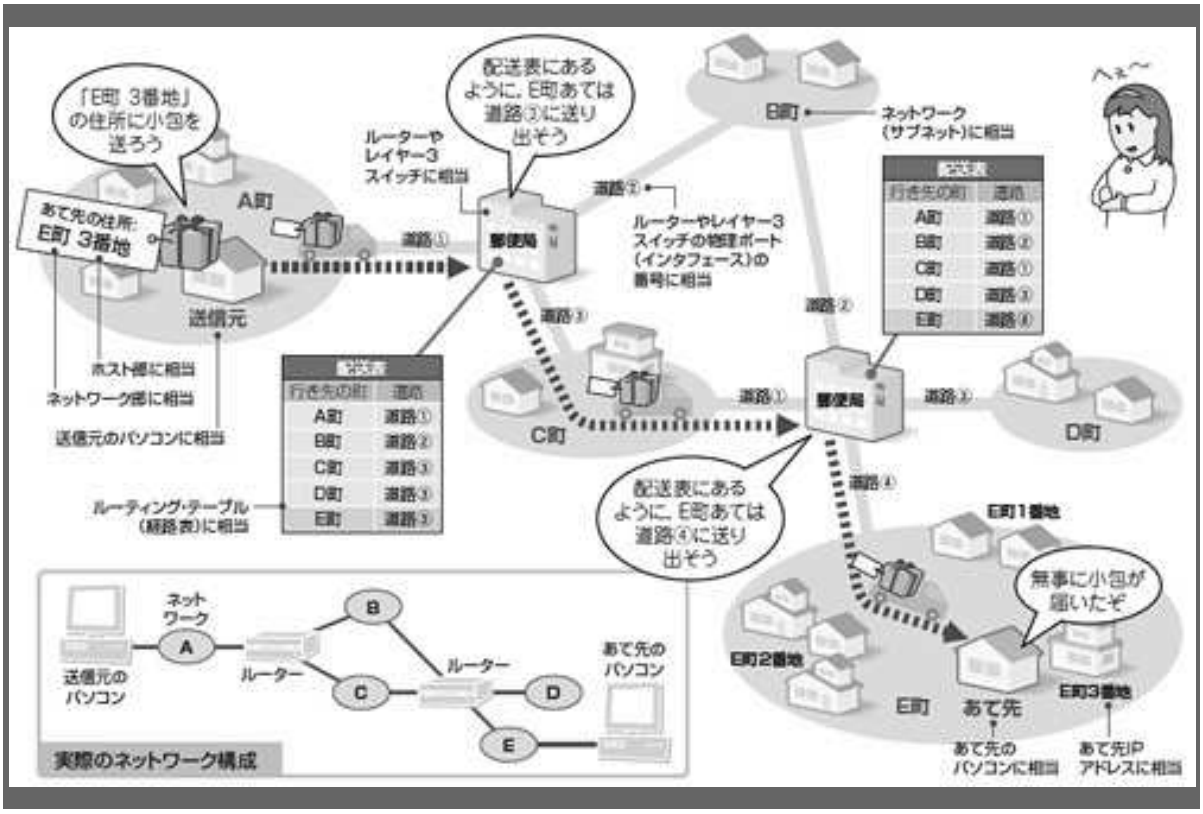
（3）のブロードキャスト・アドレスは、LAN 内のすべての端末に同じデータを同時に送信するときに使われます。

最後の（4）は、自分自身を示すためのアドレスです。

１．１．４．あて先ネットワークだけを見て転送先を決めるルーティング

〔1〕ルーティング

ルーティングは、IP パケットの転送経路を決め、目的地に届ける一連の処理のことです。小包を配送することになぞらえて説明するとわかりやすいので、その例えを使ってルーティングのしくみを理解していきましょう（図 4-1）。



（図 4-1：ルーティングの基本的な考え方）

IP パケットを小包、ルーターやレイヤー3 スイッチを郵便局になぞらえて説明しています。ポイントは、郵便局では最終的な住所（ホスト・アドレス）ではなく、おおまかな町単位（ネットワーク・アドレス）で配送先を把握していることです。

〔2〕配送表を見ながら小包を転送

ここで小包は当然、IP パケットを表します。（図 4-1）では小包を届ける相手を「E 町」の中の「3 番地」としています。「E 町」が IP アドレスのネットワーク部、「3 番地」がホスト部に相当します。小包のあて先を示す荷札は、あて先 IP アドレスに当たります。

また、小包の配送先を決める郵便局がルーターに相当します。小包に付けられた荷札に書いてあるあて先の住所を見て、どの町に転送するかを決めます。

では、送信元の家からあて先の家まで小包がルーティングされる一連の様子をたどっていきましょう。

最初に、A 町にある送信元の家から、A 町の最寄りの郵便局に小包が届けられます。

各郵便局には、最終的な行き先と、次に送り出す道路がセットになった配送表が用意されています。郵便局は、小包のあて先にある住所のうち、どの町に配送すべきかを確認します。さらに、行き先の町に対応する道路を配送表から調べて、その道路から小包を転送します。この例では、E 町行きの小包は C 町へ続く道路（3）に転送することになっています。なお、B 町経由でも E 町に到達できますが、その道路は配送表にないので使われることはありません。

この配送表は、実際の IP の世界では「ルーティング・テーブル」と呼ばれます。ルーティング・テーブルには、あて先ネットワークとルーターの物理ポートがセットになって登録されています。ルーターはこの表を参照して、IP パケットの転送先を決めます。

C 町を通って、次の郵便局に届いた小包。その郵便局も同様に配送表を見て、転送先を確認します。

最終的に小包を届けるあて先の家は、その郵便局と道路でじかにつながっている E 町にあるので、小包はその家に直接届けられます。

ここまでの説明の中に、ルーティングのしくみを理解するためのポイントが二つあります。

一つは、ルーターが転送先を決める際に利用するのは、あて先 IP アドレスのネットワーク部だけであること。今回の例で説明すると、小包を配送する際に「E 町 3 番地」までを見るのではなく、「E 町」という部分だけに着目します。

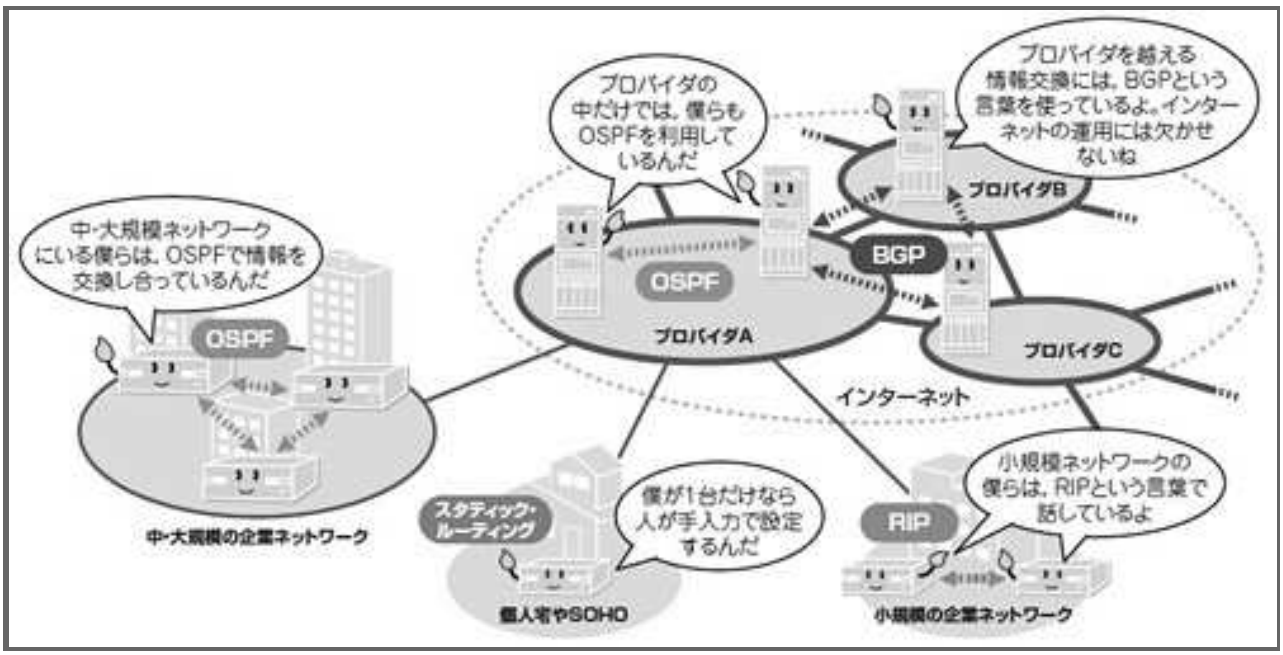
もう一つは、ルーターは最終的なあて先のネットワークを把握しているわけではなく、次の転送先だけを知っていることです。

〔3〕3 種類のプロトコル

ルーティング・テーブルの内容を登録する方法には、手作業でルーターに設定する「スタティック・ルーティング」と、ルーター同士が経路に関する情報をやりとりして、自動的に作る「ダイナミック・ルーティング」の 2 種類があります。

ルーターが 1 台しかない家庭 LAN のような場合は、スタティック・ルーティングを使います。複数のルーターで構築したネットワークは、ダイ ナミック・ルーティングで運用されます。その際にルーター同士の情報交換に使われるのが「ルーティング・プロトコル」と呼ばれるプロトコルです。

現在主に利用されているルーティング・プロトコルは「RIP」、「OSPF」、「BGP」の 3 種類です。これらは、場所ごとにうまく使い分けられています（図 4-2）。



（図 4-2：場所に応じて使い分けるルーティング・テーブルへの登録方法）

手動で登録する「スタティック・ルーティング」と、ルーティング・プロトコルを使って自動的に登録する「ダイナミック・ルーティング」があります。主に使われているルーティング・プロトコルには、「RIP」、「OSPF」、「BGP」の 3 種類があります。

１．１．５．チャレンジコーナー

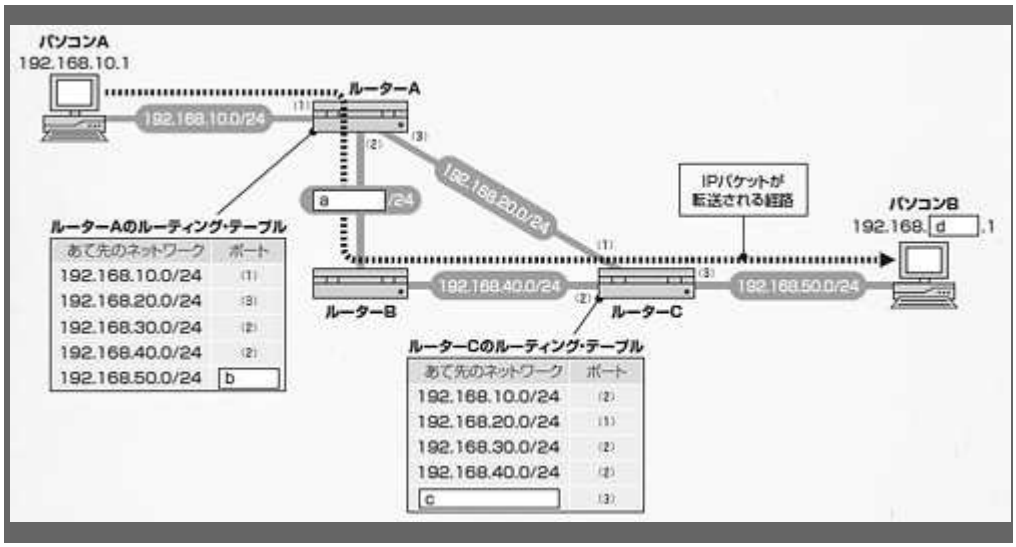
■問 1　以下は、IP を説明した文章です。空欄[　a　]～[　d　]にあてはまる字句を選択しなさい。

IP は、さまざまなアプリケーションのデータを[　a　]という単位で転送するプロトコルです。パソコンやサーバーなどの端末が作った[　a　]は、IP アドレスという情報に基づいて[　b　]やレイヤー 3 スイッチによって転送されます。IP アドレスには、端末の場所を示すホスト・アドレスとネットワークの場所を示す[　c　]があります。[　b　]は，ルーティング・テーブルを参照することで、[　a　]の転送先を決めます。そのルーティング・テーブルの内容を自動的に登録するには、RIP、[　d　]、BGP といったルーティング・プロトコルを利用します。

選択肢〔IP パケット、MAC フレーム、LAN スイッチ、ルーター、ネットワーク・アドレス、MAC アドレス、HTTP、DNS、OSPF〕

■問 2　以下の図は、3 台のルーターで構築したネットワークを使って、パソコン A からパソコン B まで IP パケットを転送する経路を示しています。

空欄[a]～[d]にあてはまる数字を選択しなさい。



[a] 192.168.10.0　192.168.20.0　192.168.30.0　192.168.40.0　192.168.50.0

[b] (1)　(2)　(3)

[c] 192.168.10.0/24　192.168.20.0/24　192.168.30.0/24　192.168.40.0/24
192.168.50.0/24

[d] 10　20　30　40　50

■問 3 [a] ルーターによる IP パケットの転送回数は、IP ヘッダーにある TTL(生存時間)という情報によって制限されます。この TTL の説明として間違っているものはどれか。

①. TTL の値は、IP パケットがルーターによって転送されるときに、1 減る

②. TTL の値が 0 になると、その IP パケットは廃棄される

③. TTL の初期値は 255 と決められている

④. TTL のしくみを使うことにより、行き先の見つからない IP パケットがいつまでもネットワークを徘徊することを防ぐ

■問 3 [b] マルチキャスト・アドレスを正しく説明しているものはどれか。

①. マルチキャスト・アドレスはパソコンなどの端末に割り当ててルーターが管理する

②. マルチキャスト・アドレスは 127.0.0.1 である

③. あて先にマルチキャスト・アドレスを指定すると、同一 LAN 上のすべての端末に IP パケットが必ず届く

④. あて先にマルチキャスト・アドレスを指定すると、送信元のパソコンがグループに所属する端末ごとに同一パケットを 1 個ずつ送る

解答:

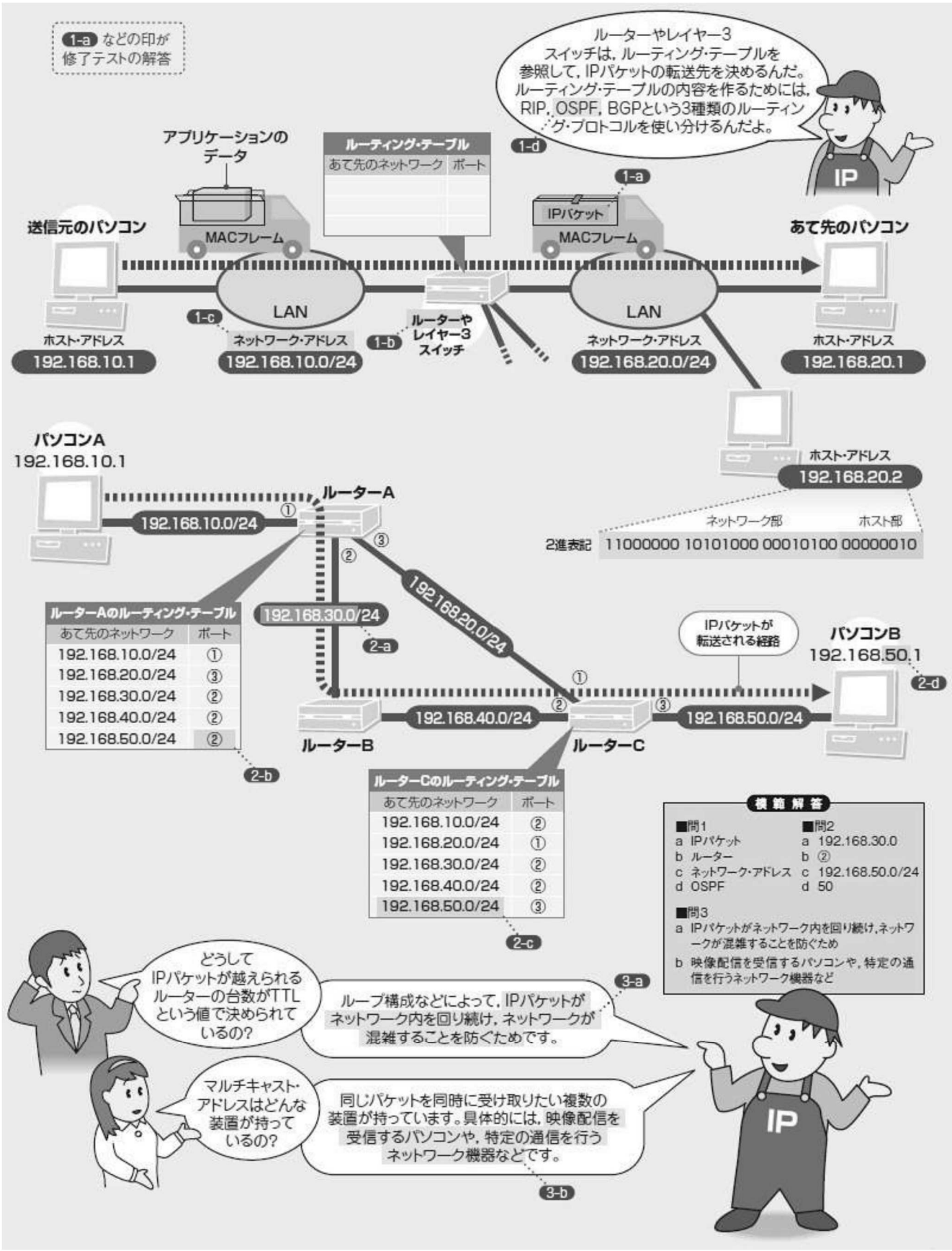


図:まとめ--IP のしくみ

１章２節　TCP（Transmission Control Protocol）

インターネットをはじめとする IP ネットワークは、データを複数のパケットに収めて送り出しても、その途中で一部のパケットが消えたり、順番が変わってしまったりすることが時々起ります。送ったパケットが確実に相手に届くとは限りません。

そこで、TCP（Transmission Control Protocol）というプロトコルが登場します。Web ブラウザと Web サーバーの間や、メーラーとメールサーバーの間など、データをきちんとやり取りする必要がある場所で利用されています。

IP ネットワークで通信するアプリケーションのほとんどは、データを誤りなく相手に届けるときに TCP を使います。データに TCP ヘッダーを付けて TCP パケットを作り、それを IP パケットに収めて相手に送ります。TCP はとても身近で、IP ネットワークには重要なネットワーク技術です。

１．２．１．TCP はアプリ間をつなぐパイプライン

TCP は、IP ネットワークを介して確実にデータをやりとりするためのプロトコルです。IP ネットワークの仕事は、パケットを通信相手に届けることです。IP ネットワークはその仕事に最善を尽くしますが、通信相手にパケットの抜けがなく、正しい順番で届くことまでは保証していません。

状況によってはネットワークが混雑して、送り出したパケットの一部が届かなかったりします。また、転送経路が変わってしまうこともあるので、送り出した順番通りにデータが届くとは限りません。

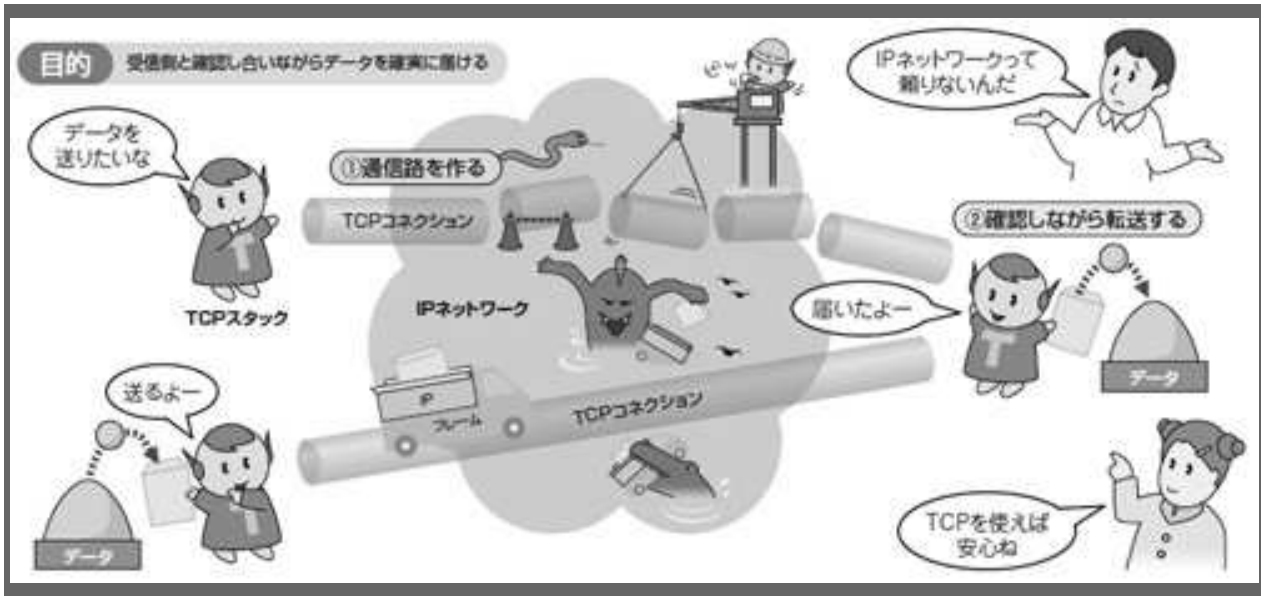
〔1〕パイプラインでデータを運ぶ

この不安定さは、データを 1 ビットでも間違うと問題が起こるアプリケーションにとってはとても困ります。メールやファイル転送といったアプリケーショ ンは、少しでもデータが抜けたり、順番が前後したりすると処理できなくなってしまいます。表示される文字が化けたり、最悪の場合プログラムが暴走したりしてしまいます。つまり IP ネットワークはそのままでは使い物にならなりません。

そこで活躍するのが TCP です。IP ネットワークにつながるパソコンはどれも、TCP に基づくソフトウエアである「TCP スタック」を備えていて、それを使ってデータをやりとりしているのです。

アプリケーションから見ると、TCP は、あてにならない IP ネットワーク上に自分専用のパイプラインを敷設して、データをやりとりしてくれるしかけに見えます（図 1-1）。パイプラインにデータを流し込めば、それがそのまま通信相手のアプリケー

ションに渡されるイメージです。



（図 1-1●TCP は頼りないネットワーク上でデータを確実に転送するためのプロトコル）

IP ネットワークでは送ったデータがきちんと相手に届くかはわかりません。そこで TCP は、通信相手との間に論理的な通信路を作り、確認しながら転送することで、データを相手に確実に届けます。

実際の TCP ではパイプラインに相当する専用の通信路を「TCP コネクション」と呼びます。アプリケーションの指示に基づいて TCP スタックは TCP コネクションを確立します（図 1-1 の ①）。TCP コネクションを確立したあと、TCP スタックはアプリケーションから渡されたデータを確実に通信相手まで届けます。そのために、互いの TCP スタックが確認し合いながら転送します（図 1-1 の ②）。

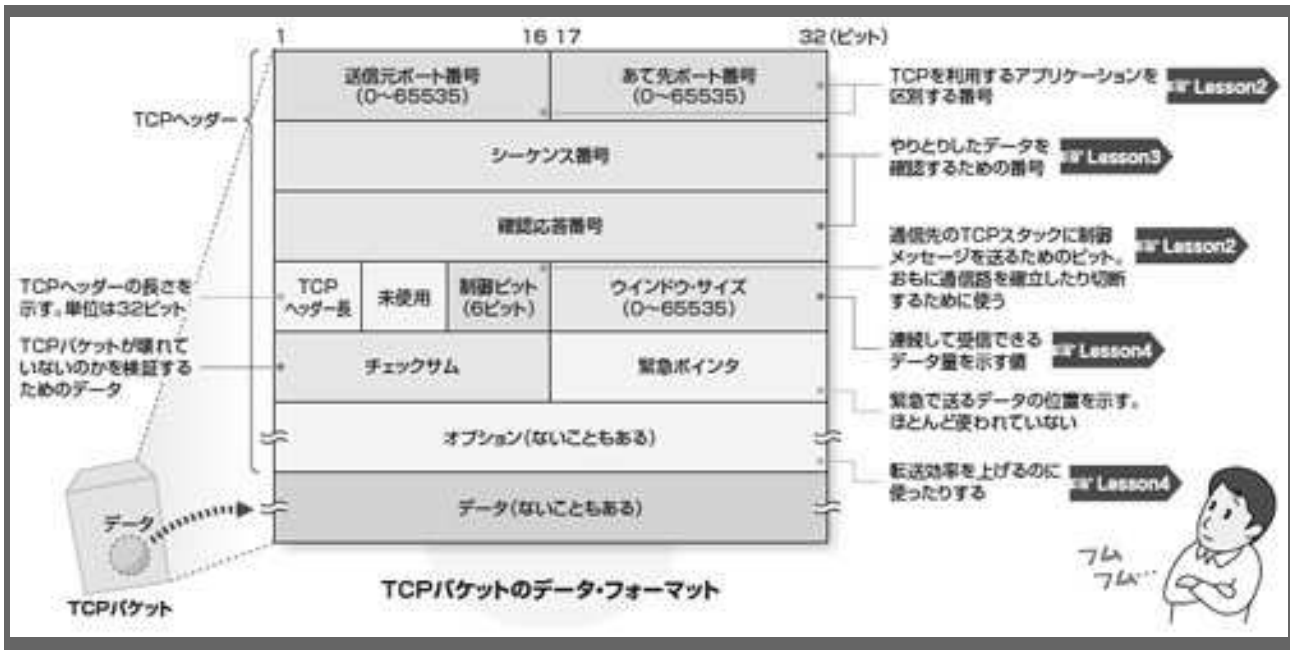
このように TCP が信頼できるデータ転送を提供することで、アプリケーションは IP ネットワークを意識しなくて済むのです。TCP スタックは、複数のアプリケーションが共同で利用できるようになっています。

〔2〕TCP ヘッダーがキモ

TCP コネクションはパイプラインのようなものですが、IP ネットワーク上に物理的なつながりができるわけではありません。あくまで TCP コネクションは論理的なものです。通信する双方の TCP スタックが、互いに「TCP コネクションを確立した」ということを確認し、約束し合うことで成立します。

TCP コネクションを確立したり、TCP コネクションを使って確実にデータを転送したりするための情報は、すべて TCP パケットのヘッダー部に書き込まれます（図

1-2)。実際にデータを送るときには、TCP ヘッダーに続けてデータが付けられます。



（図 1-2●TCP のさまざまな機能は TCP パケットで実現します）

TCP パケットは IP パケットのデータ部分に入れて送られます。データ以外の必要な情報は、すべて TCP パケットのヘッダーに含まれています

TCP ヘッダーのフォーマットは、どんなときでも基本的に同じです。TCP コネクションを確立したり、識別したりするための情報は、送信元ポート番号、あて先ポート番号、制御ビットという部分に書き込まれます（図 1-2 のオレンジ色）。一方、データを確実に送り届けるための情報は、シーケンス番号、確認応答番号、ウインドウ・サイズ、チェックサムといった部分に書き込まれます（図 1-2 の緑色）。

さらに TCP はヘッダーのオプション部分を使って、標準的な機能以外の機能を実現するための情報が書き込めるようになっています。

例えば、1 個のパケットで転送できるデータ・サイズを調整するための「最大セグメント・サイズ」や、連続して送れるデータ量を拡大するための「ウインドウ・スケール」、指定したパケットだけを再送してもらうための「SACK（サック）」（【1.2.4】で解説）といった情報は、オプション部分に書き込まれます。

オプションを拡張しながら、TCP は今でも進化し続けています。

このように、TCP の制御はすべて TCP ヘッダーを持つ TCP パケットのやりとりで実現します。この部分の細かい機能をこれから見ていくことにしましょう。

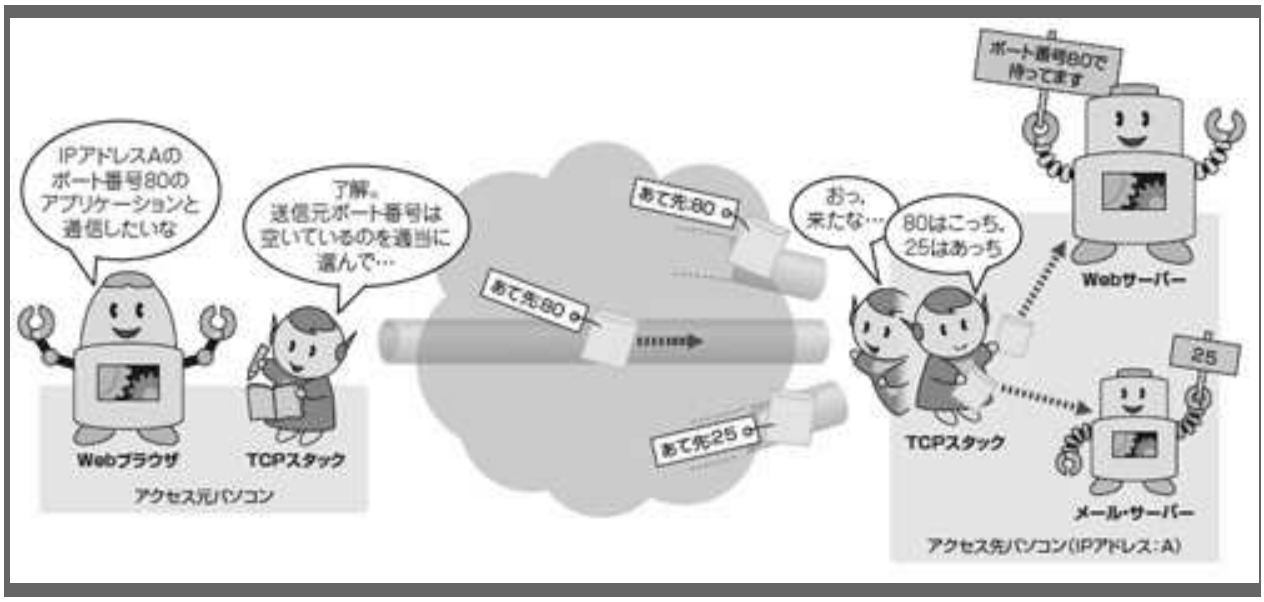
１．２．２．アプリの指定はポート番号で

TCP で通信するために、まずしなければならないのが TCP コネクションの確立です。

TCP コネクションは、アクセス元のアプリケーションからアクセス先のアプリケーションを呼び出す形で確立します。このときに、通信相手のパソコン内で動いているアプリケーションを間違いなく呼び出す必要があります。

〔1〕ポート番号でアプリを指定

TCP は、ポート番号という情報を使ってアプリケーションを指定するようになっています（図 2-1）。



（図 2-1：ポート番号でどのアプリケーションの通信かを指定します）

主要なサーバー・アプリケーションはポート番号が決まっていて、アクセス元のパソコンはその番号を宛先に指定することで的確にサーバーと通信できるようになっています。クライアント・アプリケーションのポート番号は、クライアント側の TCP スタックが空いている番号を選んで割り当てることが多い。

ポート番号はあて先ポート番号と送信元ポート番号が組になって TCP パケットに書き込まれます。アクセス元のアプリケーションは、相手のアプリケーションのあて先ポート番号をあらかじめ知っています。

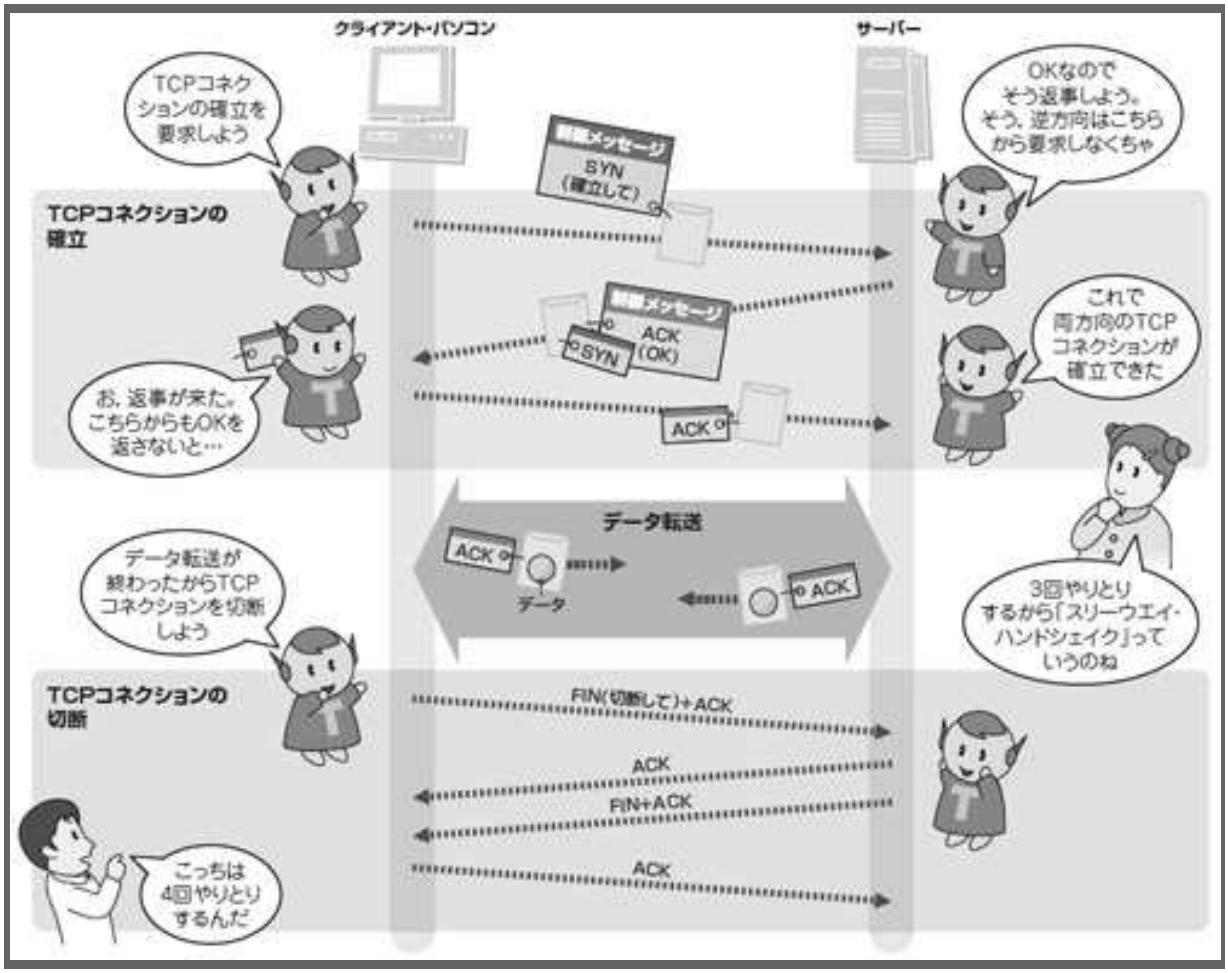
アクセス元のアプリケーションは、あて先ポート番号を指定して TCP コネクションの確立を依頼します。送信元ポート番号は特に指定しなくても、TCP スタックが空いている番号を割り当ててくれます。

TCP スタックは、相手の TCP スタックに対して TCP コネクションの確立を要求し

ます。アクセス先の TCP スタックは、あて先ポート番号を見て、コネクションをアクセス先のアプリケーションにつなぎます。このようにして、TCP コネクションは番号で明確に識別されます。

〔2〕３回のやりとりで確立する

次は、IP ネットワークで TCP コネクションがどうやって作られるのかを見てみましょう。TCP コネクションは、三つの TCP パケットをやりとりして確立します（図 2-2）。パケットが 3 回行き来するので、このやり方を**「スリーウエイ・ハンドシェイク」**とよんでいます。



（図 2-2：TCP コネクションは TCP パケットのやりとりで確立します）

確立するときはメッセージを託したパケットを 3 回やりとりします。このためスリーウエイ・ハンドシェイクともいわれます。切断するときは TCP パケットを 4 回やりとりします。

TCP スタックはまず、通信相手の TCP スタックに「コネクションを確立して」という意味の TCP パケットを送信します。「コネクションを確立し て」という意味を持たせるために、ヘッダー部の制御ビットのうち、5 ビット目の SYN（シン）（同期）

フラグを「1」にします。こうしたパケットを「SYN パケット」と呼びます。

SYN パケットを受け取った TCP スタックは、制御ビットの 2 ビット目の ACK（アック）（肯定応答）フラグと、SYN フラグを「1」にした 「SYN＋ACK パケット」を送り返します。SYN フラグも「1」にするのは、TCP コネクションには向きがあり、受け取った側が改めて逆方向でコネクションの確立を要求する必要があるからです。

最初に SYN パケットを送った TCP スタックは、相手からの SYN＋ACK パケットを見て、ACK パケットを送り返します。このパケットを相手が受け取ることで双方向の TCP コネクションが確立し、データ転送に移ります。

一通り通信が終わると、TCP スタックはアプリケーションからの指示でコネクションを切断します。ちなみに切断のことは開放ともいいます。切断するときは制御ビットの 6 ビット目の FIN（フィン）（終了）フラグと ACK フラグを「1」にした FIN＋ACK パケットを送信します。FIN＋ACK パケットを受け取った通信相手は、切断を了解したという意味の ACK パケットを送信して、改めて FIN＋ACK パケットを送信します。その返信の ACK パケットを受け取ることで、コネクションは切断されます。

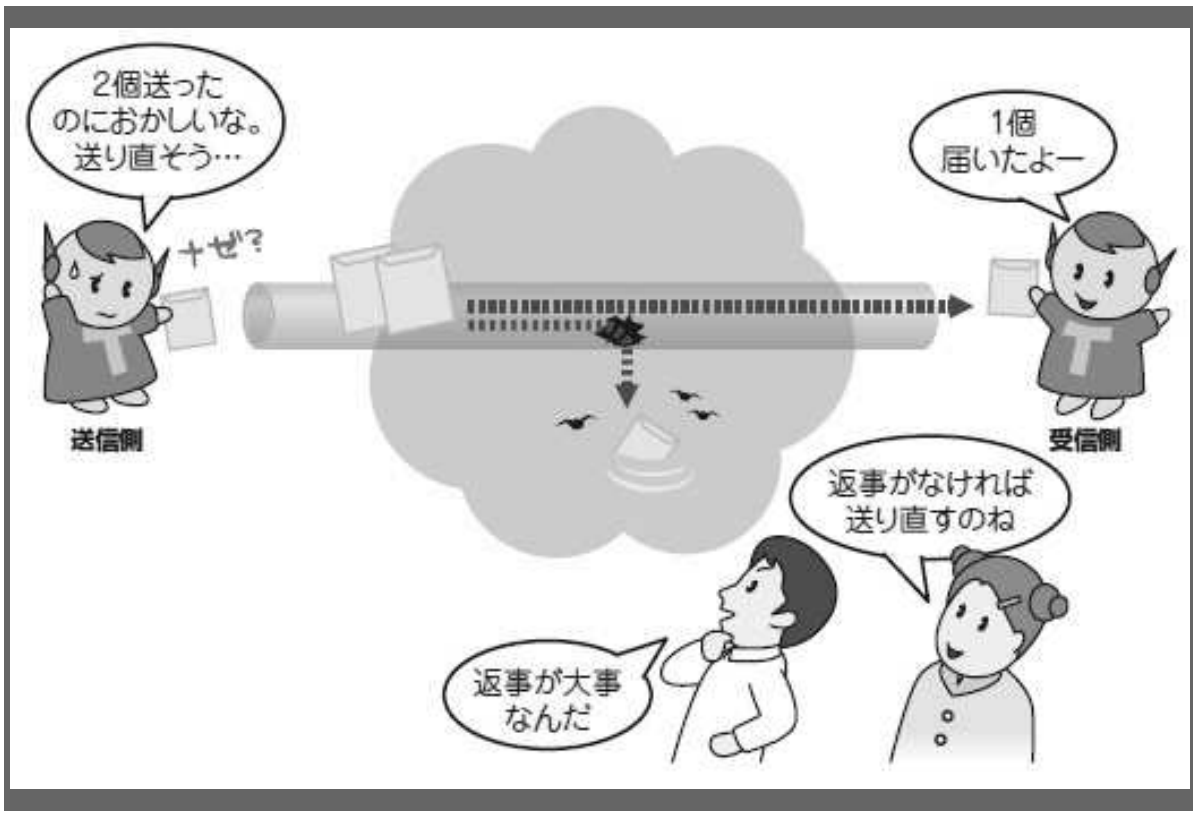
１．２．３．シーケンス番号と確認応答番号

TCP コネクションを確立したからといって、それだけでデータが間違いなく転送できるわけではありません。IP を使って TCP パケットを送る以上、ネットワークの中でパケットが失われたり順番が入れ替わったりすることがあるからです。

でも大丈夫。TCP は、パケットが失われたり順番が変わったことを検出して、再送したり正しい順番に入れ替えるメカニズムを備えています。

〔１〕返事が無ければ再送する

パケットが失われた場合、パケットが失われたことを見つけ出して、失われたパケットを再送する必要があります。そこで TCP スタックは、送信相手から TCP パケットが届いたという確認を送り返してもらいます（図 3-1）。送ったのに返事が来なければ届いていないとみなして、TCP パケットを送り直します。こうすれば、データの抜けはなくなります。



（図 3-1：届いたことを応答してもらい、応答がなかった分をもう一度送ります）

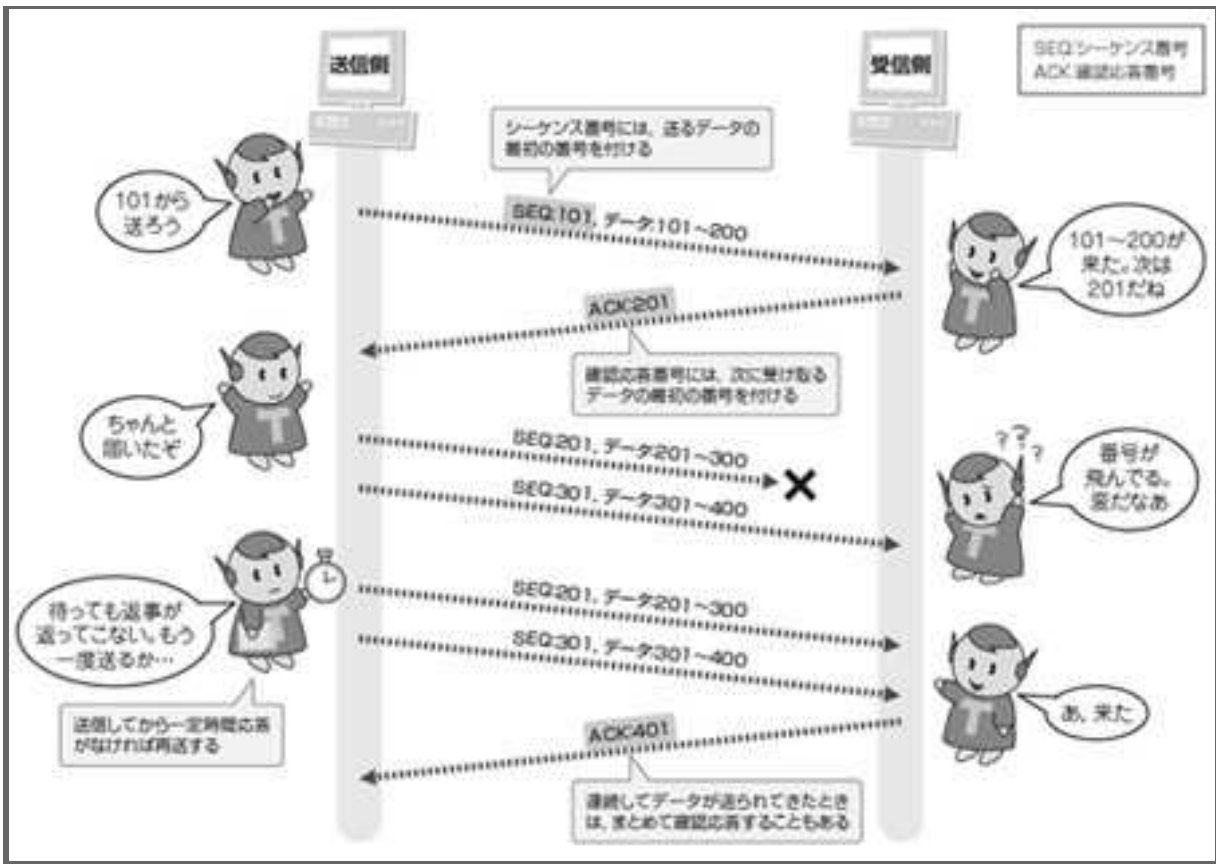
データを確実に転送するために、受信側が応答することになっています。応答がなければ再送します。

ここで問題になるのは、元のデータをいくつかのデータに分割して複数の TCP パケットで送る場合です。データの大きさが 1 個の TCP パケットで転送できる量を超えたりすると、連続した複数の TCP パケットを送信することになります。パケットの区別なく「届いた」というだけの返事がきても、どのパケットが届いたかがわかりません。

そこで TCP では、転送するデータをバイト単位でカウントし、その数値を連絡し合うことで、データがどこまで届いたのかがわかるようにしています。

〔2〕データの位置を数値で管理する

具体例で見てみましょう（図 3-2）。説明をわかりやすくするために、ここでは 1 個のパケットで 100 バイトのデータを転送する TCP コネクションを例に考えます。



（図 3-2：シーケンス番号と確認応答番号を使って確実に転送します）

送信側は確認応答番号を受け取ることで相手に届いたことを確認します。受信側から返事が届かないときは、直前に受け取った確認応答番号から相手に届いていないデータを推測し、再送します。

ヘッダーに含まれるシーケンス番号は、TCP コネクションができてから切れるまでの間、連続した数値が割り当てられます。シーケンス番号の初期値が 1 のケースで、送信側が 101 バイト目から 200 バイト目までのデータを送るとき、シーケンス番号は「101」となります。

このパケットをきちんと受け取った受信側は、次に受け取るはずのデータの先頭バイトを確認応答番号として送信します。200 バイト目まで受け取ったなら、確認応答番号に「201」と書き込んで送ることになります。送信側がこの「201」の確認応答番号を含む TCP パケットを受け取れば、200 バイト目まではきちんと転送できたことがわかります。

シーケンス番号は、受信側がパケットの順番をそろえるのにも利用されます。順番が入れ替わって届いても、シーケンス番号順に並べ替えれば元のデータが正確に再現できます。

〔3〕時間を待ってまとめて再送する

ネットワークの中で一つの TCP パケットが消えてしまった場合を考えてみましょ

う。受信側は確認応答番号を更新した TCP パケットを送信できません。そこで送信側は、データを送信したのに一定時間対応する確認応答が受け取れないときには、そのデータが届いていないと判断して、最後に受け取った確認応答番号のところからパケットを再送します。このため、受信 側に同じパケットが複数回届くこともあります。ちょっと効率は悪いのですが、重複したパケットは受信側で整理することになっています。

TCP コネクションではこのようにしてシーケンス番号とそれに対応する確認応答番号で確実にデータを転送します。ただし、データを送る TCP パケット一つずつに確認応答があるとは限りません。連続してパケットを受信したときは、複数のパケットにまとめて確認応答してもよいことになっています。

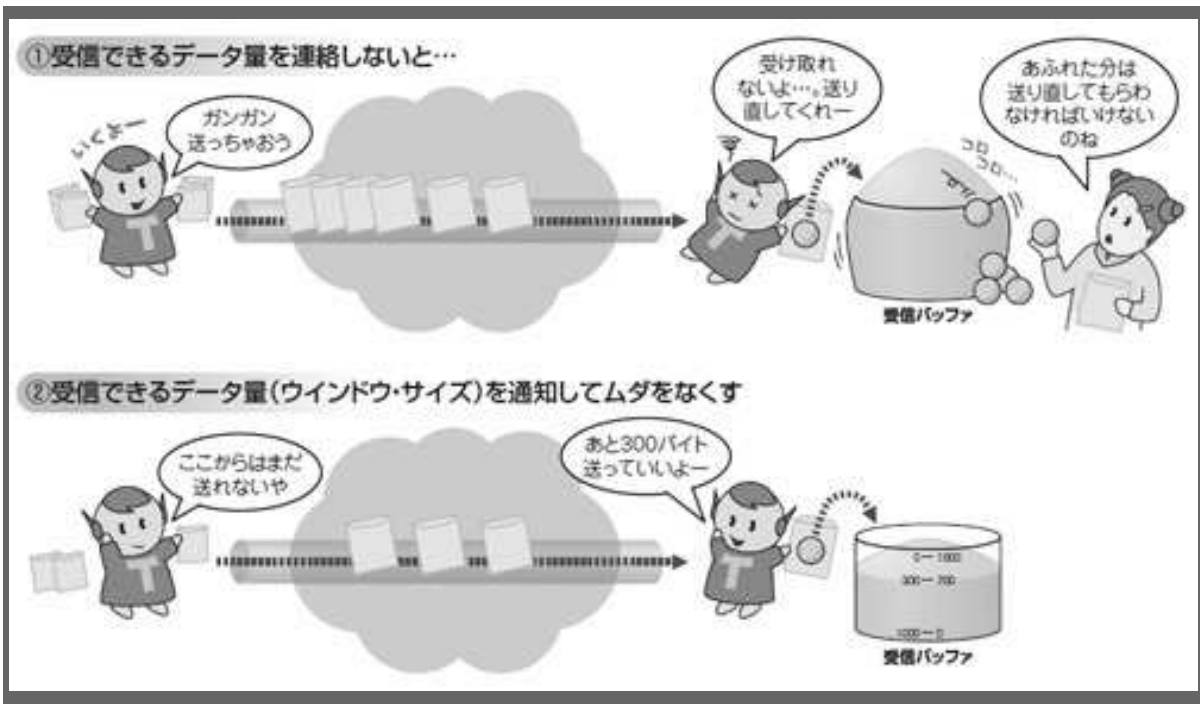
実際の TCP コネクションでは、データが双方向に流れます。このため、データを載せた TCP パケットが、逆方向のデータの確認応答を兼ねたりします。また、TCP パケットには、シーケンス番号と確認応答番号の両方が必ず書き込まれることになっているので、実際のやりとりはもう少し入り組んで見えます。しかし、基本を押さえたうえで見れば、難しいことをしているわけではないことがわかります。

１．２．４．転送効率アップの工夫

ここまでで TCP の基本は理解できたことでしょう。でも、TCP の世界は奥が深いんです。効率よくデータを転送するために、さまざまな工夫が加えられています。その一端を覗いてみましょう。ここでは、受信できる量を通知することでデータを効率よく送れるようにするフロー制御と、再送の効率を上げる工夫を紹介します。

〔1〕受信側から転送ペースを調節する

まずはフロー制御です。TCP では、データを載せた TCP パケットを連続して転送できます。ただし、際限なく連続して転送するとちょっと困ったことが起こります(図4-1)。



（図 4-1●受信できる量を通知することで転送効率を上げます）

TCP では連続受信できるデータ量をウインドウ・サイズとして連絡し合います。的確に連続転送することで転送時間を短くできます。こうした処理を「フロー制御」と呼びます。

TCP スタックは、受け取ったデータをいったん受信バッファに溜めて届け先となるアプリケーションに引き取ってもらいます。際限なくデータが転送されてくると、受信バッファからデータがあふれてしまいます。

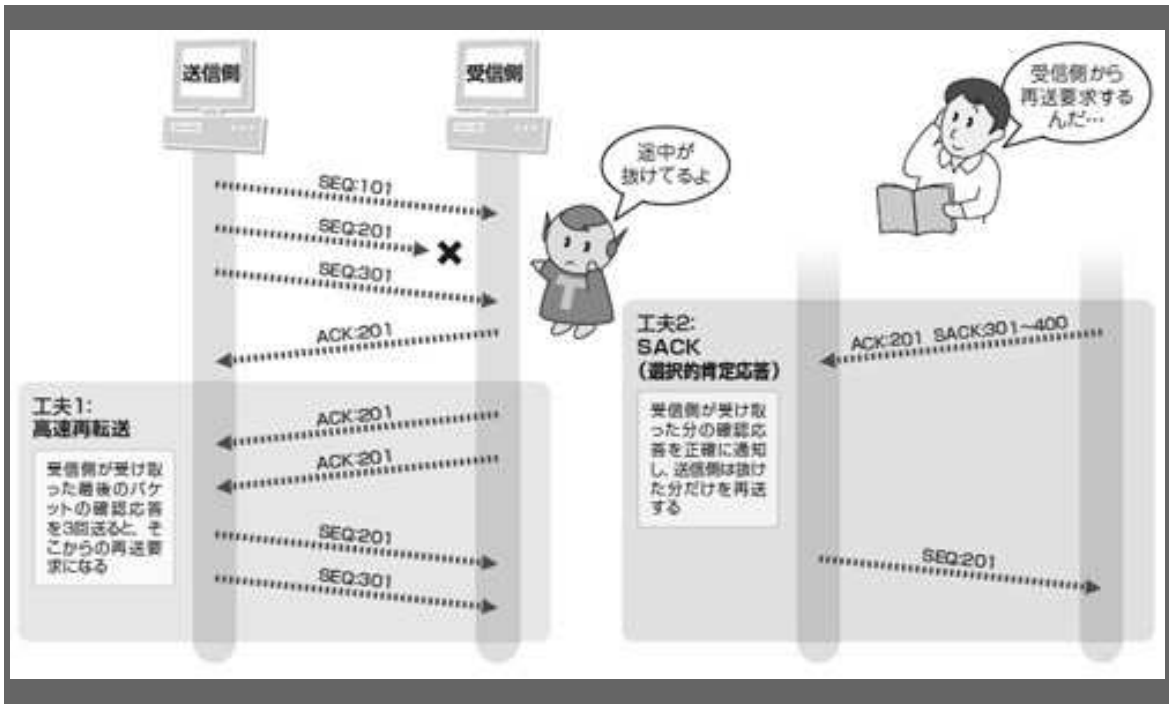
そこで TCP では、連続して受信できるデータ量を**「ウインドウ・サイズ」**というバイト単位の値でお互いに通知し合うことになっています。一挙に大量のデータが届いたり、送信先のアプリケーションが受信バッファからデータを引き取るのが遅れたりすると、受信バッファの空きが小さくなります。通知するウインドウ・サイズを空きに応じて変更することで、データを受信しきれなくなることを防いでいます。

〔2〕受信側から再送を要求する

効率を上げる工夫はほかにもあります。TCP の元々の標準として決められた方法ではありませんが、データが届かなかったときに受信側が再送を要求できるようになっているのもその一つです。

【1.2.3】で見たように TCP の再送の基本は、送信側が一定時間内に確認応答を受け取れなかったときにデータを送り直すというものです。ただ、これでは時間切れを待たないとデータを再送できません。データの抜けに気付いた受信側が積極的に再送を要求できれば、転送効率は向上します。

そこで登場したのが再送の二つの工夫です（図 4-2）。



（図 4-2：再送の効率を上げる二つの工夫があります）

時間切れを待ってすべてのデータを再送するのは効率が悪い。そこで TCP は、受信側から再送を要求したり、受信側が抜けたデータを送信元に通知したりするしくみを用意しています。

順番に見ていきましょう。まずは「高速再転送」からです（図 4-2 の工夫 1）。パケットの抜けに気付いた受信側が、抜けたデータの前のパケットの確認応答を、正規のものを含め合計 3 回以上連続して送ります。送信側が確認応答を連続して受信したら、再送要求だと判断して、ただちにパケットを再送します。

TCP スタックが高速再転送で再送を要求したときに、万一相手が高速再転送に対応していないと、相手はわからずに時間切れを待って再送します。効率は上がりませんが不具合も起こりません。

高速再転送は、TCP の機能強化の位置付けです。今ではほとんどの TCP スタックで使えます。

もう一つの SACK（サック）は、 途中のパケットが抜けている場合、受信側が送信側に抜けたパケットを通知する方法です（図 4-2 の工夫 2）。飛び飛びに受信したデータを通知することによって、抜けているデータがわかるようになっています。送信側はその内容を見て、抜けている部分だけを送り直します。SACK と高速再転送を併用すれば、受信側が抜けたデータだけ直ちに再送してもらえます。

SACK を使用するときは、TCP コネクションの確立時に双方の TCP スタックの間で SACK を使うかどうかをあらかじめ確認しておきます。受信したデータを正確に伝える情報は、TCP ヘッダーのオプション部分に書き込みます。

SACK はすべての TCP スタックで使えるわけではありません。ただし、Windows 2000 以降の Windows では使えるようになっています。

１．２．５．チャレンジコーナー

■問 1　以下は、TCP を説明した文章です。空欄［　a　］～［　d　］に当てはまる字句を選択肢から選びなさい。

TCP は、アプリケーション間に［　a　］通信路を提供するプロトコルです。IP ネットワーク上に確立した TCP ［　b　］上でデータを転送します。TCP で送るデータにはシーケンス番号を付けます。データを転送した相手からは確認応答番号が返ってきます。これらの番号を互いに確認することで、パケットが相手に届かなかったときの［　c　］制御を実現しています。また、ウインドウ・サイズを通知することで、転送ペースを調整する［　d　］制御を実現しています。

選択肢：

信頼できる	ベスト・エフォートの	コネクション	セッション
再送	異常通知	フロー	速度

■問 2　つぎの図は、TCP アプリケーションがパケットを送受信するときのやりとりを示した図です。空欄 ［　a　］に TCP パケットの種類を、空欄［　b　］にシーケンス番号の値を、空欄［　c　］、［　d　］に確認番号の値を答えなさい。ただし、TCP パケッ トのデータはすべて 100 バイトとします。

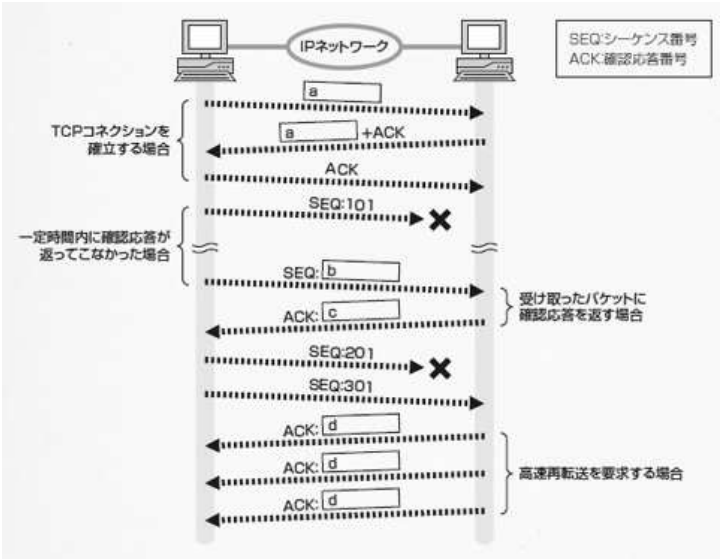


図:問 2

[a]　FIN　GET　RST SYN
[b, c, d]　001　101　201　301

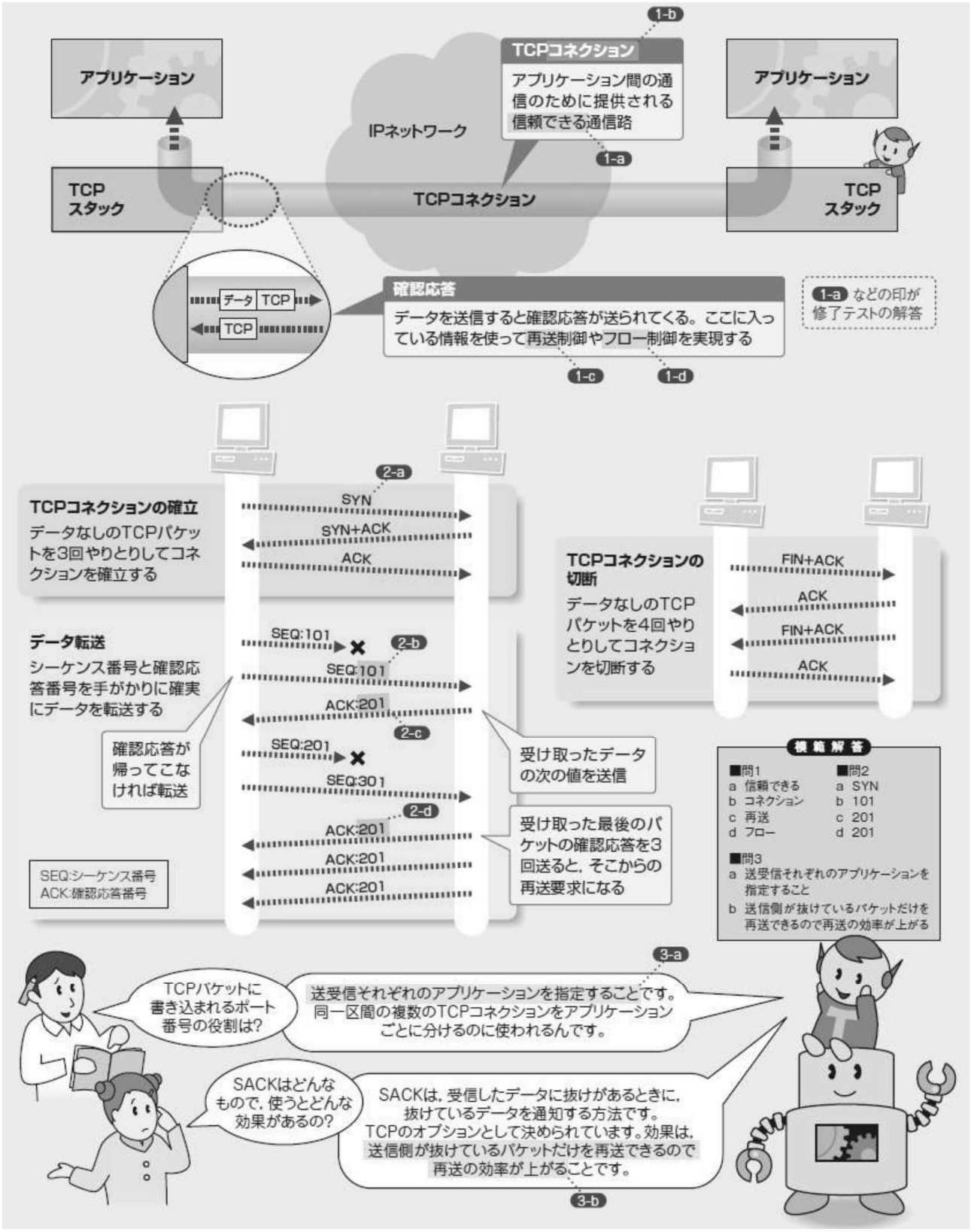
■問 3 [a] TCP パケットに書き込まれるポート番号の説明として正しいものを選びなさい。

①. 送受信それぞれの端末を指定する
②. 送受信それぞれのアプリケーションを指定する
③. 端末を接続する LAN スイッチのポートを指定する
④. ルーターが IP パケットを転送するときの転送先ポートを指定する

■問 3 [b] TCP で、SACK というオプションを使うことの効果として正しいものはどれか。

①. 受信側が確実にデータを受け取れたかを、送信側で確認できる
②. 通信相手のアプリケーションの処理速度に合わせてデータの送信速度を調整できる
③. ほかのアプリケーションに間違ってデータを送ってしまうことが避けられる
④. 送信側が抜けているパケットだけを再送できるので再送の効率が上がる

解答：



図●まとめ--TCP のしくみ

１章３節　UDP（User Datagram Protocol）

※UDP は、スマート家電の中心となるプロトコルです。

前節で学んだように、TCP は、転送の途中でなくなったパケットを送り直したり，通信量を調整したり、パケットの順番を整えるといった機能を備えています。アプリケーションは、TCP に処理を任せていれば、信頼性の高いデータ通信を実現できます。

その一方で，世の中には通信の信頼性を確保するよりも「とにかくデータを素早くやりとりできること」を重視するアプリケーションが存在します。また、「なるべくサーバーに負担をかけずに通信できること」を要求するアプリケーションもあります。こうしたアプリケーションには、TCP の利用が向いていなかったり、TCP が備える信頼性確保のしくみがかえって邪魔になったりすることさえあるのです。

そこで登場するのが UDP（User Datagram Protocol）です。UDP は、TCP の持つ信頼性確保のしくみを切り捨てる代わりに、TCP よりも素早く通信できたり、通信処理にかかる負荷を減らせたりできるようになっています。UDP を利用するアプリケーションは身のまわりでたくさん使われています。UDP は後の章で解説するスマート家電の中心となる ECHONET Lite 通信プロトコルでも利用されています。

１．３．１．ポート番号でアプリの識別

UDP は、IP ネットワークを介してアプリケーション同士をつないで、データをやりとりするためのプロトコルです。

IP ネットワークの役割は、IP パケットを通信相手のパソコンに届けることです。しかし、IP だけでは通信は成り立ちません。受信側のパソコンは、ただ IP パケットを受け取っただけでは、送られてきたデータをどのアプリケーションに渡せばいいのか判断できないからです。

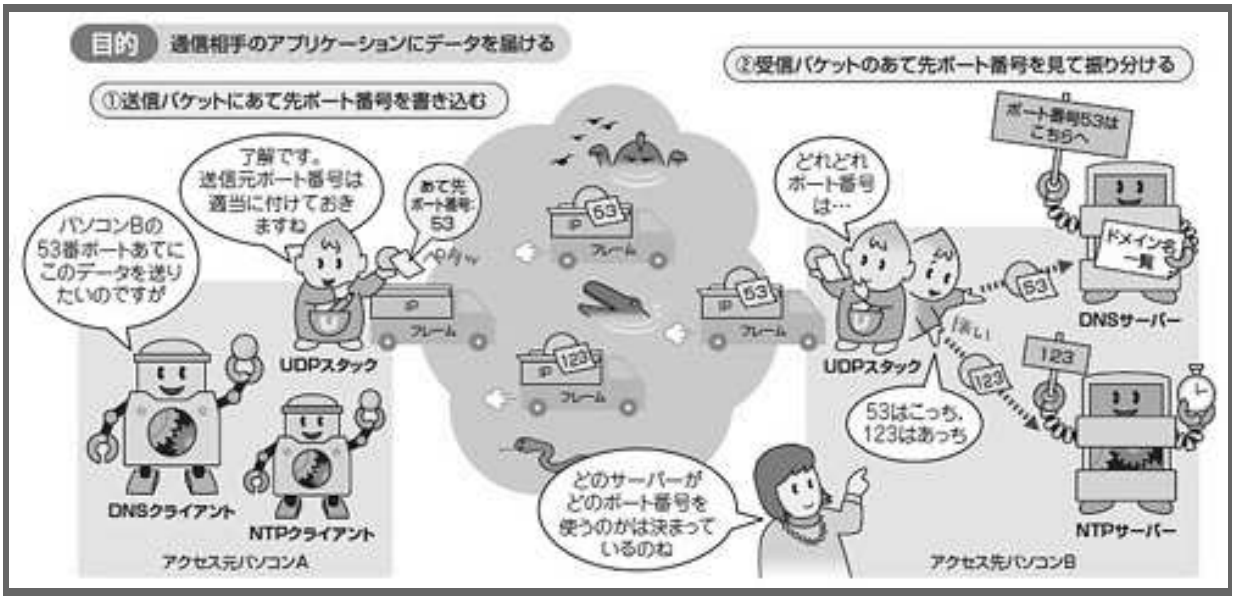
そこで UDP が登場します。

〔1〕アプリごとに固有の番号で識別

UDP は、IP パケットの中身のデータがどのアプリケーションからどのアプリケーションあてに送られたものかを伝える役割を持ちます。このとき使うのが「ポート番号」です。0～65535 の値をとります。アプリケーションごとに異なるポート番号を使うことで、通信相手のパソコンは受け取ったデータを適切なアプリケーションに届けられるようになります。複数のアプリケーションが同時に通信したときでも、データが、ごちゃ混ぜになってしまう事態を避けることができます。

それでは、アプリケーション同士が UDP を使って通信する様子を具体例で見てみ

ましょう（図 1-1）。



（図 1-1●UDP はアプリケーション間の通信を橋渡しするプロトコルです）

ネットワーク・アプリケーション同士が TCP/IP で通信するには、やりとりする IP パケットがどのアプリからどのアプリあてに送られているのかを識別する必要があります。UDP はその識別に使うポート番号を管理するために特化したプロトコルです。

アクセス元のパソコン A ではまず、アプリケーションが UDP を処理するソフトウエア・モジュール「UDP スタック」に対してデータ送信を依頼します。例えば、UDP を使う典型的なアプリケーションである DNS の場合なら、DNS クライアントが UDP スタックに「このデータを 53 番ポートあてに送ってください」と依頼します。

依頼を受けた UDP スタックは、受け取ったデータにあて先ポート番号などの情報を付けて UDP パケットを作り、IP ネットワークに送り出します。パソコン A の UDP スタックが行う処理は基本的にこれだけです。

一方、データを受信するパソコン B の UDP スタックが受け持つ仕事は、届いた UDP パケットに書かれているあて先ポート番号をチェックしてアプリケーションに振り分けることです。あて先ポート番号が 53 番なら、サーバーとして 53 番ポートでアクセスを待っている DNS サーバー・ソフトへデータを渡すという具合です。

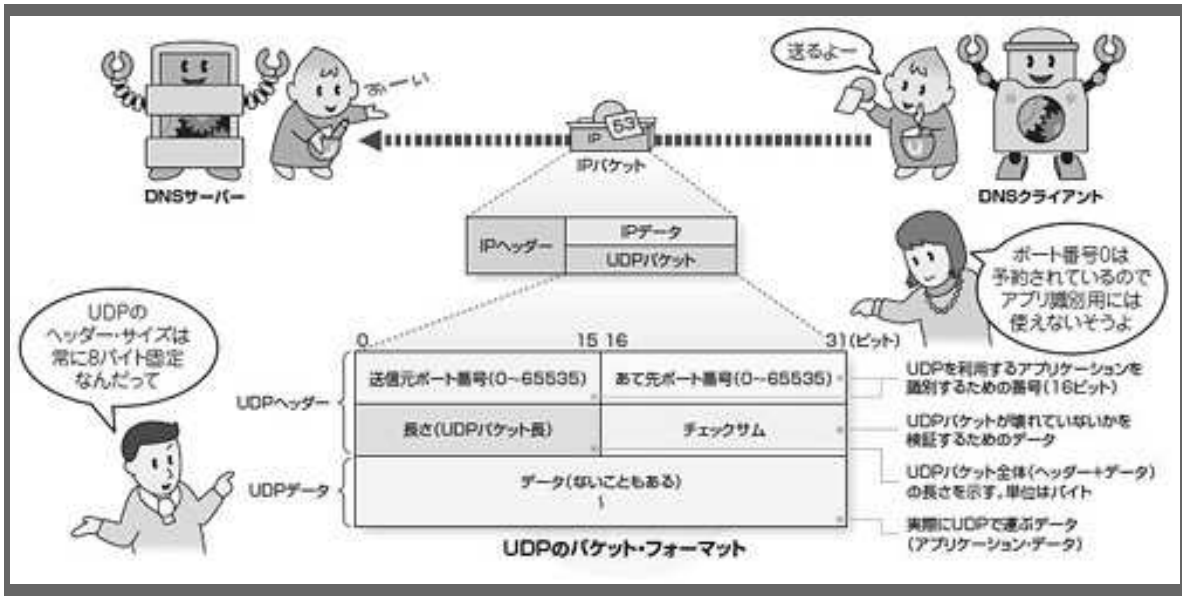
UDP の通信はこれで完了です。このやりとりを繰り返すことで、UDP を使うアプリケーションの通信は進んでいきます。

〔2〕たった 4 項目の情報がすべて

実は、ポート番号によるアプリケーションの対応付けは、UDP の専売特許ではありません。UDP と対になるプロトコルの TCP もまったく同じ役割を持っています。

では、「UDP ならでは」といえる機能はほかにあるのでしょうか。UDP を初めて学ぶ人にとっては少々驚きかもしれません、そのようなものは何もありません。ポート番号の管理以外は何もしないことが UDP の最大の特徴であって、これこそが UDP を使うメリットなのです（この意味については後で説明しています)。

ほかに何の機能もないことは、UDP のパケット・フォーマットを見れば明らかです（図 1-2)。



（図 1-2●ポート番号は UDP ヘッダーに書き込んでやりとりします）

UDP パケットは、IP パケットのデータ部分に入れて運ばれます。UDP パケットは UDP ヘッダーと UDP データで構成され、UDP ヘッダーにはアプリケーション識別用のポート番号など四つのフィールドがあります。

UDP パケットは、UDP ヘッダーと UDP データで構成します。UDP データには個別のアプリケーションが使うデータが入ります。UDP が果たす機能は UDP ヘッダーを使います。

といっても、UDP ヘッダーには、(1) 送信元ポート番号、(2) あて先ポート番号、(3) 長さ、(4) チェックサム　という四つのフィールドしかありません。長さは 8 バイト固定です。TCP ヘッダーが持つ通信の制御にかかわるフィールドは、UDP ヘッダーには一切ありません。あとから機能を追加できる余地すら残していません。データをアプリケーションに届けるために必要な最低限の情報以外はすべて削ってしまっているのです。

これほど割り切ったプロトコルである UDP を使うことにいったいどんなメリットがあるのでしょうか。これから見てみましょう。

１．３．２．接続手順なしでパケットを送受信する

これまで見てきたように、UDP の役割はポート番号を使ってアプリケーション同士を結びつけることだけです。通信の信頼性を高めるような処理は一切していません。

本来、通信というものはできるだけ信頼性を高めるのが理想のはずです。TCP を使えば簡単にその信頼性が手に入るのに、信頼性を省いた UDP をわざわざ使うアプリケーションがあるのはなぜでしょうか。その答えは UDP の「軽さ」にあります。

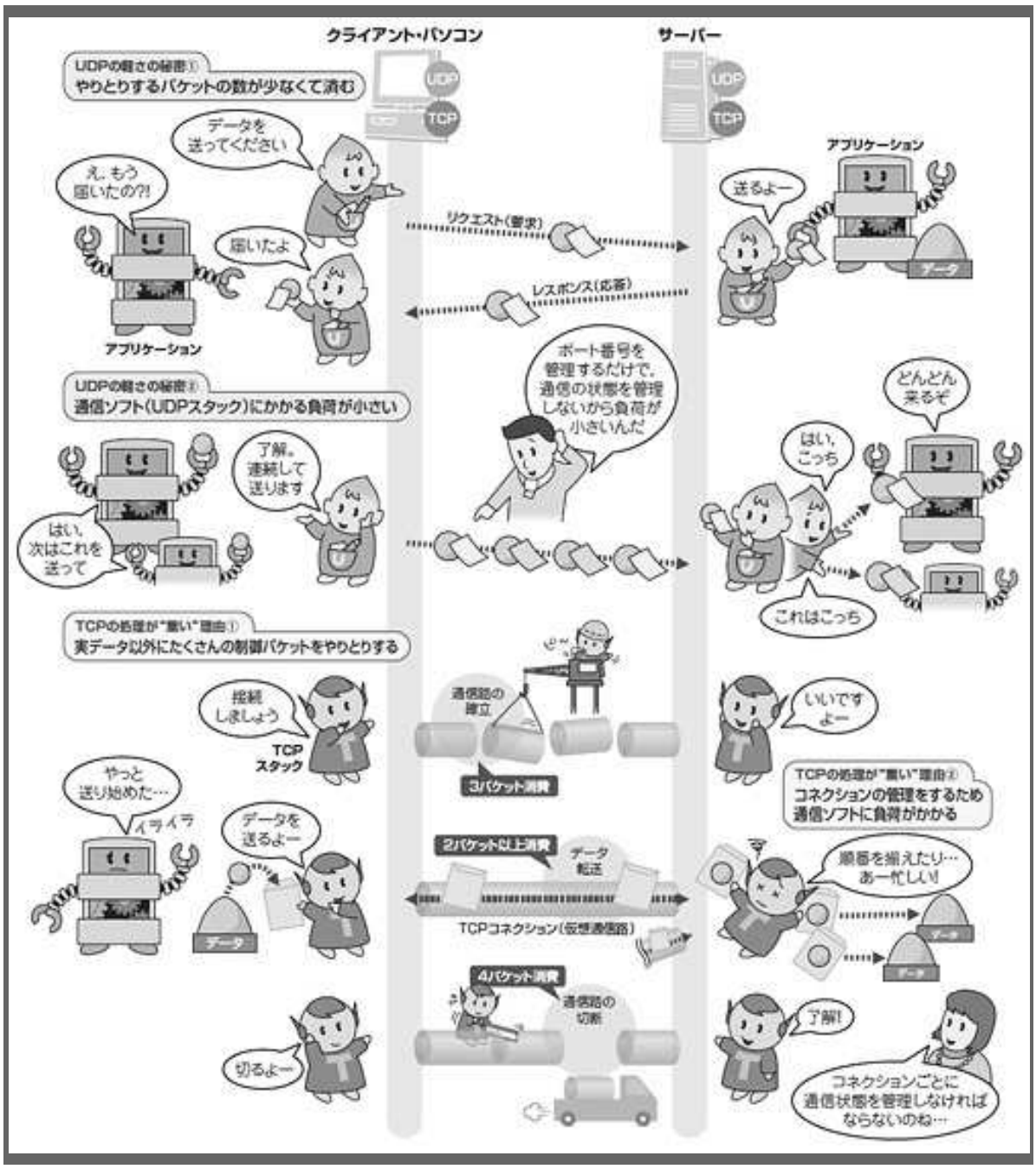
〔1〕通信路の確立や状態管理が不要

そもそも軽さとは相対的な言葉です。つまり、UDP が軽い理由は TCP が重い理由とセットで考えて初めて正しく理解できます。そこで、両者を比較しながら UDP の軽さの意義を探っていきましょう。

UDP が TCP よりも軽い理由は二つあります。それは、(1) やりとりするパケットの数が少なくて済むことと、(2) 通信ソフトにかかる負荷が小さいことです。順番に見てみましょう。

まずは、やりとりするパケットの数が少なくて済むことから。これは、UDP と TCP を使ったデータ通信の手順を比べれば一目瞭然です。

UDP を処理する UDP スタックは、アプリケーションからデータを受け取るとすぐに UDP パケットを作って相手に送ります。つまり、1 パケットに収まる小さなデータを相手に送り、相手からも同じ 1 パケットのデータを受け取るようなアプリケーションの場合、UDP なら無駄なく 2 パケットのやりとりで済んでしまいます（図 2-1 上の ①）。



（図 2-1：UDP を使う最大のメリットは“軽さ”にあります）

UDP の通信では、余計な手順を踏まずにいきなりデータを送れるので、小さなデータの受け渡しならわずか 2 パケット（1 往復）のやりとりで済んでしまいます。同じケースで TCP なら 9 パケット（4.5 往復分）が必要になります。また、UDP は通信状態の管理をせずパケットを送受信するごとに通信が完了します。このため通信ソフト（プロトコル・スタック）にかかる処理負荷は TCP よりずっと軽いのです。

一方、TCP では、相手と通信する際にまず通信路の確立が必要です。これだけで三つのパケットのやりとりが発生します（図 2-1 下の ①）。

続くデータのやりとりは UDP と変わらない（2 パケット）のですが、TCP の場合、さらに通信を終えるときにも切断手続きに 4 パケット消費します。つまり、たった 2

パケットのデータをやりとりするために、TCP を使うと 9 パケットも必要になるのです。パケットの無駄もさることながら、お互いの距離が離れていればいるほど通信を終えるまでにかかる時間も長くなります。

例えば DNS は、まさに上で示したように 1 パケットに収まる小さなデータが 1 往復するだけのやりとりで完了するのが基本となっています。このため、「DNS の通常の問い合わせに UDP ではなく TCP を使うなど、無駄が多すぎる」のです。

〔2〕サーバーの負荷はもっと差がつく

もう一つの軽さの理由である、通信ソフトにかかる負荷について UDP と TCP を比べてみましょう。

【1.3.1】で見たように、UDP スタックはポート番号を管理する以外の処理は何もしません（図 2-1 上の ②）。送信側の UDP スタックは、アプリケーションからデータを受け取って、ポート番号を書き込んだ UDP パケットを送り出したら仕事は終わりです。受信側の UDP スタックも、受け取ったパケットを、あて先ポート番号を基にアプリケーションに渡したらひと区切りです。受け取ったパケットをその都度処理するだけなので、それ以外のことにメモリーを消費しません。

一方の TCP は、通信相手との間で通信路（コネクション）を確立したり切断したりする「コネクションの管理」が必要になります（図 2-1 下の ②）。それぞれのコネクションごとにメモリーを確保してパケットの再送や並び替えといった制御を行います。このため、TCP スタックにかかる負荷はとても大きくなります。

裏返すと、UDP はこうした処理が必要ないので TCP よりもずっと軽くなります。とくにサーバーのように多数の相手と同時に通信するケースではその差は圧倒的です。

１．３．３．信頼性はアプリが自分で確保する

【1.3.1～1.3.2】で見たように、UDP はアプリケーション同士を橋渡しするポート番号だけを管理し、それ以外のしくみを省略することで「軽さ」を実現しています。

このため、ただ単純に UDP を使ってデータを運ぼうとすると、データはどんどん送り出せるけれど、相手に届く保証はまったくないという、信頼性がきわめて低い通信になってしまいます（図 3-1）。

（図 3-1●UDP そのものは通信の信頼性を高めるしくみを備えていません）

UDP の仕事はポート番号の管理だけで、紛失したパケットを再送するといった通信を制御するしくみは備えていません。パケットが途中でなくなろうが順番が入れ替わろうが、まったく関知せずやりとりを続けます。

送信側の UDP スタックは、UDP パケットを送ったら、送りっ放しであとは知らんふり。受信側の UDP スタックも、UDP パケットが届こうが届くまいがおかまいなしです。

こんな通信しかできなければ、いくら処理が軽いからといっても、UDP を使う意味はありません。

〔1〕アプリ側で処理をまかなう

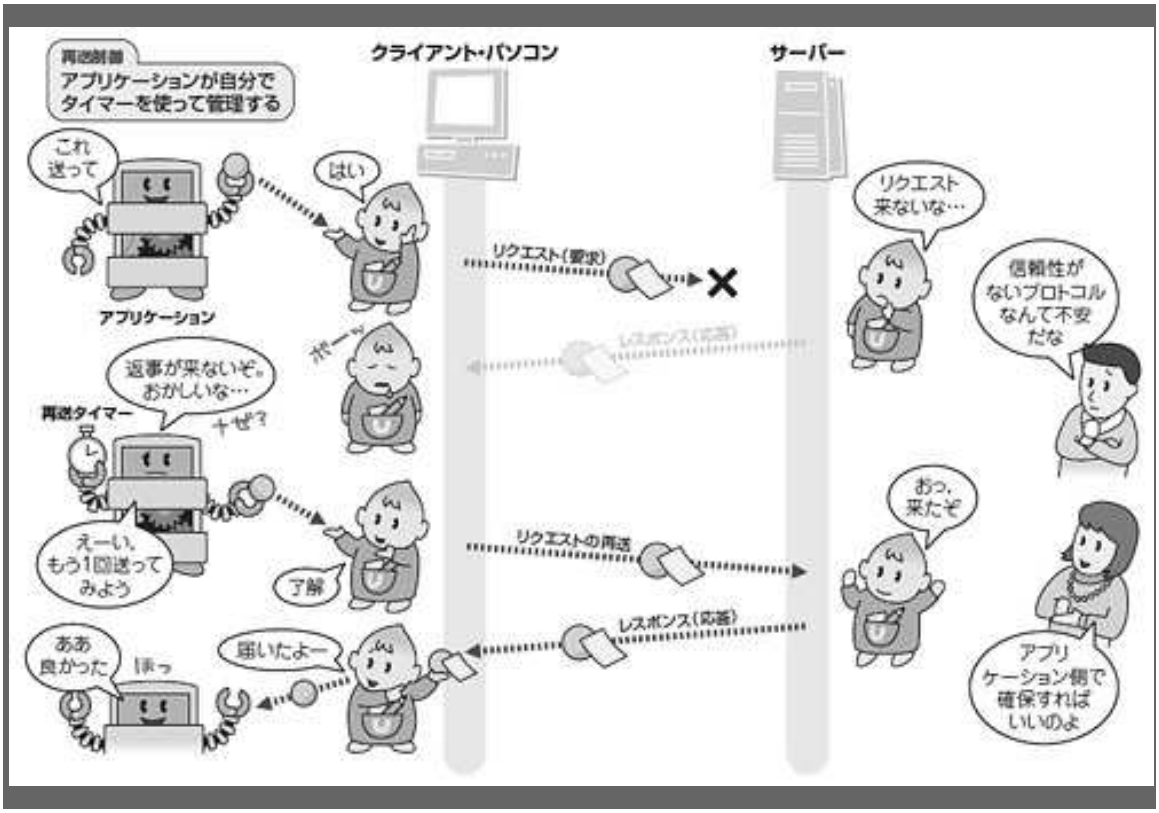
ではどうすればいいのでしょうか。まず考えられるのは、UDP 自体に何とか信頼性を持たせられないかということでしょう。しかし、【1.3.1】で見たようにそれは不可能です。UDP ヘッダーには通信の制御に使える空き領域は 1 ビットもありません。

UDP に頼れないなら、残された手は一つです。アプリケーションが自ら工夫して信頼性を確保するしかありません。パケットが途中でなくなったら再送し、データがあふれそうになったらパケットの流量を調整し、パケットの順番が入れ替わっても大丈夫なように番号を付けておく――。こうした制御をアプリケーションが主体となって行うことで、UDP の特性を生かしつつ信頼性のある通信を実現できます。

実際に、世の中で使われている UDP アプリケーションの多くは、こうした信頼性確保のための制御のしくみを持っています。ここでは、そうした制御のしくみのうち、最も基本的な「パケットの再送制御」について見てみましょう。

〔2〕アプリ自身がタイマーで監視

UDP を使うアプリケーションがパケットの再送制御を実現するには、「そもそも送信したパケットが相手に届かないことをどうやって判断するのか」ということがポイントとになります。一般に、その判断にはアプリケーション自身が管理する「再送タイマー」を使います（図 3-2）。



（図 3-2●通信の信頼性はアプリケーション側で高めます）

TCP を使うアプリケーションの場合、TCP に通信の信頼性確保のための制御をすべて任せておけます。一方、UDP はそうした制御を何ひとつしないので、上位アプリケーションが自ら信頼性を高めるための制御をするしかないのです。

UDP アプリケーションは、パケットの送信を UDP スタックに依頼したあとすぐにタイマーを使って時間を計り始めます。そして、もし一定時間内に相手からの応答が返ってこなかったらパケットがなくなったと判断して、同じデータを再送信します。同様に、相手側のアプリケーションでもタイマーを使って再送制御をすることで、通信経路上でたまたまパケットが紛失したことによる通信の失敗も避けられます。

ただし、相手から応答パケットが返ってこなかったからといって、必ずしもパケットが経路上で紛失しているとは限りません。相手のアプリケーションがパケットの受け取りを拒否したり、返事を返さずに捨ててしまったりしている可能性もあります。そんな状況でひたすら再送し続けても無駄なだけです。

そこで多くの UDP アプリケーションは、タイマーによる再送制御機能に加えて、

永遠に再送し続けないように処理する機能も併せ持っています。「3 回送ってダメだったらやめる」とか「再送信のたびに送信間隔を広げる」といった具合です。

具体例を挙げると、IP 電話の制御用プロトコルである SIP の場合、「相手を呼び出す INVITE（インバイト）リクエストは、相手から応答がなかったら最初は 0.5 秒、次は 1 秒という具合に送信間隔を 2 倍ずつ増やしていきながら最大 6 回までパケットを再送する」ようになっています。UDP を使っても、工夫次第で TCP 並みに信頼性を高められるのです。

1．3．4．UDP しかできない一斉同報やストリーミング

【1.3.3】では UDP でも通信の信頼性を確保できることを解説しました。「それなら、結局 TCP と UDP はどちらを使っても大差はないのでは？」と思われるかもしれません。それは半分当たっていますが、半分は不正解です。

確かに、世の中には TCP と UDP のどちらでも利用できるアプリケーションが存在します。例えば、【1.3.3】で登場した SIP がそうです。処理が軽いということで現在は専ら UDP が使われていますが、規格上では TCP も利用できます。

このように UDP と TCP の両方で使えるアプリケーションがある一方で、UDP でしか実現できないという、UDP を使うメリットが大きいアプリケーションも存在します。【1.3.4】では、そうした「UDP ならでは」の通信方法やアプリケーションについて見てみます。

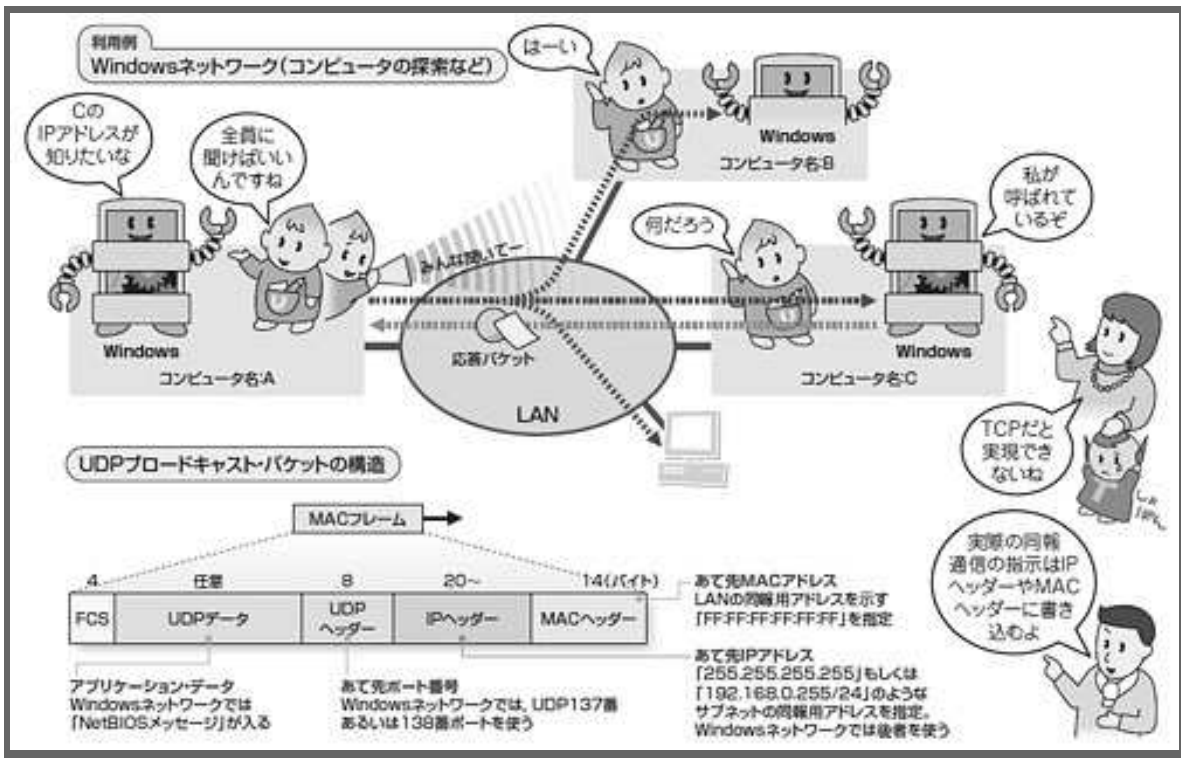
〔1〕TCP では実現できない同報通信

UDP なら簡単に実現できるのに、TCP では逆立ちしても実現できない通信方法があります。それは、ネットワーク上にいる全員にメッセージを届ける「ブロードキャスト」（同報通信）や特定グループに送る「マルチキャスト」です。

なぜ、UDP でないとブロードキャストやマルチキャストは実現できないのでしょうか。理由は簡単、【1.3.2】で見たように、TCP は通信相手との間で必ずコネクションを確立して通信します。しかし、一つのメッセージをネットワーク上の複数の相手に届けるのに、それらの相手と同時にコネクションを確立できないからです。

一方の UDP は、コネクションを確立せずに通信を行います。事前の準備なしでいつでも誰とでも同時にパケットをやりとりできます。このためブロードキャストやマルチキャストが実現できるのです。

UDP でブロードキャストを使う典型例が Windows ネットワークです。Windows ネットワークでは、コンピュータ名の一覧を表示したり、コンピュータ名から IP アドレスを調べたりする目的で UDP によるブロードキャストを使います（図 4-1）。



（図 4-1●同報通信（ブロードキャスト）ができるのは UDP だけです）

UDP は、通信相手と個別に接続状態を作らずにパケットを送信できて、複数の相手から同時にパケットを受け取れるので、TCP では不可能な同報通信を実現できます。

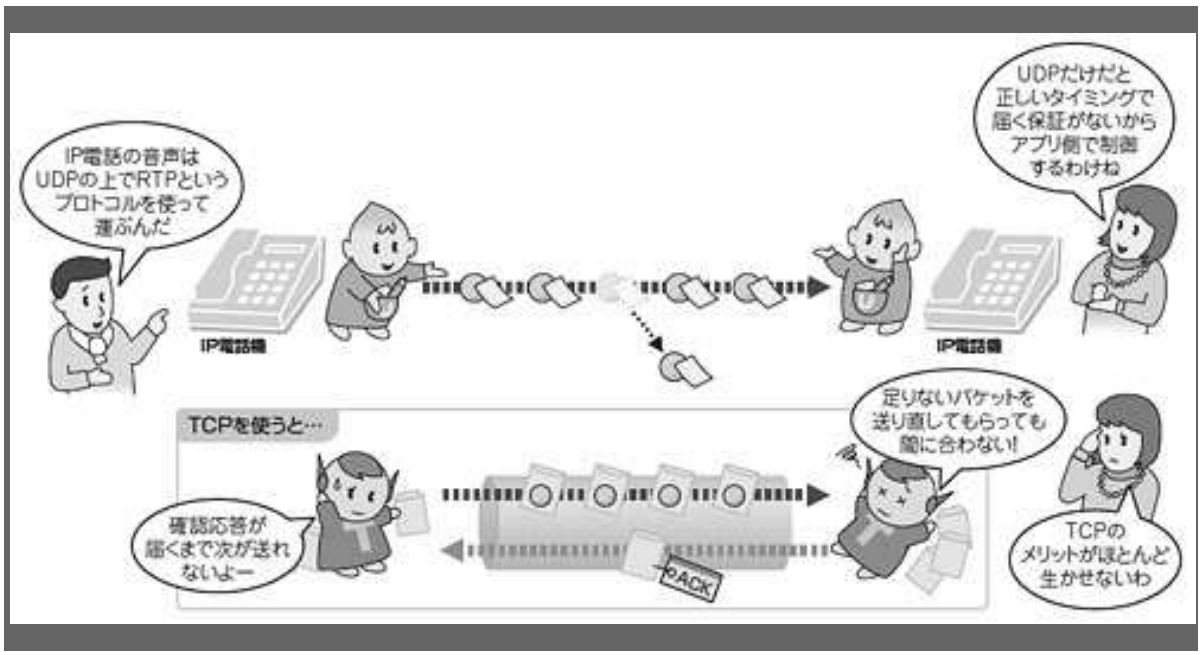
LAN 上で UDP を使うブロードキャスト・パケット（MAC（マック）フレーム）の中身も見ておきましょう。構造は、UDP パケットを IP パケットのデータ部に入れて、その IP パケットが MAC フレームのデータ部に入る形になります。

このなかで、あて先 MAC アドレスとあて先 IP アドレスで共に同報用アドレスが指定されています。一方、UDP ヘッダーや UDP データにはブロードキャストを示す情報は何もありません。UDP は「何もしない」ことでブロードキャストの実現を手助けしているわけです。

〔2〕リアルタイム通信も TCP は不適格

次に、UDP を使うことのメリットが大きいアプリケーションを見てみます。それは、IP 電話の音声転送や映像配信といったリアルタイム系やストリーミング系のアプリケーションです（図 4-2）。

※ECHONET Lite のセンサーノードなども、これに該当します。



（図 4-2●リアルタイム通信も UDP が適しています）

IP 電話の音声パケットのやりとりやライブ映像のストリーミング配信など、リアルタイム性が高いアプ リケーションは、(1)パケットを再送しても間に合わない、(2)確認応答なしに必要なタイミングで必要な量だけデータを送れる--といった理由から圧倒 的に UDP が使われています。

こうしたアプリケーションでは、コンスタントに決まった数のパケットを相手に送り続ける必要があります。こうした処理に TCP は向いていません。一定数のパケットを送り続けたいのに、TCP では勝手に再送制御やウインドウ制御が働き、送出速度が調整されたり、あとから届いても意味がないのにパケットを再送したりするからです。

それに対して UDP は、何の問題もなくパケットを送り続けられます。ただし、UDP 自体には通信の信頼性を確保するしくみや、順番が前後して届いたパケットの順序を正しく直すしくみ、さらには再生のタイミングを補正するといったしくみがありません。これらの処理は上位アプリケーション側で用意する必要があります。

１．３．５．チャレンジコーナー

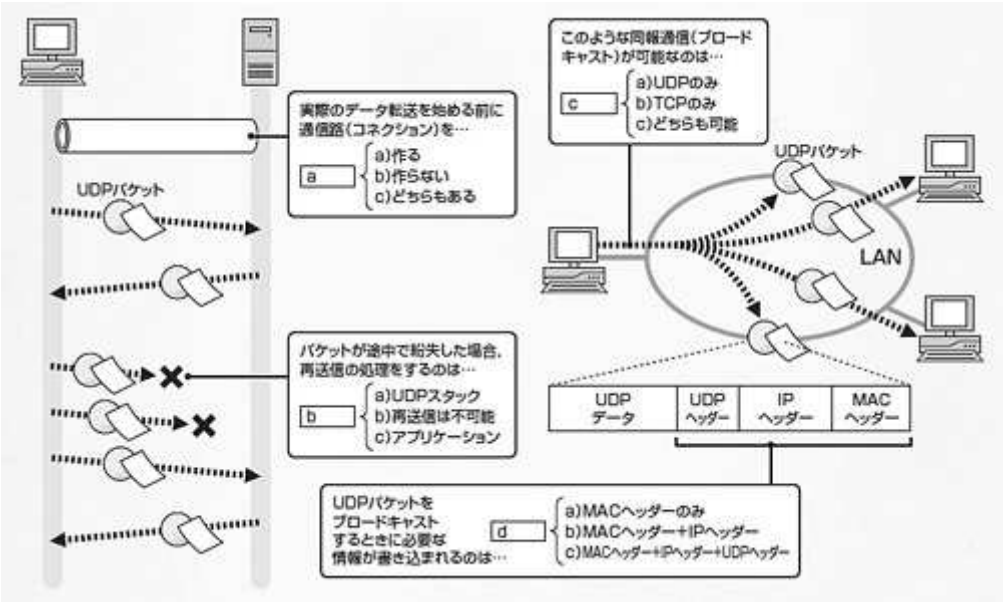
■問 1　以下は、UDP を説明した文章です。空欄［　a　］～［　d　］に当てはまる字句を選択肢から選びなさい。

UDP は、IP で通信する相手との間で［　a　］同士を結びつける役割を提供するプロトコルであす。そのために使う識別情報として［　b　］を利用します。［　b　］は、パケット中の［　c　］に書かれていて、0 から［　d　］までの値を設定できます。［　c　］には、［　b　］のほかに長さ（Length）とチェックサム（checksum）の二つの情報が書かれています。これらを加えた［　c　］の合計サイズは［　e　］バイトです。

選択肢：

パソコン	信頼できる	UDP ヘッダー	アプリケーション
UDP データ	シーケンス番号	ポート番号	オプション・フィールド
8	20	1023	65535

■問 2　次の図は、UDP を使ってアプリケーションが通信する様子を示したものです。図中にある［　a　］,［　b　］,［　c　］,［　d　］について、選択肢の中から正しいものを一つ選びなさい。



図：問 2

選択肢

［ a ］　作る　　作らない　　どちらでもない

［ b ］　UDP スタック　　再送信は不可能　　アプリケーション

［ c ］　UDP のみ　　TCP のみ　　どちらも可能

［ d ］　MAC ヘッダーのみ

MAC ヘッダー+IP ヘッダー

MAC ヘッダー+IP ヘッダー+UDP ヘッダー

■問 3 ［ a ］IP 電話で音声のやりとりに UDP が使われる理由として正しいものを選べ。

①. 音声を乗せたパケットが消えてしまったら再送する必要があるから

②. 音声を乗せたパケットを相手に途切れることなく届ける必要がある

③. 音声を乗せたパケットを送出する速度を調整する必要があるから

④. 音声を乗せたパケットが相手に届いたことを確認してから次のパケットを送る必要があるから

■問 3 ［ b ］UDP の通信処理が TCP よりも「軽い」と言われる理由として間違っているものはどれか。

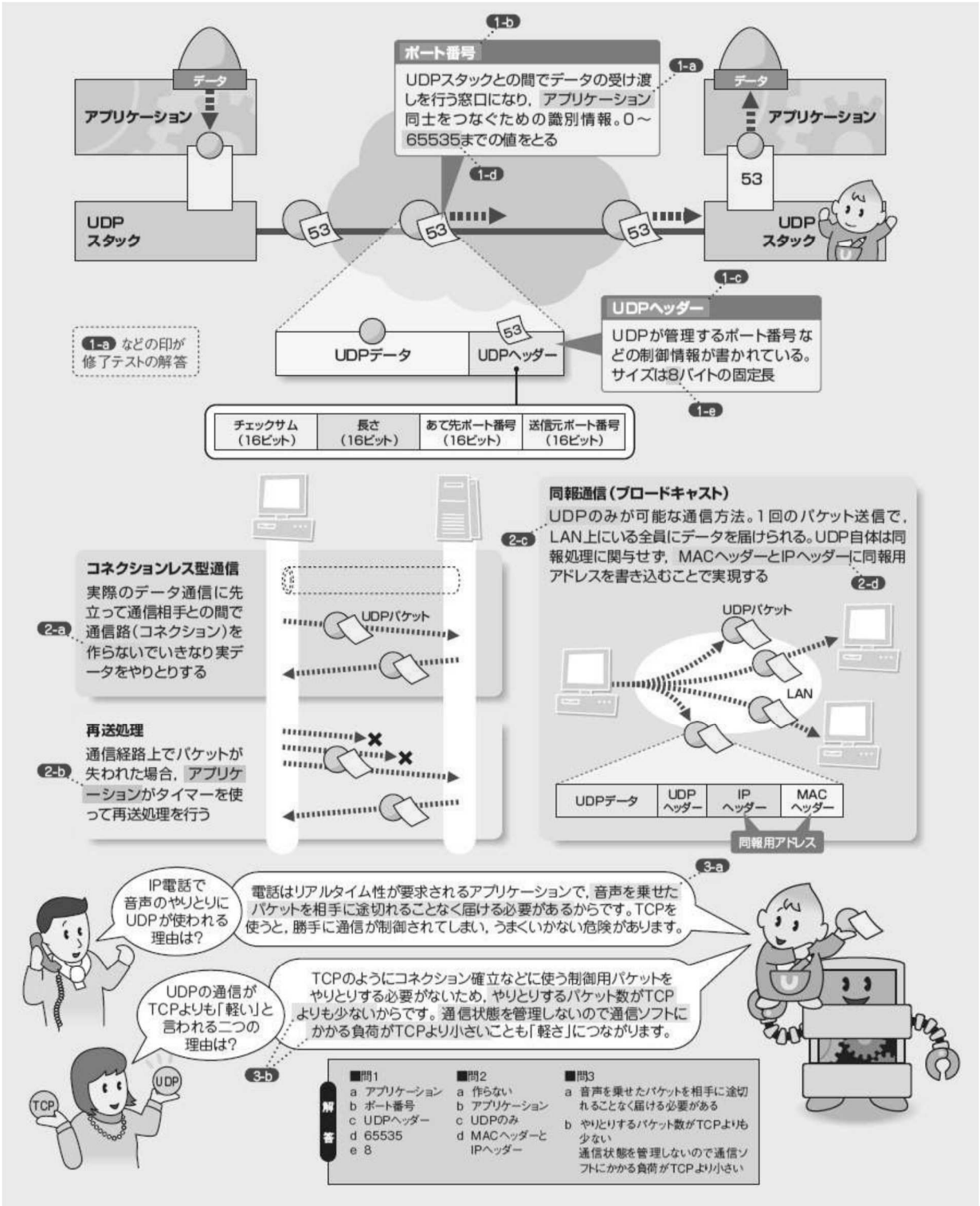
①. やりとりするパケット数が TCP よりも少ない

②. 通信状態を管理しないので通信ソフトにかかる負荷が TCP より小さい

③. UDP はコネクション確立などに使う制御用パケットをやりとりする必要がない

④. UDP はポート番号でアプリケーションを識別しない

解答：



図：まとめ--UDP のしくみ

１章４節　イーサネット・フレーム

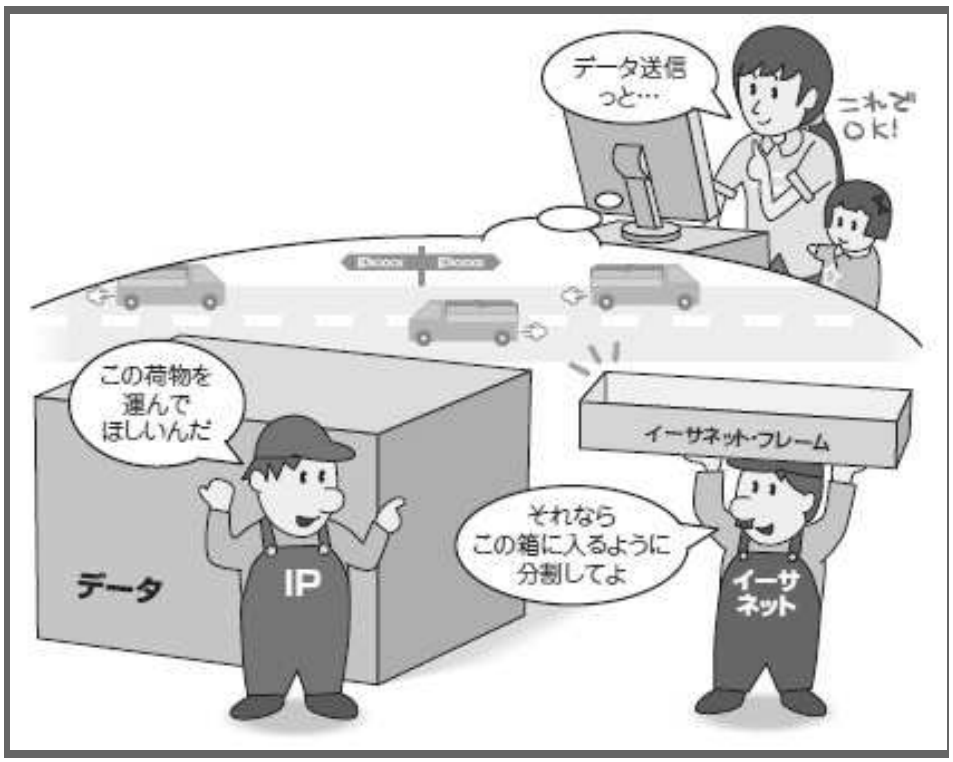
１．４．１．最大データ長 1500 バイトに合わせてデータ分割

【1.4.1】では、イーサネット・フレームとはそもそも何か、どのような役割を果たすのかを説明します。

〔1〕LAN でデータを運ぶ固まり

イーサネット・フレームは、イーサネットで機器が通信するときの基本単位です。それは具体的にどういうことなのでしょうか。

コンピュータや通信機器がやりとりするデータは「0」と「1」が並ぶビット列でできています。これをイーサネットで送るときは、パルスの信号に変換しなければいけません。このとき、データがある程度以上の量の場合は、決まった長さの"固まり"に切り分けます。この固まりがイーサネット・フレームです（図 1-1）。



（図 1-1：イーサネット・フレームはイーサネットでデータを送る基本単位で）

イーサネットでデータを送るときは、決まった長さの固まりに分割します。この固まりを「イーサネット・フレーム」といいます。

イーサネットにおける通信では、イーサネット・フレーム単位ですべての処理をします。イーサネット・フレームごとに「MAC アドレス」というあて先や送信元の情

報を付けて送信し、受信時にはデータが正しく届いているかどうかをチェックします。

これは宅配便などで小包を送るのと似ています。小包はトラックの荷箱に載せられて、配達されます。着いた小包のチェックも荷箱ごとに行います。小包がデータ、荷箱がイーサネット・フレームに置き換えられます。

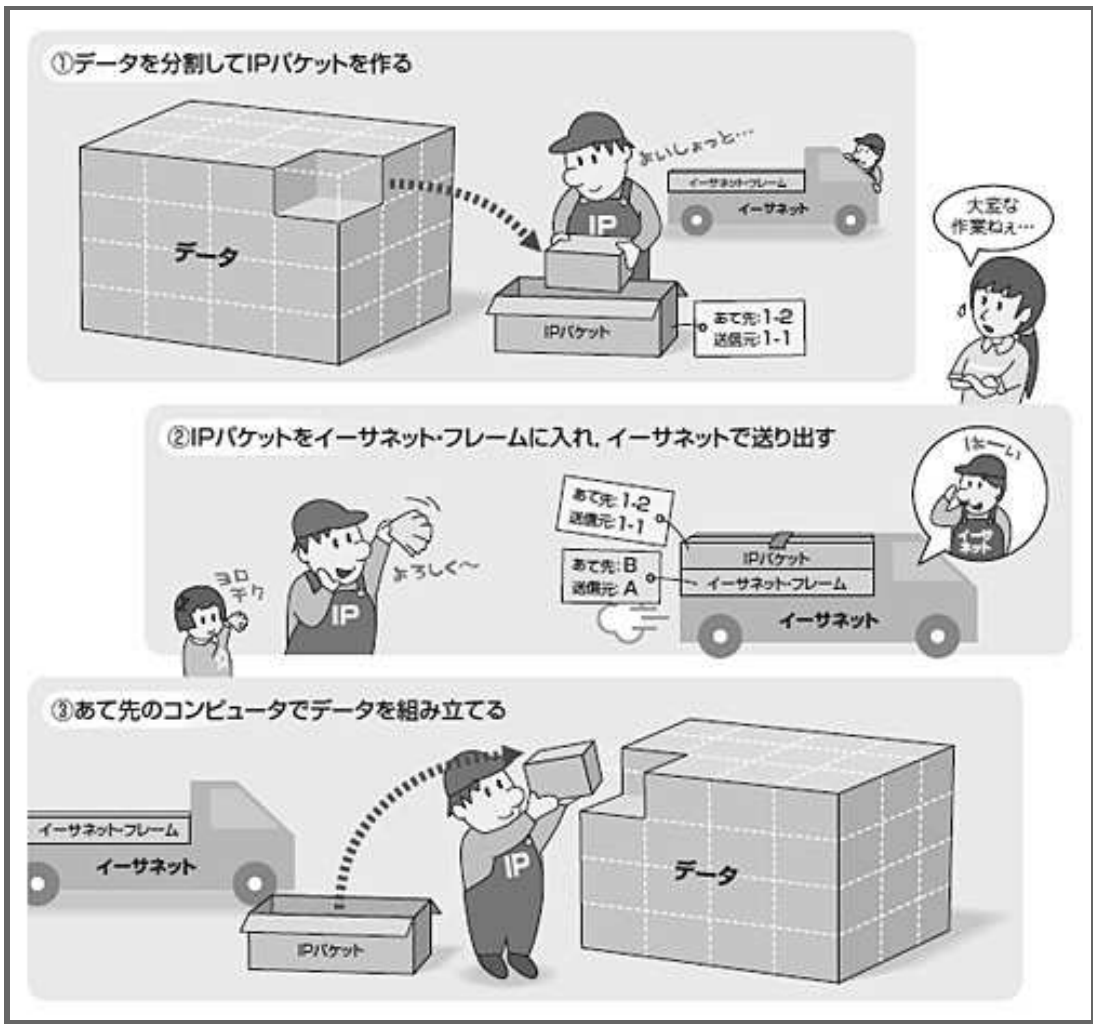
〔2〕データは区切った方が扱いやすい

なぜ送る際にいちいちデータを細かく区切るのでしょうか。理由は大きく二つあります。

一つは、特定の通信がイーサネットを占有するのを防ぐことです。イーサネットは1本のケーブルを複数台のコンピュータや機器で共有するのが基本的な考え方です。仮にあるコンピュータがデータを区切らずに送ってしまうと、そのデータが送り終わるまで、ほかの機器はデータをまったく送信できなくなってしまいます。

もう一つの理由は、1度に送れるデータのサイズが決まっていた方がコンピュータや通信機器が処理をしやすいからです。いろいろなサイズのデータが存在すると、まとめて処理したり、特定の部分だけを見て簡易的に処理したりするのが難しくなります。むしろ無駄なデータを入れてでもサイズをそろえたほうがコンピュータでは扱いやすいのです。小包を運ぶ場合に荷箱のサイズが決まっている方が荷物の積み込みや配達、チェックがしやすいのと同じです。

そのために、イーサネットではデータをイーサネット・フレームに収まる長さに分割していきます。最も一般的なIPの例で見てみましょう（図1-2）。まず、データをイーサネット・フレームに収まる長さのIPパケットに分割して、それをイーサネット・フレームに入れてネットワークに送り出しています。



（図 1-2●イーサネット・フレームの長さに合わせてデータを分割します）

IP はアプリケーションから受け取ったデータをイーサネット・フレームの長さに合わせて分割して、IP パケットを作ります。これをイーサネット・フレームに詰めて、ネットワークに送り出します。

送り出されたイーサネット・フレームがあて先コンピュータに届くと、送信するときとは逆の手順で開かれて、元のデータに組み上げられます。

〔3〕運べるデータは 1500 バイト以内

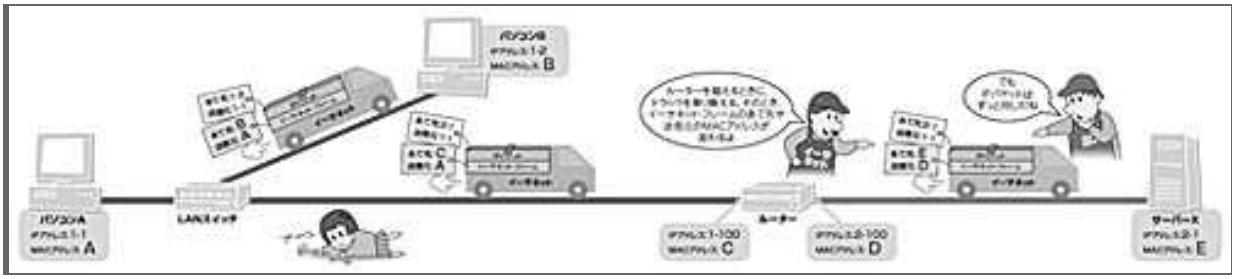
では、実際にイーサネット・フレームの長さはどれくらいなのでしょうか。イーサネット・フレーム全体の長さは 64～1518 バイトと決められています。これを「フレーム長」といいます。この中には、あて先や送信元を示す MAC アドレスなどの制御情報も含みます。そのため、一つのイーサネット・フレームで実際に運べるデータは 46～1500 バイトになります。

１．４．２．宛先 MAC アドレスで届け先指定

【1.4.2】では、送信元からあて先までイーサネット・フレームが届く流れを追ってみましょう。ここではイーサネット上の通信の大半を占める IP を例に説明します。

〔1〕IP パケットを乗せて出発

イーサネット・フレームが送信元からあて先に届く流れは、小包を配達する流れと似ています。そのため、図では小包を IP パケットに、トラックをイーサネットに、トラックの荷箱をイーサネット・フレームに例えました（図 2-1）。



（図 2-1●別の LAN にデータを送る場合は、いくつかのイーサネット・フレームを乗り継ぎます）

IP パケットには送信元 IP アドレスとあて先 IP アドレスが書いてあり、これはあて先に届くまで変わらない一方、イーサネット・フレームはルーターで中継されるときに作り換えられます。そのとき、送信元とあて先の MAC アドレスも変わります。

パソコン A からパソコン B にデータを送る場合を見てみましょう。パソコン A とパソコン B は LAN スイッチを介して同じ LAN 上にあります。

送信元のパソコン A では、パソコン内の TCP/IP ソフトが ARP（アープ）というプロトコルを使い、パソコン B の IP アドレスからパソコン B の MAC アドレスを調べます。MAC アドレスがわかると、データを分割して IP パケットに詰め、この IP パケットをイーサネット・フレームに入れます。

このとき IP パケットには､IP パケットの送信元であるパソコン A の IP アドレスと、あて先であるパソコン B の IP アドレスが付きます。一方のイーサ ネット・フレームには、フレームの送信元であるパソコン A の MAC アドレスと、あて先であるパソコン B の MAC アドレスが付きます。これは、送り状が IP アドレスで記してある小包を荷箱に入れて、その荷箱に MAC アドレスであて先と送信元を記した送り状を張り付けたイメージです。

こうしてできたイーサネット・フレームを LAN カードに渡すと、LAN カードが電気信号や光信号に変換して LAN に送り出します。

〔2〕自分に関係するものだけ処理

パソコンAが送出した信号はLANスイッチを介してあて先であるパソコンBに確実に届きます。パソコンBのLANカードは受け取ったイーサネット・フ レームのあて先MACアドレスを見て自分あてとわかると、このフレームからIPパケットを取り出してTCP/IPソフトに渡します。TCP/IPソフトは、IPパケットのあて先を見て、自分あてと判断すると、IPパケットからデータを取り出して元の通り組み立てます。これがイーサネット・フレームを使ったイーサネットの通信です。

なお、リピータ・ハブを使って複数の機器を接続している場合は、パソコン A から送り出されたイーサネット・フレームがLAN上の関係ない機器にも届 いてしまうことがあります。その場合、パソコンB以外の機器のLANカードはあて先MACアドレスが自分に関係ないことを確認すると、そのイーサネット・フ レームを処理せずに廃棄するようになっています。これは、1本の銅線を共有して通信していた初期のころから変わらないイーサネットの基本動作です。

〔3〕ルーターでフレームを作り直す

LAN でイーサネット・フレームが届く流れを説明しましたが、実際のネットワークはLAN内で完結するとは限りません。ルーターを介して外部のLANとデータをやりとりすることもあります。イーサネット・フレームは一つの LAN 内にしか届かないので、そのままでは通信できません。

その場合、ルーターが信号を受信して中継します。ルーターはフレームが自分あてとわかると中の IP パケットのあて先を見ます。あて先が自分ではないとわかると、その機器にIPパケットを届けるには次にどのIPアドレスに転送すればいいのかを自分が持つ経路表で確認します。そのIPアドレスを基にARPでMACアドレスを調べます。そして、そのMACアドレスあてのイーサネット・フレームにIPパケットを入れて送り出すのです。

例えば、パソコンAから別のLANに属するサーバーXにデータを送信するとしましょう（図 2-1 参照）。

パソコン A はあて先がサーバーX の IP アドレス、送信元がパソコン A の IP アドレスの IP パケットをイーサネット・フレームに乗せて送り出します。このフレームの送信元はパソコンAのMACアドレス「A」、あて先はルーターのMACアドレス「C」です。

ルーターはイーサネット・フレームを受信すると、IPパケットのあて先を確認します。サーバーX あてとわかると、サーバーX 向けにイーサネット・フレー ムを作り

直して送り出します。このとき、送信元はルーターの MAC アドレス「D」に、あて先はサーバーX の MAC アドレス「E」に変わります。

サーバーX がイーサネット・フレームを受信すると、サーバーX の LAN カードはフレームのあて先を確認します。自分あてとわかると、中から IP パケットを取り出して TCP/IP ソフトに渡します。

〔4〕作り替えるのはフレームだけ

このように、イーサネット・フレームはルーターで次の行き先向けに作り直されます。これは小包を遠方の届け先に届けるときに似ています。その場合も、小包は長距離トラックや行き先地域別のトラックを乗り継いで相手先に届きます。

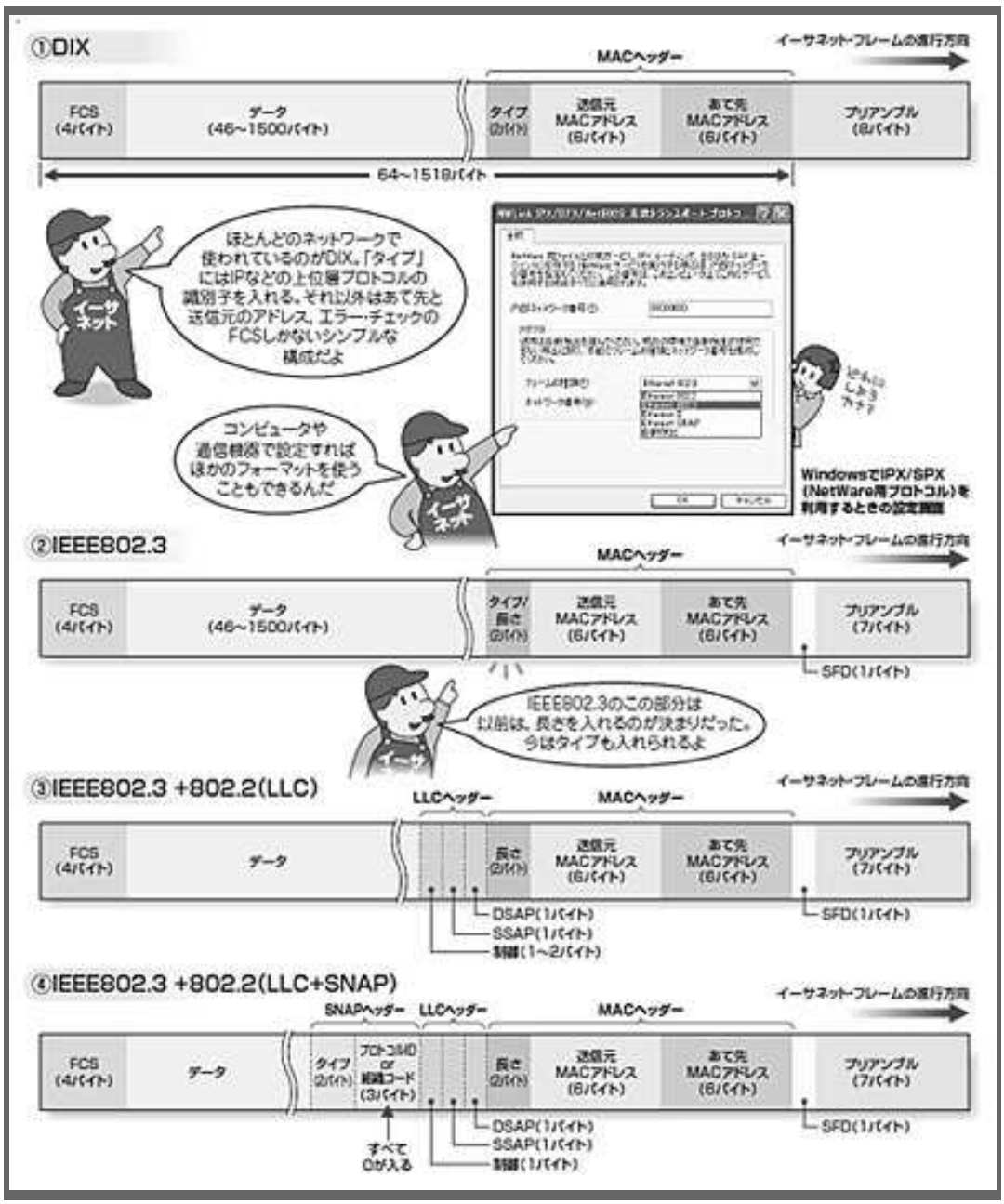
ルーターを越える場合、作り替えるのはイーサネット・フレームだけです。IP パケットのあて先と送信元は最終的なあて先のコンピュータに届くまで変わりません。

1．4．3．シンプルなフレーム構成

【1.4.3】は、イーサネット・フレームそのものに目を向けましょう。

実はイーサネット・フレームには「DIX（ディックス）」と「IEEE（アイトリプルイー）802.3」、IEEE802.3 から派生した「IEEE802.3＋802.2（LLC）」（LLC）、「IEEE802.3＋802.2（LLC＋SNAP（スナップ））」（SNAP）の四つの規格があります。

どの規格を使うかは実装次第で、プロトコルによって選択できる場合もあります。ただ、「今のネットワークは DIX を使う TCP/IP が主流で、LAN 上のイーサネット・フレームの約 9 割は DIX になっています」。ここでは DIX を中心に説明します（図 3）。



（図 3-1：イーサネット・フレームの構成）

イーサネット・フレームは、「プリアンブル」「MAC ヘッダー」「データ」「FCS」の四つの部分に大きく分かれています。全体の長さは、プリアンブルを除き 64～1518 バイトと決まっています。イーサネット・フレームの規格は 4 種類あり、細部の構成が違います。

〔1〕四つの部分で成るシンプルな構成

イーサネット・フレームは、「プリアンブル」「MAC ヘッダー」「データ」「FCS」の四つの部分からできています。

プリアンブルはフレームの始まりを表す合図のようなものです。中には「10101010101010101…10101011」という特殊なパターンの信号が並んでいます。LAN

上の機器はこのパターンを検知してフレームの送信開始を認識し、データを受信するタイミングを取ります。

プリアンブルの次が「MAC ヘッダー」と呼ぶ部分です。「あて先 MAC アドレス」「送信元 MAC アドレス」「タイプ」の三つのフィールドからできています。LAN 機器はプリアンブルの最後の「10101011」を検出すると、次のビットから 6 バイトをあて先 MAC アドレス、その次の 6 バイトを送信元 MAC アドレス、さらに次の 2 バイトをタイプと機械的に判断します。タイプには、データ部分に入っているデータのプロトコル（例えば、IP や NetWare 用プロトコルである IPX など）を示す識別子が入ります。

データ部分の次に付いているのが FCS です。FCS には、MAC ヘッダーやデータ部分に誤りがないかを検査するための値が含まれています。この値は送信元がフレームを作るときに計算して追加するもので、受信側がフレームの受信時に同じ計算をして値を照合する。値が合わなければ誤りがあると判断して、そのフレームを丸ごと廃棄します。

〔2〕長さとタイプは数値で判別

DIX 以外の規格も見てみましょう。基本構成はほぼ変わりません。違いは、送信元 MAC アドレスの次のフィールドがデータの「タイプ」ではなく「長さ」を示すことがある点です。

IEEE802.3 の策定時はこのフィールドは長さを示すことになっていました。ただ、実際のネットワークではタイプを示す DIX のイーサネット・フレームが主流だったため、DIX のタイプも IEEE802.3 のフレームとして扱えるように規格を変更しました。

示しているのが長さかタイプかはフィールド内の値で見分けます。

値が 1500 以下の場合は長さを示します。データ部分に入るデータは 1500 バイト以下なので、長さは 1500 を超えないからです。値が 1536 以上の場合はタイプを表します。例えば、IP は 10 進表記で 2048、IPX は 33079 です。なお、1501～1535 は値として規定されていません。

LLC と SNAP は IEEE802.3 を拡張した規格です。LLC は信頼性のある通信を実行するために作られた 802.3 の上位プロトコル（IEEE802.2 で規定）です。LLC ヘッダーは、通信内容を内部で識別するためのアドレス情報「DSAP」/「SSAP」と，制御情報からなる。一方の SNAP は、DIX や 802.3 で使ってきたプロトコル識別子を LLC 利用時に使うためのフレーム・フォーマットです。SNAP ヘッダーの中にある「タイプ」フィールドの使い方は、DIX や 802.3 の「タイプ」と同じです。

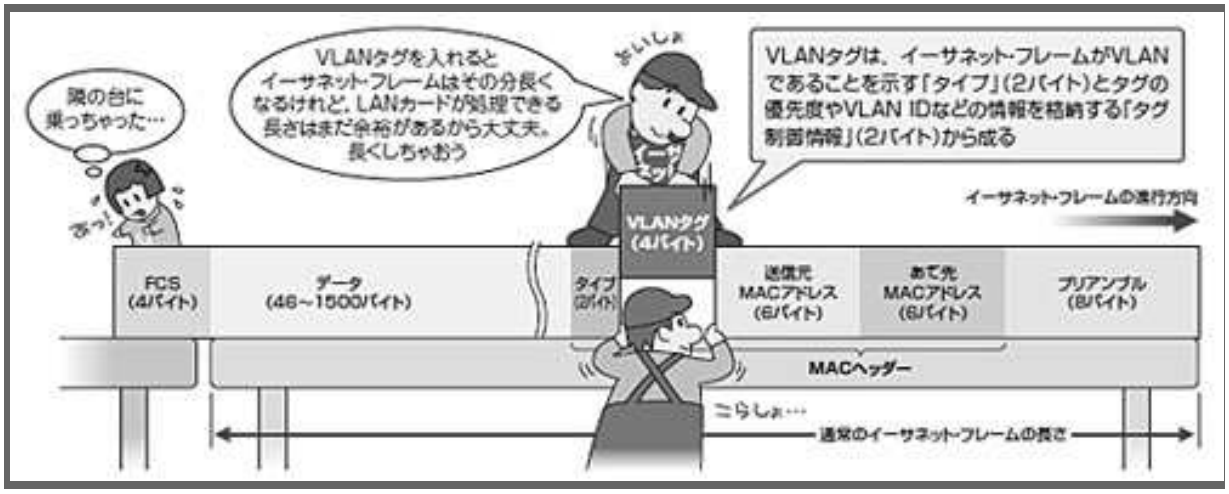
1．4．4．VLANとジャンボ・フレーム、フレームの拡張

ここまでイーサネット・フレームのフレーム長は最長 1518 バイトに決まっていると繰り返し述べてきました。しかし、実は、フレーム長を拡張している技術があります。VLAN やジャンボ・フレームです。【1.4.4】は応用編として、これらの技術について説明します。

〔1〕４バイトの VLAN タグを間に挿入

VLAN とは、ケーブルや機器の物理的な構成とは異なる構成の LAN を仮想的に実現する技術です。代表的な実現技術としてポート VLAN とタグ VLAN という二つの方法がありますが、このうちイーサネット・フレームを拡張して実現するのがタグ VLAN です。タグ VLAN は IEEE802.1Q と IEEE802.3ac という規格で策定されたものです。

タグ VLAN では、イーサネット・フレームの「送信元 MAC アドレス」フィールドと「タイプ」フィールドの間に仮想的な LAN を識別する 4 バイトの VLAN タグを挿入します（図 4-1）。VLAN タグは、そのフレームが VLAN であることを示す「タイプ」と、VLAN タグ・フレームの優先度や VLAN ID を示す「タグ制御情報」から成っています。



（図 4-1●VLAN はイーサネット・フレームにタグ情報を追加します）

VLAN のイーサネット・フレームは、送信元 MAC アドレスとタイプの間に「VLAN タグ」を挿入します。

LAN カードや LAN スイッチは VLAN タグの中のタイプ・フィールドを見て、10 進表記で「33024」という VLAN を示す値が入っていればこのフレームが VLAN タグ・フレームと判断します。そして、次に続く 2 バイトが VLAN タグのタグ制御情報、その後に続くのが通常のタイプ以降のフィールドと認識します。タグ制御情報には、VLAN タグ・フレームの優先度や属する VLAN の情報が入っています。

〔2〕1522 バイトになっても大丈夫

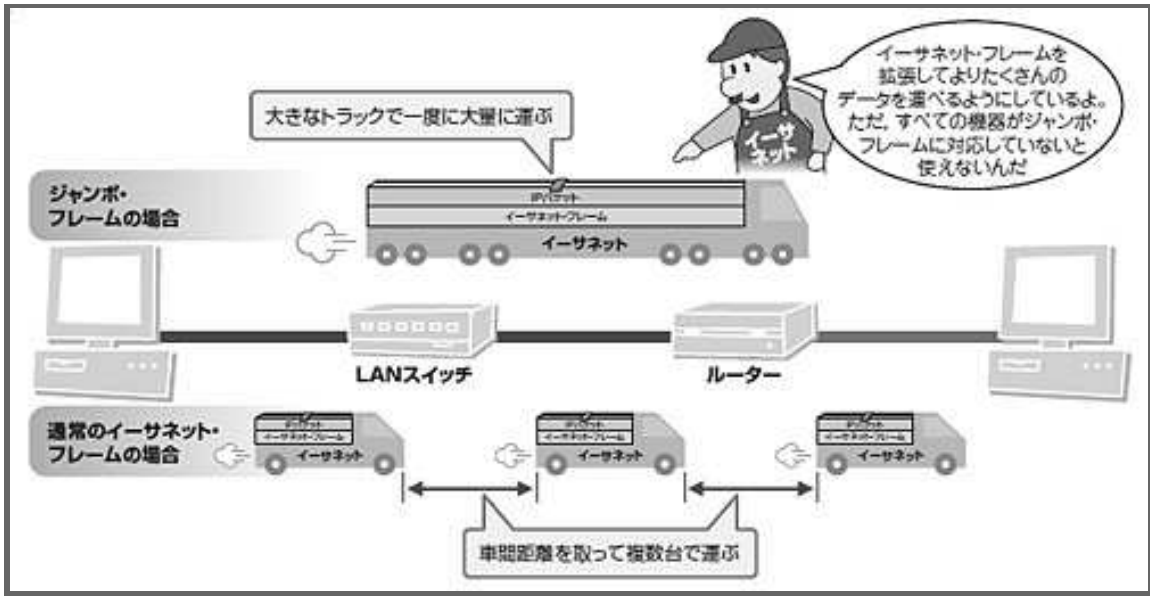
ここで「イーサネット・フレームが本来の 1518 バイトより大きくなっても大丈夫なのか」と疑問に思う人がいるかもしれません。確かに、VLAN タグを挿入すると、フレーム長は 1522 バイトになってしまいます。

ただ、コンピュータや通信機器を VLAN に対応させるのはさほど難しくありませんでした。イーサネット・フレームのフレーム長は規格で 1518 バイトまでと決まっていますが、実はコンピュータや通信機器の LAN カードの多くは、効率を考えた結果、フレーム長の最大値が 1536 バイトまたは 2048 バイトまでなら処理できるように作られていたのです。そのために、ドライバを VLAN に対応させて、VLAN タグを付加したり判別したりできるようにすれば問題ありませんでした。

〔3〕データの長さを６倍に拡張

これに比べると、ジャンボ・フレームへの拡張は難しいのです。

ジャンボ・フレームはデータ部分に入るデータ量を増やして、転送効率を上げる技術です（図 4-2）。



（図 4-2●イーサネット・フレームを拡張して、たくさんのデータを運べるようにしたのがジャンボ・フレームです）

ジャンボ・フレームはイーサネット・フレームのデータ部分に入るデータ量を 1500 バイトよりも大きくすることで、データの転送効率を上げる技術です。データ部分に入れるデータ量はベンダーによって異なりますが、一般的には 8000～1 万 6000 バイト程度です。

もともとイーサネット・フレームのフレーム長が 1518 バイト以内に決められたのは、あまり大きいフレームを送ると転送完了まで時間がかかり、その間 LAN を占有してしまう恐れがあるからでした。しかし、LAN が高速化した現在は、フレームを小分けにして送ると、送信間隔や個別のフレームの処理時間が増えるために、かえって効率が悪くなってしまいます。

そこで、フレームを拡張してより多くのデータを格納できるようにしたのがジャンボ・フレームです。ジャンボ・フレームのフレーム長はベンダーや機器によって異なりますが、8000〜1 万 6000 バイト程度に設定されています。ヘッダーなどの部分は変わらないので、通常のイーサネット・フレームの 5〜10 倍のデータを一度に送れることになります。

ただ、ここまでフレームが大きくなると、通常の LAN カードでは処理できません。そのため、ジャンボ・フレームは対応した機器間での利用に限られます。通信経路の途中に 1 台でも対応していない機器があったら通信できません。

また、ジャンボ・フレームは VLAN と違って規格が標準化されていないので、実装がベンダーによって異なります。処理できるイーサネット・フレームの最大長もまちまちです。送信側よりも受信側の機器の方が処理できる最大のフレーム長が小さいと、送信するフレームの大きさを受信側に合わせるので転送効率が上がらない場合があります。

１．４．５．チャレンジコーナー

■問 1　以下は、イーサネット・フレームを説明した文章です。空欄［　a　］～［　d　］にあてはまる字句を選択肢から選びなさい。

イーサネット・フレームは、コンピュータ同士がイーサネットでやりとりするデータの単位です。イーサネット・フレームは、あて先と送信元の［　a　］などを格納する「MAC ヘッダー」、送信するデータを格納する「データ」、誤り検出用のビット列の「［　b　］」の三つの部分からできています。データ部分に格納できるデータは、通常［　c　］～［　d　］バイトまでと決まっているので、コンピュータがデータを送信する場合、このサイズに合わせてデータを区切ります。

選択肢：

インターネット　　FCS　　MAC アドレス　　1500　　64　　LAN
IP アドレス　　46　　FSC　　1518　　IP ネットワーク

■問 2　以下は､パソコンからサーバーに IP パケットが運ばれるプロセスを記した図です。空欄［　a　］～［　h　］にあてはまる IP アドレスやイーサネット・フレームを選択肢から選び、図を完成させなさい。

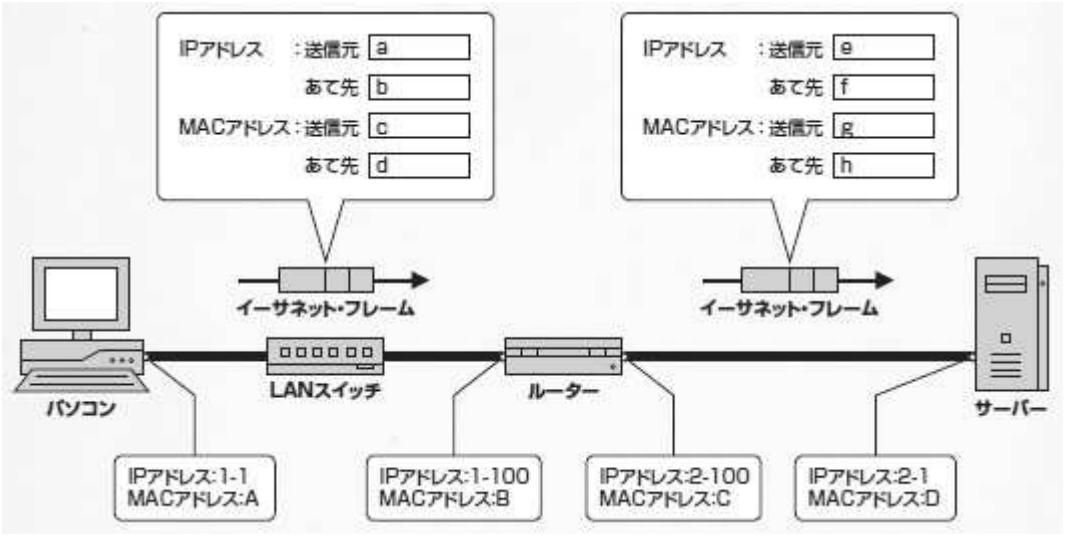


図:問 2

選択肢：

1-1　　1-100　　2-100　　2-1　　A　　B　　C　　D　　再送
1-2　　異常通知　　フロー　　速度

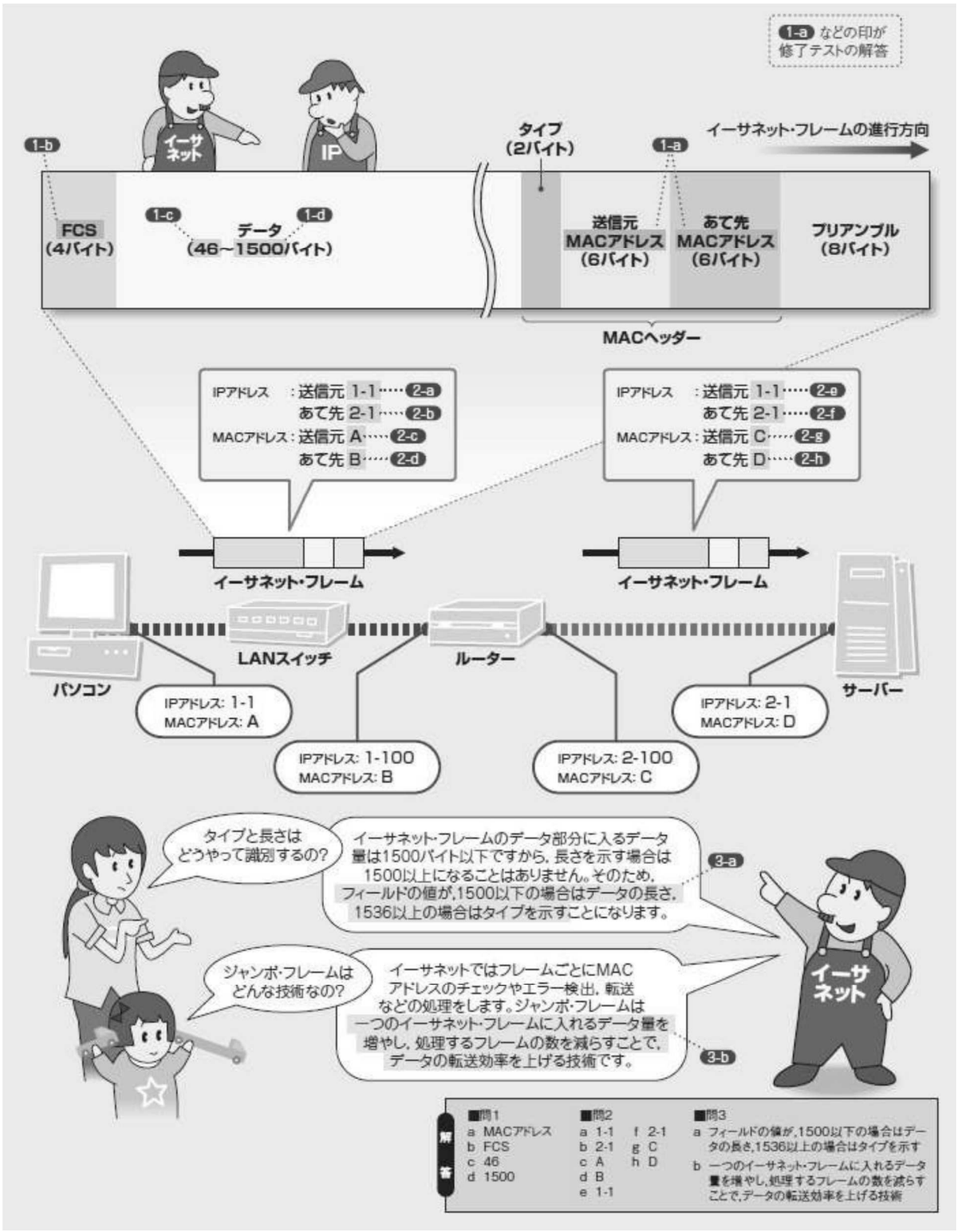
■問 3　[a]イーサネット・フレームの「あて先イーサネット・フレーム」「送信元イーサネット・フレーム」に続くフィールドは、データ部分に入っているデータの「タイプ」を示す場合と、データの「長さ」を示す場合があります。このフィールドを識別する方法として正しいものを選びなさい。

①．1500～1536 の場合はデータのタイプ、それ以外の場合はデータの長さを示す
②．1500～1536 の場合はデータの長さ，それ以外の場合はデータのタイプを示す
③．1500 以下の場合はデータのタイプ、1536 以上の場合はデータの長さを示す
④．1500 以下の場合はデータの長さ、1536 以上の場合はデータのタイプを示す

■問 3　[b]　最近のネットワーク機器が搭載するようになった機能の一つとして「ジャンボ・フレーム」があります。そのジャンボ・フレームの説明として間違っているものはどれか。

①．データ転送の信頼性を高める技術である
②．一つのイーサネット・フレームに入れるデータ量を増やす技術である
③．処理するフレームの数を減らせる技術である
④．データの転送効率を上げる技術である

解答：



図：まとめ--イーサネット・フレームのしくみ

１章５節　DHCP（Dynamic Host Configuration Protocol）

パソコンをネットワークにつないだだけで、すぐに Web アクセスやメール送受信ができる―。慣れてしまった私たちは当たり前のように使っていますが、よく考えてみるとちょっと不思議です。ネットワークを使うためにパソコンは、自分の住所に当たる IP アドレスなどの情報がわかっていなければうまく通信できないはずです。なぜ何も設定せずに通信できるのでしょうか？

実は、パソコンは、ユーザーが気付かない間にネットワーク接続に必要な設定を済ませています。この設定作業をしているのが、この節で取り上げる DHCP（Dynamic Host Configuration Protocol）というしくみです。パソコン OS が内蔵する DHCP クライアントというソフトが、設定に必要な情報を自動的に調べて通信できるように設定してくれています。目立たないけど重要な仕事をする「縁の下の力持ち」です。

また、DHCP はちょっと変わったしくみを使っています。ネットワークが設定されていない状態でもネットワークを利用して、必要な情報を収集できるようになっているのです。

１．５．１．自動化される接続情報の取得、管理や設定の負担軽減

DHCP は、身近ですがあまり意識することはありません。でも DHCP がなくなるとパソコンをネットワークにつなぐのは面倒になります。いったい DHCP は何のためにどんなことをしてくれているのでしょうか。ここでは DHCP の役割を説明します。

〔1〕通信にはさまざまな情報が必要

パソコンがネットワークを利用するには、いろいろな情報をあらかじめ設定しておく必要があります。DHCP は、それらの情報を自動で取得して設定するしくみです。

どんな情報を事前に設定しておく必要があるのでしょうか。いくつかありますが、一番大事なのはネットワークの中で自分を識別するための「IP アドレス」です。TCP/IP 通信では、IP アドレスという住所を使って通信します。そのために、自分の IP アドレスを正しく設定しなければ、そもそもうまく通信することができないのです。IP アドレスは、製造時にパソコンに割り当てられておらず、接続するネットワークに合わせてネットワーク管理者が割り当てる必要があります。

IP アドレス以外にも、事前に設定しておくべき情報はあります。パソコンは、ルーターという機器に通信の中継を依頼することがあります。どのあて先のときにはルーターに中継を依頼するのかを示す情報（サブネット・マスク）、依頼するルーターはどこにあるのかを示す情報（デフォルト・ゲートウエイ）などを正しく設定しておかないと、うまく通信できません。

さらに、「www.123abc.co.jp」といったよく見る名前を指定して通信するには、その

ドメイン名という名前から IP アドレスを調べる DNS というしくみを使う必要があります。その DNS のサーバーの場所もあらかじめ知っておく必要があります。

〔2〕手作業では大変な手間

これらの情報をすべて手作業で設定するとしたら結構な手間がかかります（図 1-1）。

（図 1-1●パソコンのネットワーク接続を手作業で設定するのは大変！！）

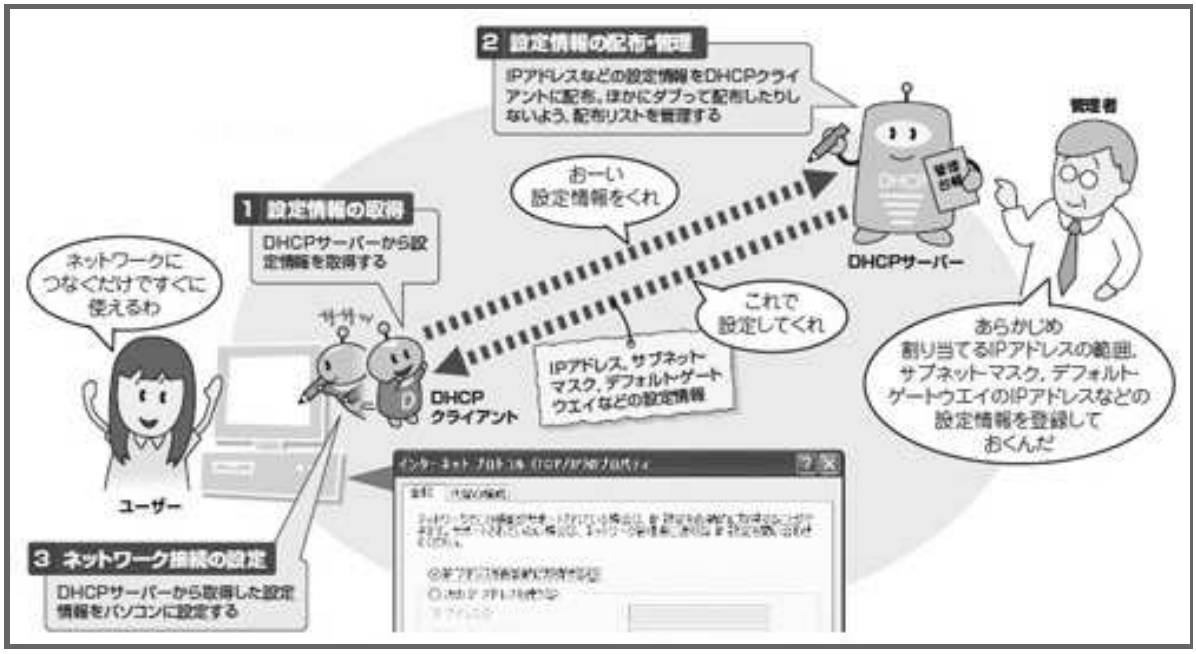
ほかのパソコンとダブらないように管理者から設定情報を割り当ててもらい、それを間違いなく設定する必要があります。

まずネットワーク管理者から自分に必要な設定情報を教えてもらう必要があります（図 1-1 の ①）。管理者は、パソコンをつなぐネットワークを調べて、適切な設定情報を割り振ります。例えば、IP アドレスはほかの端末とダブらないようにきちんと管理しなければいけません。だから管理者は、一般に設定情報の管理台帳を用意しています。その管理台帳で使用中の IP アドレスを把握し、空いている IP アドレスを割り当てて記録を更新しておくわけです。

管理者から設定情報を教えてもらったユーザーは、受け取った情報を間違えずに設定しなければいけません（図 1-1 の ②）。設定を間違えると自分のパソコンが通信できなくなるだけではありません。ほかのパソコンと同じ IP アドレスを設定してしまうと、ほかのパソコンまで通信できなくなってしまいます。

〔3〕手間を省いて間違いもなくす

DHCP は、これらの設定作業の大部分を自動化します。パソコン内部でユーザーの代わりに働くのが DHCP クライアント、ネットワーク上で管理者の代わりに働くのが DHCP サーバーです（図 1-2）。



（図 1-2●DHCP はネットワーク設定を自動化します）

パソコンをネットワークにつなぐと、パソコンの DHCP クライアントが DHCP サーバーから設定情報を取得します。さらに取得した情報をパソコンに設定してくれます。

DHCP クライアントは、パソコンがネットワークにつながったときに DHCP サーバーに設定情報を問い合わせます（図 1-2 の 1）。DHCP サーバーは、管理台帳で空いている IP アドレスを調べて、ほかの設定情報と共に DHCP クライアントに配布します（図 1-2 の 2）。もちろん、配布したあとに管理台帳は更新しておきます。

設定情報を取得した DHCP クライアントは、その情報をパソコンのネットワーク情報として設定します（図 1-2 の 3）。こうしてパソコンはユーザーが設定しなくても通信できるようになるのです。

〔4〕DHCP を使うのが当たり前

DHCP を使うのには特別な設定は要りません。DHCP クライアントは、ほとんどの OS に備わっています。たいていの場合は、パソコンを起動すれば同時に DHCP クライアントも有効になります。

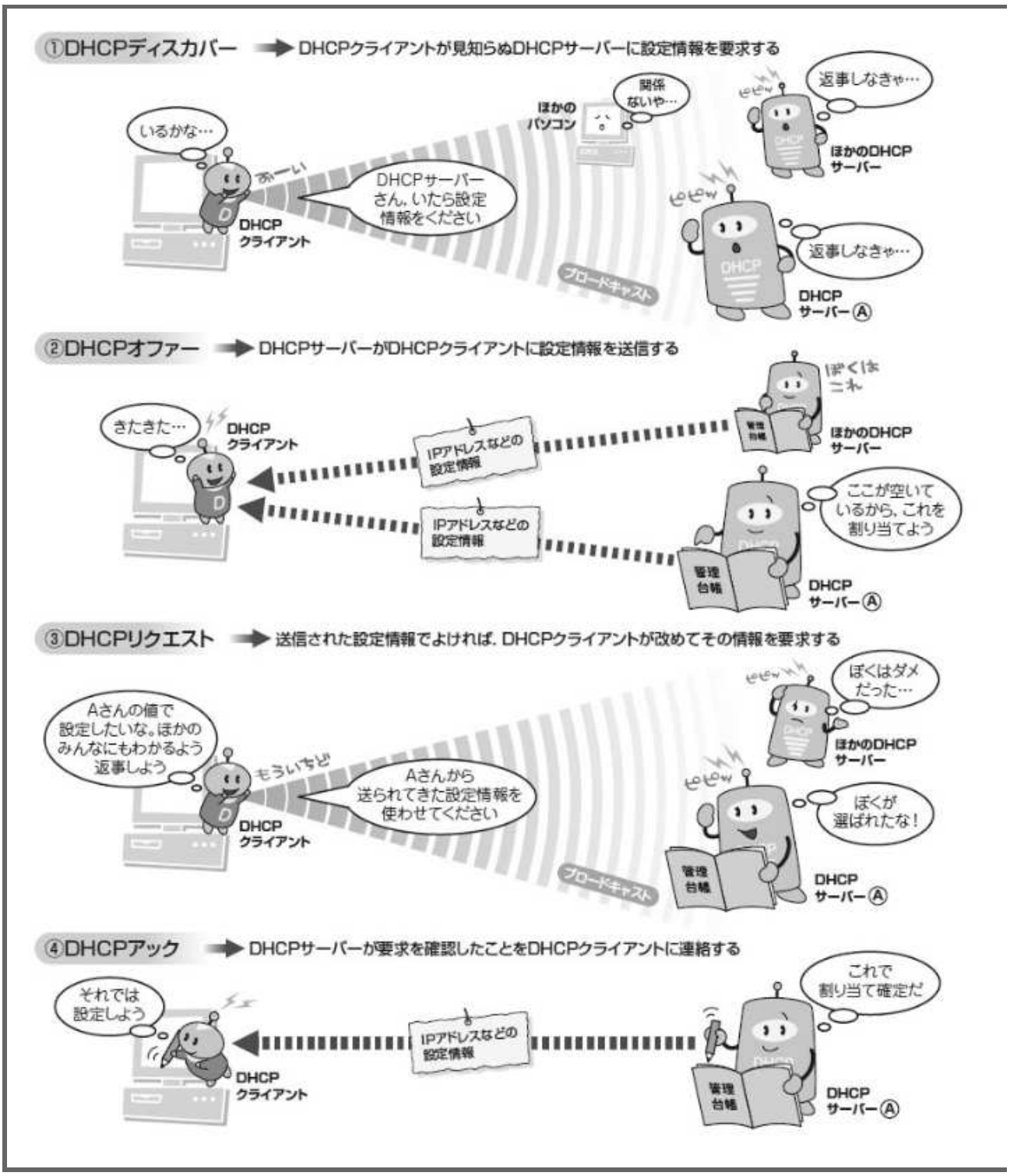
一方、家庭で使うブロードバンド・ルーターは、初期設定が済んだ状態の DHCP サーバーを備えていて、個人ユーザーであればほとんどそのままで使えます。企業のネットワークでは、ルーターの DHCP 機能や専用サーバーを使うことが多く、こちらは、ネットワーク管理者が自社ネットワークに合わせて設定することになります。

1．5．2．設定を取得は２往復、巧みに使うブロードキャスト

【1.5.1】では、DHCP の役割について説明しました。【1.5.2】では、パソコンの DHCP クライアントが設定情報を取得するしくみについて少し詳しく見てみましょう。

〔1〕提案を受けてから正式に要求

パソコンをネットワークにつなぐと、パソコンの DHCP クライアントは DHCP サーバーと 2 往復のメッセージをやりとりします（図２）。



（図２●DHCP クライアントは２往復のやりとりで設定情報を受け取ります）

最初の１往復で設定情報の提示を受け取り、それでよければ次の１往復で割り当てを受けます。

DHCP クライアントはまず、DHCP サーバーを探すために「DHCP ディスカバー」というメッセージを送信します（図２の ①）。DHCP サーバーがどこにいるかはわからないので、すべてのマシンに届ける「ブロードキャスト」という方法で送信します。

DHCP ディスカバーを受け取った DHCP サーバーは、パソコンの IP アドレスや、

ルーターに通信を依頼する範囲を示す「サブネット・マスク」、ルーターの場所を示す「デフォルト・ゲートウエイ・アドレス」、DNS サーバーのアドレスなど、設定情報を用意します。それらの情報を「DHCP オファー」というメッセージに入れて DHCP クライアントに送信します（図２の ②）。

この1往復で、DHCP サーバーを探し出して、設定情報を受け取ることになります。

次に DHCP クライアントは、使いたい設定情報を「DHCP リクエスト」で伝えます（図２の ③）。DHCP リクエストを受け取った DHCP サーバーは、OK を意味する「DHCP アック」を返信します（図２の ④）。これを DHCP クライアントが受け取った段階で設定割り当ては完了です。

単純に考えれば、最初に DHCP オファーで送信された情報を DHCP クライアントがそのまま設定してしまえばよいように思えます。そうしないのは、DHCP サーバーがネットワーク上に二つ以上存在することがあるからです（※１）。そのようなときに、どのサーバーからの設定情報を使用するのかをすべての DHCP サーバーに通知する必要があります。自分が送信した設定情報が採用されなかったことがわかれば、DHCP サーバーはその情報を別の DHCP クライアントで使えるようになるからです。

図 2 をもういちど見てみましょう。DHCP クライアントは相手の DHCP サーバーの IP アドレスがわかっているのに、DHCP リクエストをブロードキャストで送信しています。ここでブロードキャストを使うのは、どの DHCP サーバーから送られてきた設定情報を使うかを、すべての DHCP サーバーに知らせるためです。

（※１：と言うことは、ネットワーク上に複数の DHCP サーバーが動いていても良いということです。）

〔2〕特別なアドレスを使って通信する

このような DHCP のメッセージは、どうやってやりとりしているのでしょうか。通信に必要な設定情報を取得するために通信します。冷静に考えると、そこに矛盾があるように思えます。

でも大丈夫。設定前のパソコンは、設定情報の取得が完了するまでの間、「0.0.0.0」という特別な IP アドレスを使って通信します。つまり、送信元 IP アドレスを「0.0.0.0」にして DHCP ディスカバーや DHCP リクエストを送信します。

ただし、「0.0.0.0」ではネットワーク上のどこを指すかはわかりません。では DHCP サーバーは、どのようにあて先を指定して返信するのでしょうか。原則として DHCP サーバーは、クライアントからの IP パケットを運んでくる LAN フレームに書き込まれている送信元の「MAC アドレス」をあて先に指定します。ただし、クライアントの中には IP 設定が終わるまで自分あての LAN フレームをうまく受け取れないものがあります。このようなときはクライアントからの DHCP メッセージに書き込まれた指

定に従って、ブロードキャストで応答することもあります。

１．５．３．２種類 IP アドレス割り当て法、同じ設定を長く使う

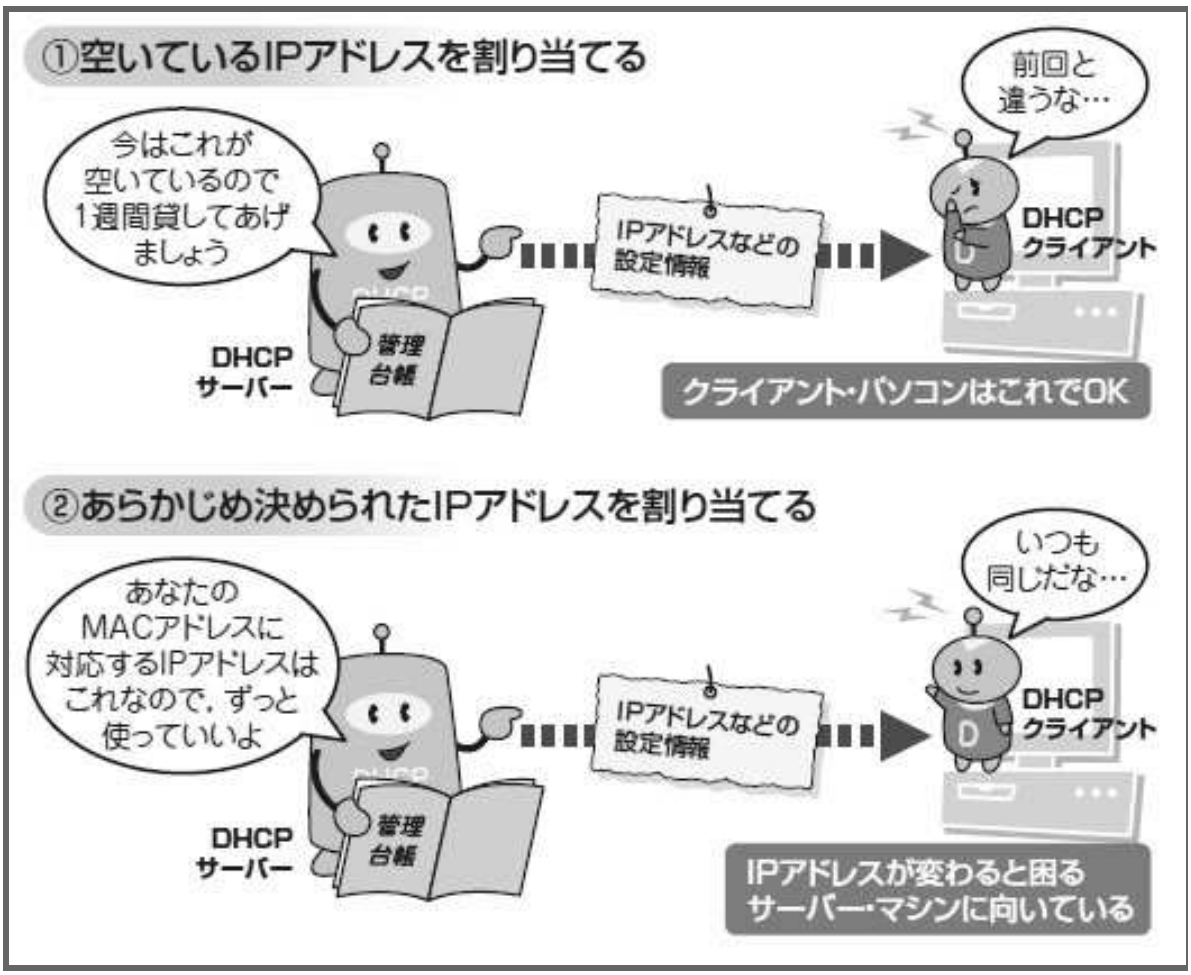
【1.5.3】では、IP アドレスを中心とした割り当てる設定情報の中身について学びます。実は DHCP での IP アドレスの割り当て方は、その場で空きを探して動的に割り当てるものと、あらかじめ決めておいて固定的に割り当てるものの 2 種類があります。

〔1〕期間を指定して割り当てる

DHCP で設定する情報のうち、特に重要なのは IP アドレスです。IP アドレスだけはそれぞれのパソコンに別のものを設定する必要があります。動的に割り当てるようにしておけば、管理者はあれこれ考えずに済みます。

しかし、配布できる IP アドレスには限りがあります。1 回割り当てた IP アドレスを使わないようにすると、配布可能な IP アドレスはなくなってしまうことがあります。

そこで DHCP では、利用期間を指定して設定情報を配布する「リース」という方法を用意します。ネットワーク管理者がリース期間を指定しておけば、その間だけ使える IP アドレスを割り当てます（図 3-1 の ①）。



（図 3-1●DHCP の設定情報の割り当て方はおもに次の 2 種類があります）

空いている IP アドレスを順番に期間を区切って割り当てる方法と、あらかじめ決めておいた IP アドレスを割り当てる方法があります。

このとき DHCP サーバーは、割り当てた IP アドレスと、割り当て先パソコンの LAN カードに付いている MAC アドレス、割り当ての有効期限を管理台帳に記録しておきます。有効期限を過ぎた IP アドレスは、別のパソコンに割り当てることもあります。

〔2〕固定のアドレスを割り当てる

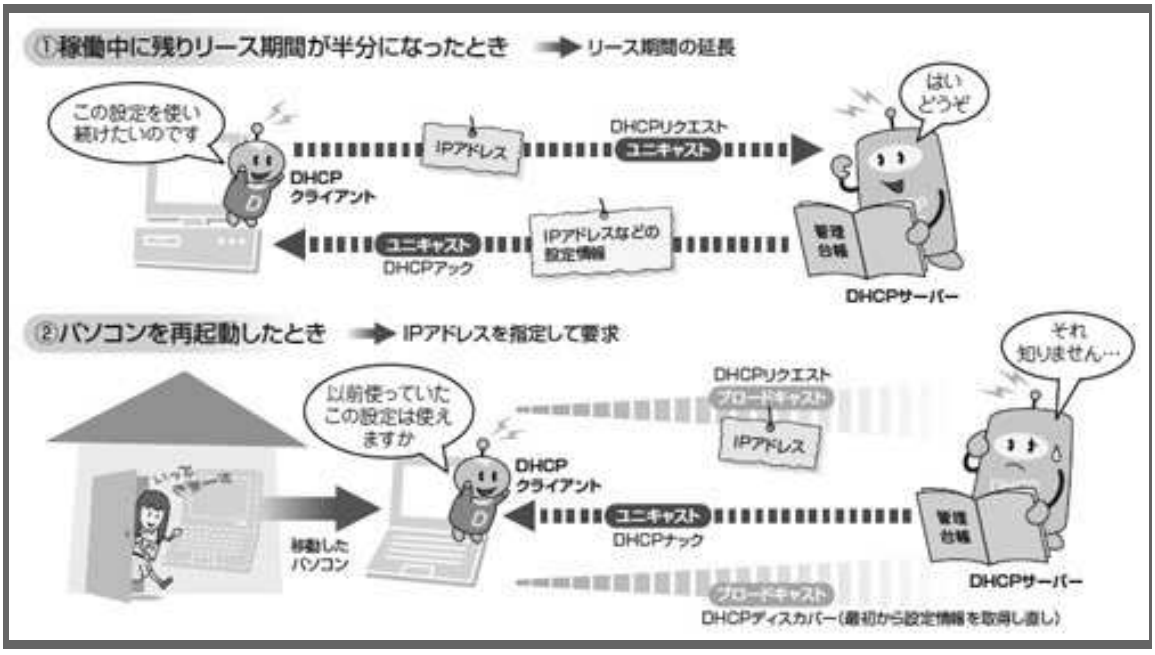
一方、あらかじめ決まった IP アドレスを配布したほうがいいパソコンもあります。例えば DNS に情報を登録するようなサーバーは、IP アドレスが変わらないようにしたいのです。DHCP でも、同じパソコンに特定の IP アドレスを固定的に割り当てられるようにできます。

決まった IP アドレスを割り当てる場合には、割り当てる IP アドレスとそれに対応するパソコンの LAN カードの MAC アドレスを管理者があらかじめ登録しておきます。DHCP サーバーは、DHCP クライアントからのメッセージに書き込まれたパソコンの MAC アドレスと管理台帳を参照して、あらかじめ割り当てる IP アドレスが決ま

っていればその IP アドレスを割り当てます（図 3-1 の②）。

〔3〕なるべく同じアドレスを使い続ける

リース方式で設定情報を割り当てる場合、リース期間が終わるたびに IP アドレスが変わるのでしょうか。実はそんなことはありません。DHCP は、同一のクライアントがなるべく同じ IP アドレスを使い続けるように工夫されています（図 3-2）。



（図 3-2●期間を区切った割り当てでもなるべく同じ設定を使い続けようとします）

期間を区切った割り当てでも、なるべく同じ IP アドレスを使い続けたほうが都合がよいのです。そこで DHCP クライアントはリース期間を延長したり IP アドレスを指定して要求したりします。

ネットワークにつながったままのパソコンは、リース期間を延長することで同じ IP アドレスを使い続けます。

DHCP クライアントは、DHCP サーバーから指定されたリース期間の残り時間が半分になった時点で、リース期間の延長を DHCP サーバーに要求します（図 3-2 の ①）。要求には DHCP リクエストを使います。ただし、ほかの DHCP サーバーに伝える必要はないので、ブロードキャストではなく、ユニキャストという方法で要求先の DHCP サーバーだけに送ります。

延長要求を受け取った DHCP サーバーは、OK を意味する DHCP アックを返します。すると、残り時間はリセットされて、そこから改めてリース期間のカウントが始まります。

また、パソコンを再起動したときも、可能な限り同じ IP アドレスを使い続けるようにしています（図 3-2 の ②）。

再起動すると DHCP クライアントは、自分がそれまで使っていた IP アドレスを DHCP リクエストでブロードキャストします。以前と同じ DHCP サーバーがいて、以前と同じ IP アドレスが空いていれば、DHCP サーバーから DHCP アックを送られて同じ設定情報を使い続けられるようになります。

ここでブロードキャストを使うのは、再起動したときに前と違うネットワークにつながっているかもしれないからです。ブロードキャストで送れば、違うネットワークにつながっていてもそのネットワークの DHCP サーバーにリクエストが届きます。IP アドレスの情報などからリクエストが間違っていることに気付いた DHCP サーバーは、リクエストを否定する応答（DHCP ナック）を送信します。

DHCP ナックを受け取った DHCP クライアントは、違うネットワークにつながっていることに気付いて、設定情報を一から取得し直します。こうして、正しい設定情報を迅速に入手します。

１．５．４．用途を広げる巧みな技

【1.5.4】では、DHCP のオプションとルーター越え技術を見ていきます。DHCP は、接続に最低限必要な情報だけでなく、さまざまなオプション情報を指定できます。また DHCP は、本来はルーターを越えて使えませんが、それができるように DHCP リレー・エージェントというしくみも用意されています。

〔1〕IP アドレス以外はオプション

まずは、オプション情報から見てみましょう。実は、オプション情報は DHCP にとってなくてはならないものです。IP アドレス以外の情報は、すべてオプションで設定することになっています（図 4-1）。DHCP リクエストといった DHCP メッセージの種類もオプションで指定しています。

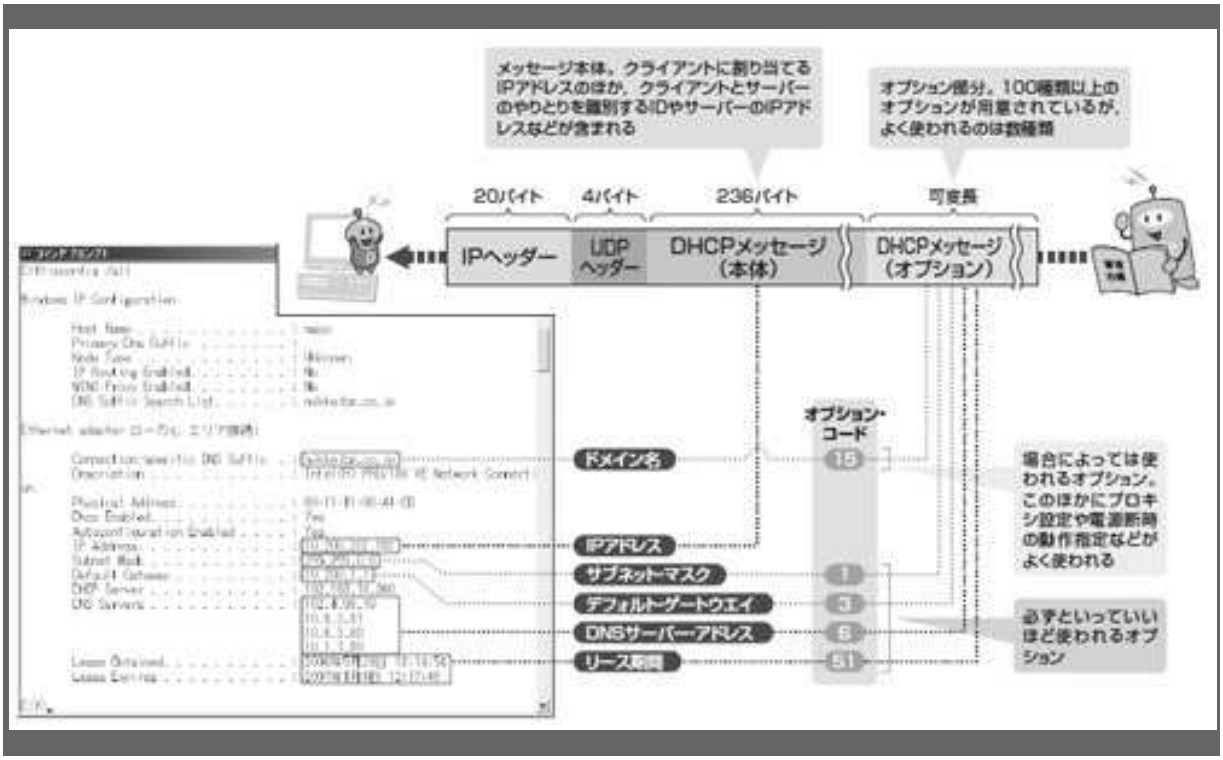


図 4-1●オプションを使うことで IP アドレス以外の設定も自動化しています

DHCP メッセージ本体で割り当てるのは IP アドレスだけ。リース期間やデフォルト・ゲートウエイ，サブネット・マスクといった情報はオプションを使って割り当てます。

それぞれのオプションは、DHCP メッセージのオプション・フィールドという部分に収められます。オプションは指定する内容によってコードが決まっています。オプション・フィールドに必要な分だけ続けてコードと内容を指定します。

オプション・コードは、正式に登録されたものだけで 100 種類以上。これらの中で、必ずといっていいほど使われるのは、サブネット・マスク、デ フォルト・ゲートウエイ・アドレス、DNS サーバー・アドレス、リース期間です。【1.5.3】で出てきた IP アドレスの延長や再利用を要求する DHCP リクエストは、要求する IP アドレスをオプション情報として送信しています。

それ以外にベンダー独自のオプションもあります。ベンダー・オプションに使えるコードの範囲はあらかじめ決まっています。例えばマイクロソフトは、Internet Explorer 用のプロキシ・サーバーの情報を配布するのにベンダー・オプションを使います。

〔2〕リレーを使って設定を一元管理

最後に DHCP リレー・エージェントのしくみを紹介しておきましょう。DHCP リレー・エージェントは、別のネットワークにある DHCP サーバーに DHCP メッセージを中継するしくみです。ルーターの機能として実装されているのが普通です。なぜそ

んなものが必要なのでしょうか。理由は、ブロードキャストで送られるメッセージはルーターを越えて別のネットワークに届かないからです。DHCP のやりとりがうまくいかなくなってしまいます（図 4-2 の（1））。

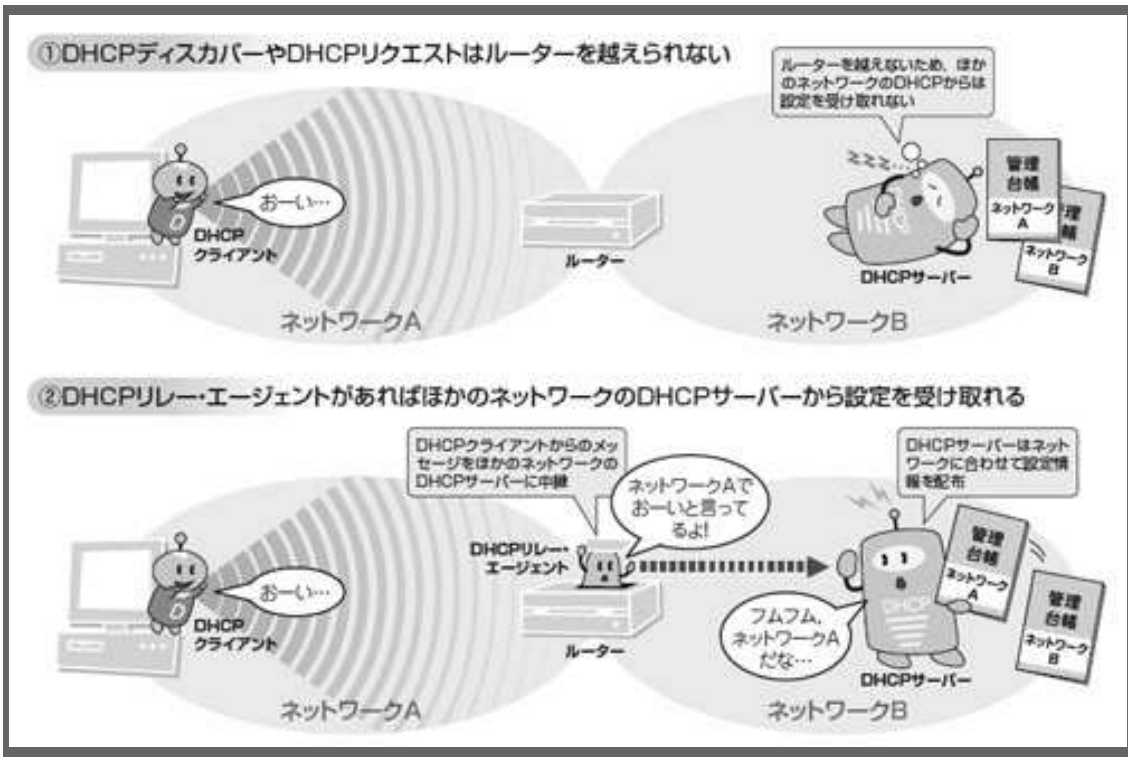


図 4-2●リレー・エージェントを使えばほかのネットワークの DHCP サーバーから設定情報を受け取れます

1 台の DHCP サーバーで、複数のネットワークの設定情報をまとめて一元管理できます。

DHCP リレー・エージェントは、DHCP クライアントと同じネットワークにあって、別のネットワークにある DHCP サーバーとのやりとりを中継します（図 4-2 の ②）。DHCP クライアントからブロードキャストで送られてくるメッセージをいったん受け取って、ユニキャストの IP パケットに直して DHCP サーバーに転送します。

DHCP サーバーからの応答も、いったん DHCP リレー・エージェントが受け取って DHCP クライアントに転送します。こうすれば、1 台の DHCP サーバーで複数のネットワークに設定を割り当てられるようになります。

DHCP リレー・エージェントには、ネットワーク管理者が対応する DHCP サーバーの IP アドレスをあらかじめ登録しておきます。また管理者は、DHCP サーバーにも DHCP リレー・エージェントの IP アドレスと対応するネットワークを登録したうえで、各ネットワークの設定情報を登録する必要があります。

１．５．５．チャレンジコーナー

■問 1　以下は、DHCP を説明した文章です。［　a　］～［　d　］に入れる適切な字句を選択肢から選びなさい。

DHCP は dynamic host ［　a　］ protocol の略で、パソコンがネットワークの設定情報を取得するためのプロトコルです。パソコンが内蔵する［　b　］が、［　c　］から設定情報を取得するのが基本的なしくみです。［　b　］を内蔵するパソコンは、まず設定情報を割り当ててくれる［　c　］を［　d　］。このために DHCP ディスカバーを送信します。次にパソコンは DHCP オファーで設定情報の提示を受け取り、あらためて DHCP リクエストで設定情報を要求します。これに対する DHCP アックを受け取って設定情報の取得が完了します。

選択肢：

configuration	commitment	DHCP サーバー
DHCP クライアント	探し出す	指定する

■問 2　次の図は、設定情報を取得するときと、リース期間を延長するときの DHCP のやりとりを示した図です。空欄［　a　］に時間を、空欄［　b　］～［　d　］に IP パケットの送信方法を選択肢から選んで記入しなさい。ただし、ここでのリース期間は 2 日間とします。

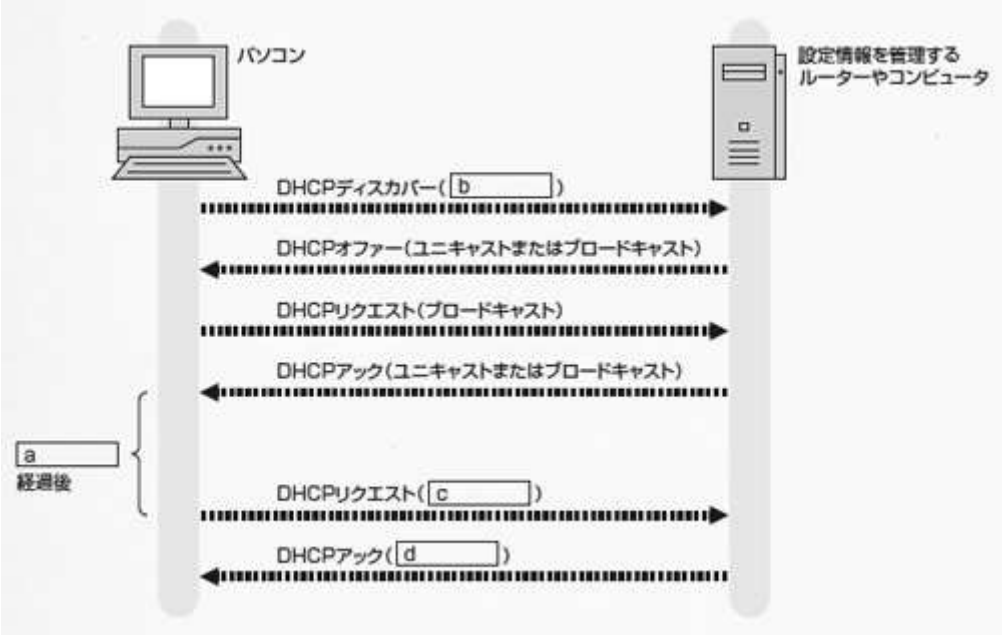


図:問 2

［　a　］ 12 時間　　1 日　　36 時間　　12 日

［　b，c，d　］ ユニキャスト

ブロードキャスト

ユニキャストまたはブロードキャスト

マルチキャスト

■問 3　[a]最初に設定情報を取得するとき、パソコンが DHCP リクエストをブロードキャストで送信するのはなぜか。その理由として正しいものを選びなさい。

①．同一ネットワークに複数の DHCP サーバーがあるときに備えて、なるべく多くの DHCP サーバーから設定情報を取得するため

②．同一ネットワークに複数の DHCP サーバーがあるときに備えて、複数の DHCP サーバーから設定情報を取得して，最適なものに絞り込んでいくため

③．同一ネットワークに複数の DHCP サーバーがあるときに備えて、どの DHCP サーバーの設定情報を使用するかをほかの DHCP サーバーにわかるようにするため

④．同一ネットワークに複数の DHCP サーバーがあるときに備えて、IP アドレスがわからなくても DHCP リクエストを送れるようにするため

■問 3　[b]　DHCP リレー･エージェントの説明として間違っているものを選べ。

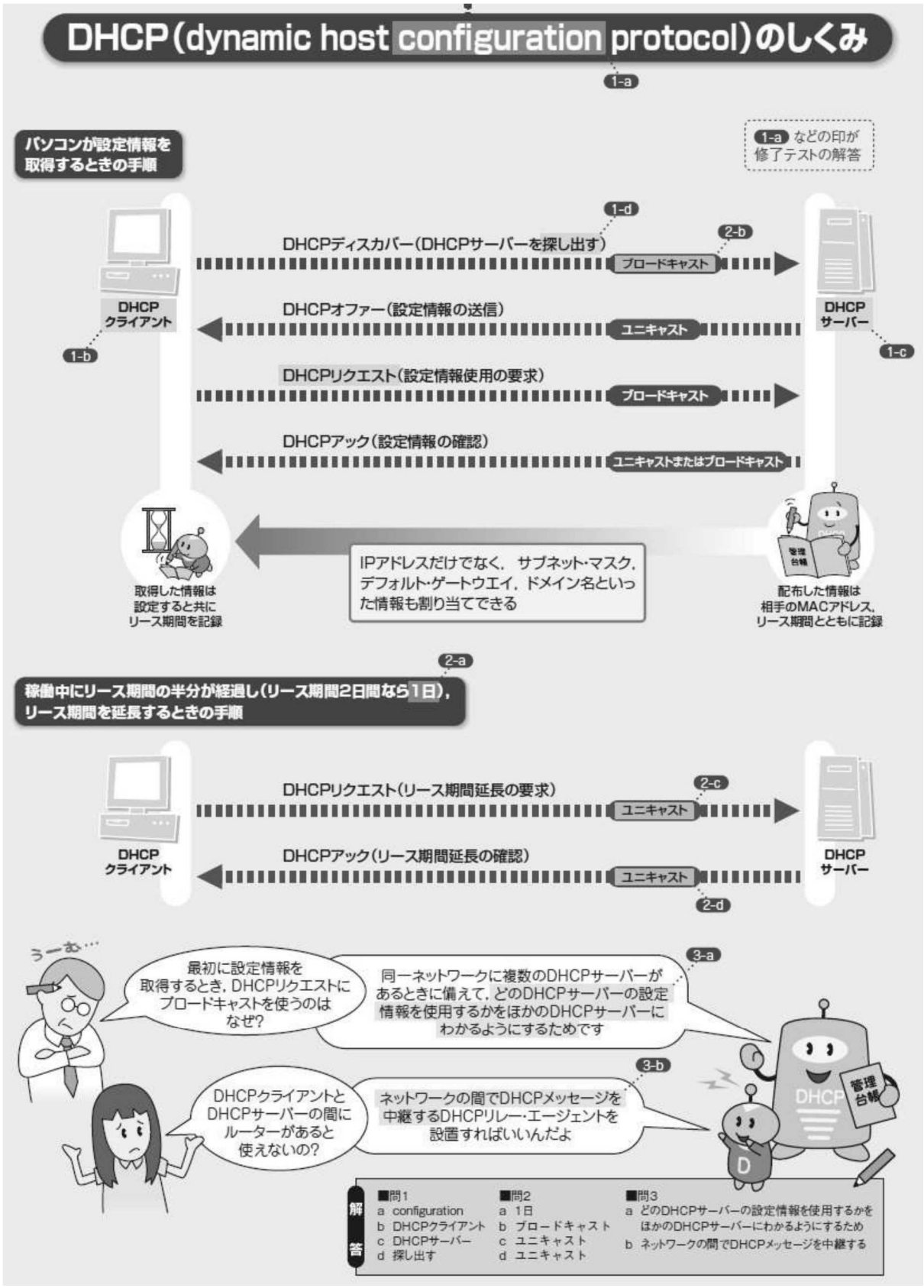
①．別のネットワークにある DHCP サーバーに DHCP メッセージを中継するためのしくみである

②．DHCP サーバーの一機能として実装されている

③．DHCP クライアントからブロードキャストで送られてくるメッセージをいったん受け取って、ユニキャストの IP パケットに直して DHCP サーバーに転送する

④．1 台の DHCP サーバーで、複数のネットワークの設定情報をまとめて一元管理できる

解答：



図：まとめ--DHCP のしくみ

２章　ECHONET Lite（エコーネット・ライト）

ECHONET Lite は、スマートハウスを実現する通信プロトコルです。経済産業省が「家庭内機器および HEMS とスマートメーター間における公知な標準インターフェース」として推奨しています。家庭内でも WiFi 規格などの無線ネットワークが簡単に活用できるようになり、スマートフォンやコントローラから、家にあるエアコン、照明などを制御したいとか、電力の無駄遣いを抑えるために家の電気代を把握したいとかいう要望が増えています。

そのような、「省エネ・快適・安全・安心」な生活を実現するためには、どのメーカーの機器でも共通に接続できる通信プロトコルが必要です。その役割を果たすために作られたのが ECHONET Lite です。

ECHONET Lite は、すでに普及しているインターネットにも対応し、簡単で使いやすさを重視した通信プロトコルになっています。これを理解すれば、みなさんが自分で開発したいろいろな機器を、家庭のネットワークに接続することができます。テーブルの上に置いた野菜栽培システムなどを繋げば、これは植物工場モデルとも言えるシステムとなって、このネットワーク規格がスマート自動車、スマートハウス、スマート農業にまで対応できる要素を持つことが分かるでしょう。

この章では、この ECHONET Lite の規格書を読んで、後の章で行う「スマート家電」関連のモデル開発に役立てようと考えています。規格書全文を掲載することはできませんが、以後の章に関連する部分や、見ておくと面白い規格は、抜粋して掲載しましたので、是非ご覧になってください。

ECHONET Lite の規格書は、エコーネット・コンソーシアム（http://www.echonet.gr.jp）で作られていて、公開された規格書は、次の Web サイトから自由にダウンロードができます（http://www.echonet.jp/spec/）。

※URL は変更になる場合がありますので、上記リンクでページが開かない場合は、「エコーネット・コンソーシアム」で検索して、規格書のダウンロードを行ってください。

２章１節　規格書の構成

ECHONET Lite の規格書の第１部で規格書の構成とその内容について次のように

説明しています。

５．１ 規格書の構成

本 ECHONET Lite 規格書の全体構成は、以下のようになっています。

第１部　ECHONET Lite の概要

ECHONET Lite の目的、特徴、全体アーキテクチャ、基本用語の定義、ECHONET Lite 機器の種類などについて説明しています。

第２部　ECHONET Lite 通信ミドルウェア仕様

ECHONET Lite 通信ミドルウェアにおける、電文フォーマット、プロトコル処理、機器オブジェクト定義、立ち上げシーケンスなどの仕様について説明しています。

第３部　ECHONET Lite 通信装置仕様

機器を通信装置ハードウェアとして見たときのハードウェア仕様や、ミドルウェアアダプタ通信インタフェース仕様について説明しています。

第４部　ECHONET Lite ゲートウェイ仕様

ECHONET Lite サービスミドルウェアとしての ECHONET Lite ゲートウェイのソフトウェア仕様について説明しています。

第５部　ECHONET Lite システム設計指針

ECHONET Lite システムを設計する上での指針について説明しています。

APPENDIX　ECHONET 機器オブジェクト詳細規定

ECHONET 機器オブジェクトの詳細仕様について説明しています。

ソフトウエア開発は、これら規格書の 第１部、第２部、第５部と Appendix を主に学習すれば、開発に必要な知識が得られます。

次の節から、該当する部分を見ていきましょう。

※注）この規格は、２０１１年１２月に Ver.1.00 と Appendix Release A が一般公開された後も、開発が続けられていて、本書執筆時点の最新バージョンは、２０１４年公開の Ver.1.11 と２０１５年６月公開の Appendix Relese G となっています。本書では、これらのバージョンについて、その一部を抜粋しています。

２章２節　（第１部）　ECHONET Lite の概要

第１部は、規格の全体象を説明しています。

２．２．１．開発背景（１．１）　※以後（）内は、規格書の項番を示します。

現代社会が抱えている環境問題を顕在化しています。

①．CO2 削減

②．地球環境

③．エネルギー

④．高齢化社会

一方では、社会のデジタル化に伴う、通信インフラの急速な整備拡大でインターネットのような繋がりを持つ機会が増加しています。また安心・安全・快適・環境保全に対応したサービスへの必要性と期待も高まっています。

すでに家庭内にも敷設された通信インフラと IoT などに伴う様々な機器・家電・設備がネットワーク化されることで、社会の期待に対応することが可能となります。

２．２．２．ECHONET Lite 開発の目的（１．２）

ECHONET（Energy Conservation and Homecare Network）は、設備系のネットワークとして、低コストで工事が不要なうえすでに建築された住宅にも対応できる通信プロトコルですが、それをさらにスリム・シンプル化し使いやすくしたものとして ECHONET Lite を開発することとしました。

２．２．３．ECHONET Lite のねらい（１．３）

ECHONET Lite は、利用者・製品開発者・システム設置者ら様々なユーザーニーズに対応するよう、次のねらいに基づいています。

(1)．マルチベンダで容易にシステム構築可能

(2)．設備機器の寿命の長さ、ホームシステム普及過程に対応

(3)．容易なシステムインストール、機器設置・交換・移設

(4)．他システムとの接続、共存可能

２．２．４．ECHONET Lite の適用対象フィールド（１．４）

図 1-1 に示すように、ECHONET Lite の対象フィールドは一般住宅だけでなく、集合住宅や店舗、あるいはビルなどにも適応しています。

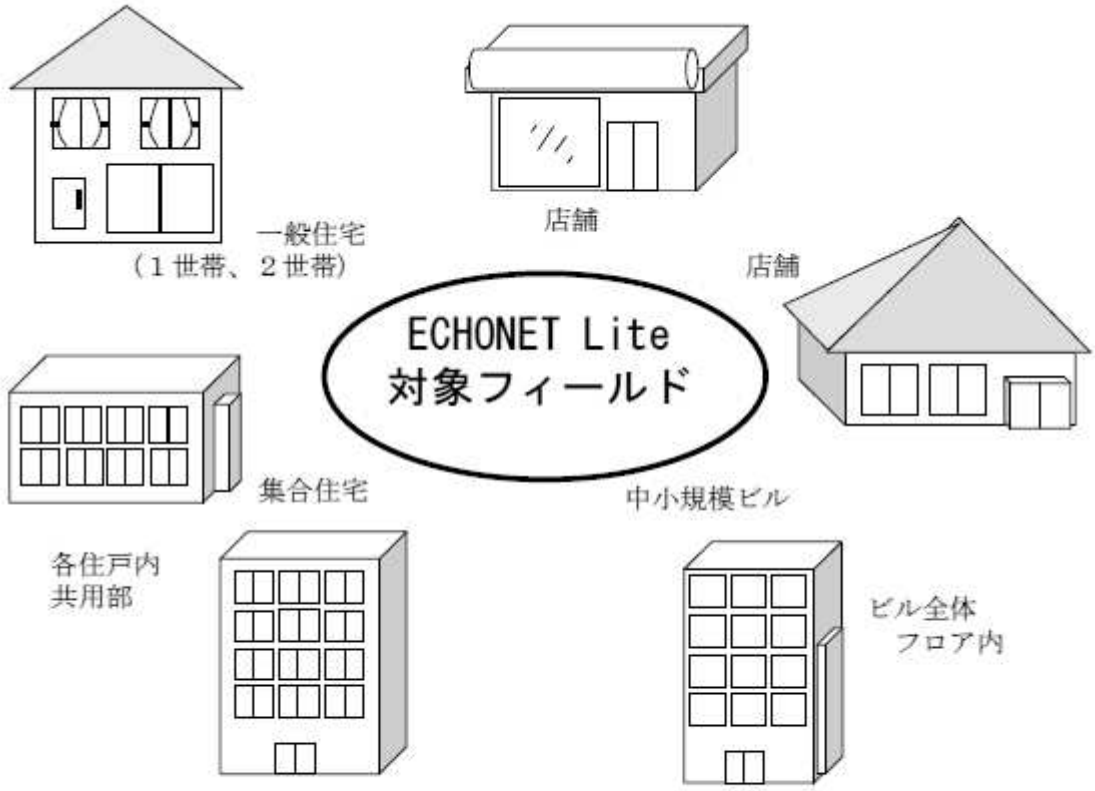


図 1-1　ECHONET Lite 対象フィールド

２．２．５．ECHONET Lite システムアーキテクチャ（２．１）

図 2-1 に ECHONET Lite のシステムアーキテクチャを示します。

①．ドメイン：ECHONET Lite のネットワーク範囲に存在する管理対象リソース（住設製品、家電製品、センサ、コントローラ、リモコンなど）

②．システム：機器、機器を監視・制御・操作するコントローラ間、機器間で通信・連携動作します

③．ゲートウェイ：ドメイン間または、ドメインと他の ECHONET Lite 外のシステムを接続します

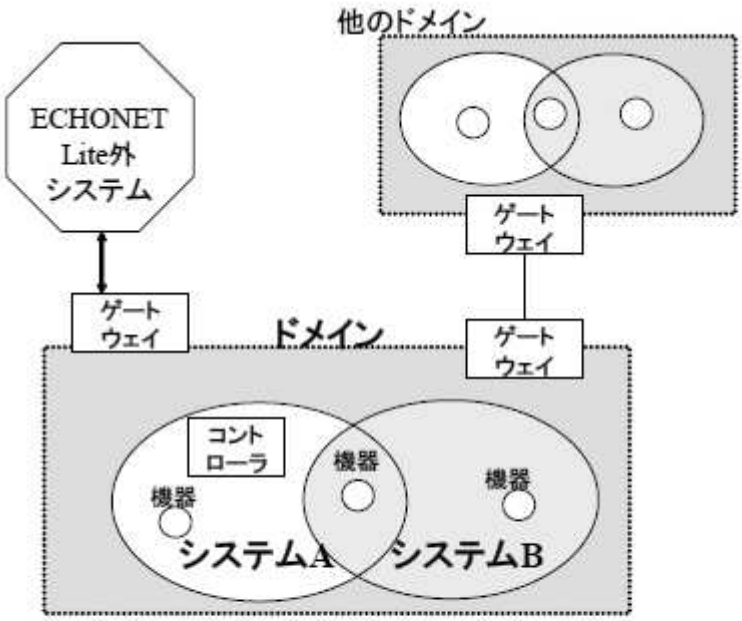


図 2-1　ECHONET Lite システムアーキテクチャ

2．2．6．ECHONET Lite ネットワーク構成（2．2）

ECHONET Lite のネットワーク構成モデルを図 2-3 に示します。ドメインは複数のサブネットで構成されます。サブネット内では、MAC アドレスもしくは IP アドレスをサブネット内での ECHONET Lite 機器のユニークな識別子として使います。

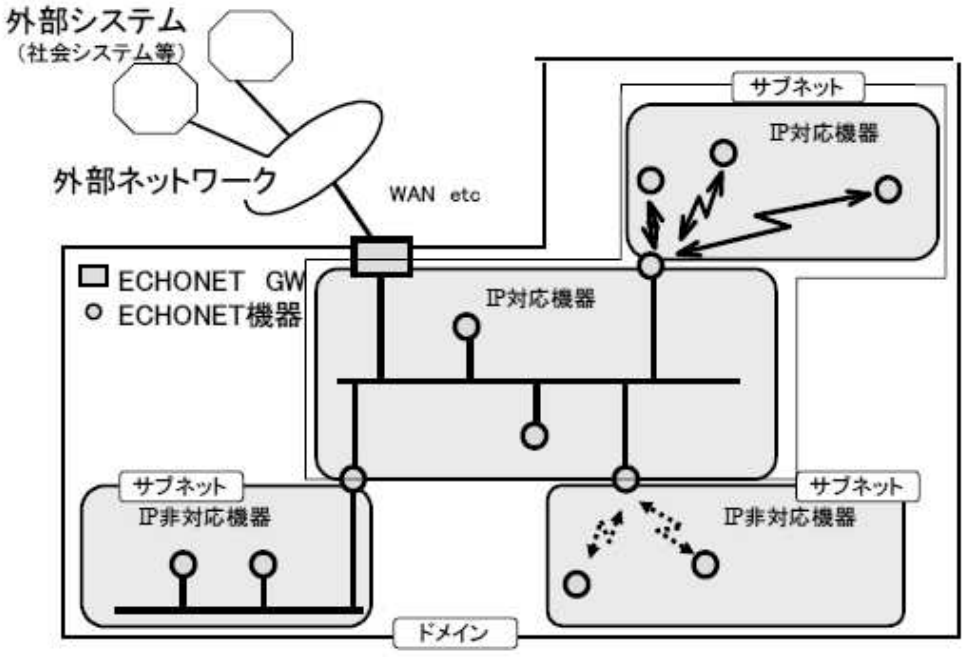


図 2-3　ECHONET Lite のネットワーク構成モデル

２．２．７．ECHONET Lite 機器構成（２．３）

ECHONET Lite 機器構成は次の通りとなっています。

⑴ ECHONET Lite ノード

ECHONET Lite 規格に準拠した通信ノード。ECHONET Lite 内では、ECHONET Lite アドレスによってユニークに識別される ECHONET Lite の通信機能です。ノードの持つアプリケーション機能に区別はなく、ECHONET Lite 上の１通信端末としての機能を述べる際に用います。ECHONET Lite ノードのうち、送受信機能を備えたものを一般ノード、自己のプロパティ値通知を行う機能のみ備えたものを送信専用ノードと定義する。

⑵ ECHONET Lite 機器

住宅設備機器、家電製品、ビル・店舗設備機器、すなわち、照明、空調、冷蔵、電力設備、一般白物家電製品、センサ、アクチュエータなどで、ECHONET Lite 規格に準拠した通信インタフェース、システム対応機能を備える ECHONET Lite ノード。また、これらを監視、制御、操作する機能を持つ集中制御装置や、操作器（リモコン等）のコントローラ機能を備える ECHONET Lite ノード。なお、常時通電されており送受信可能な機器だけではなく、電池駆動の機器など、消費電力を極力抑えたい機器にも対応するため、ECHONET Lite では送信のみ可能な機器として、自己のプロパティ値通知のみを行う送信専用機器を定義する。

⑶ ECHONET Lite ミドルウェアアダプタ

ECHONET Lite の普及過程などで、ECHONET Lite 通信ミドルウェアを具備しない機器を、ECHONET Lite に接続するためのアダプタ。機器と、ECHONET Lite ミドルウェアアダプタ間のインタフェース仕様は、別途規定する ECHONET Lite ミドルウェアアダプタ通信インタフェース仕様に準拠します。

⑷ ECHONET Lite ゲートウェイ

ECHONET Lite のドメインと、外部システム（他の ECHONET Lite ドメインを含む）とを接続する機能を有する。ECHONET Lite ゲートウェイは、接続する外部システムの違いなどにより、ドメイン内に複数存在することを可能とします。

２．２．８．ECHONET Lite と外部ネットワーク、システムとの接続（２．４）

ECHONET Lite の外部接続は、ゲートウェイを使います。識別は外部が行います。

住宅、ビル、店舗等では、ECHONET Lite の外部ネットワークとして、社会システムと接続するネットワークや、映像・情報系のデータ伝送を扱うネットワークなど、多種が存在します。フィールドネットワークとしての位置付けとする ECHONET Lite では、これらをドメイン外とし、ECHOENT ゲートウェイを介しアプリケーションレベルで接続します。また、外部からの直接メッセージ入出力を行う際にも、アプリケーションレベルでのプロトコル変換を行うものとします。また、ECHONET Lite にとってドメインを識別する特別な識別子は特に設けません。ECHONET Lite ドメインをそれぞれ区別したいのは外部システムであり、外部システムの責任において各ドメインを識別する方法が採られるものとします。

２．２．９．第３章 ECHONET Lite 通信レイヤ構成（３．１～３．２）

第３章は、規格書を抜粋して掲載します。

第３章 ECHONET Lite 通信レイヤ構成

３．１ ECHONET Lite 通信レイヤ構成の概要

ECHONET Lite 機器の通信レイヤは、大きく、アプリケーションソフトウェア、通信ミドルウェア、下位通信層の３階層に分けられます。ECHONET Lite 規格では、図 3-1 の ECHONET Lite の通信レイヤ構成図に示される処理ブロックあるいは処理ブロック間インタフェースのうち、網掛け部分の通信ミドルウェアの仕様を規定します。

・アプリケーションソフトウェア

アプリケーションソフトウェアは、大きく、コントローラなどにおいて、システムに接続される機器を遠隔制御するアプリケーションソフトウェアと、エアコンや冷蔵庫などの個別の機器において、その機器のハードウェアとして機能を実現するソフトウェアの２つに分類されます。

・ECHONET Lite 通信ミドルウェア

ECHONET Lite 通信ミドルウェアは、アプリケーションソフトウェアと下位通信層に挟まれた位置に設けられ、ECHONET Lite 通信プロトコルに沿った通信処理を行うものです。ECHONET Lite としての特徴の主なものは、この ECHONET Lite 通信ミドルウェアによって実現されています。

・下位通信層

下位通信層は、電灯線、無線、赤外線などの伝送メディア特有の通信プロトコル処理を行うソフトウェアと伝送メディアそのものを指し、主にＯＳＩ参照モデルのレイヤ１～4 に相当する通信処理を行います。

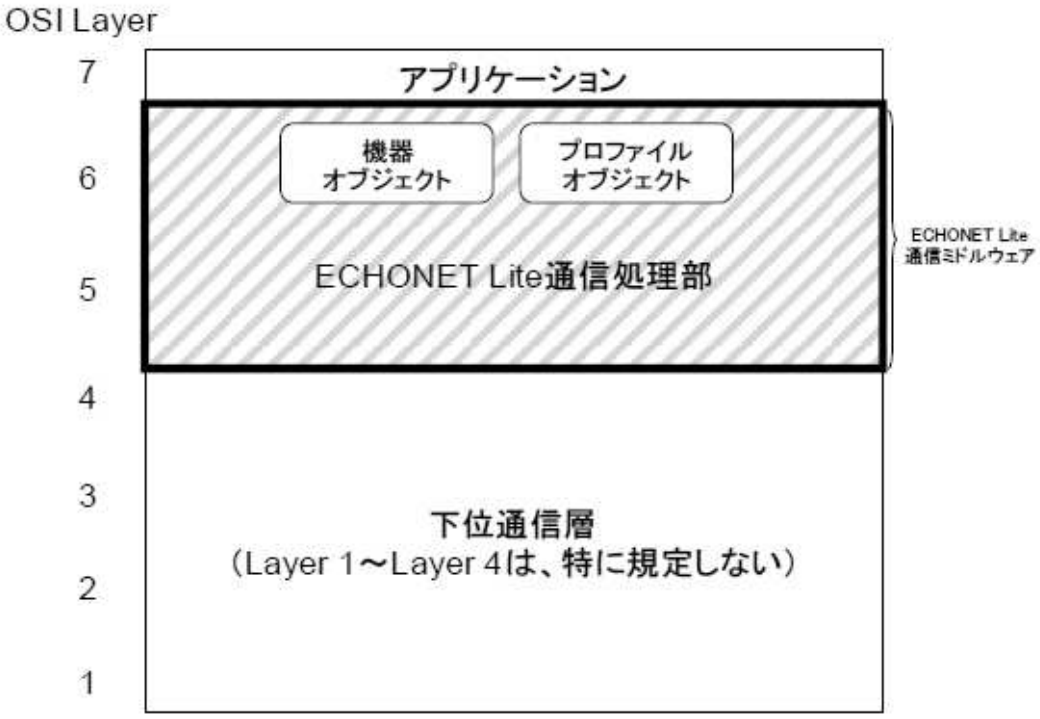


図 3-1　ECHONET Lite の通信レイヤ構成

3．2 ECHONET Lite 通信ミドルウェアの構成要素と処理対象

ECHONET Lite として規格化する ECHONET Lite 通信ミドルウェアには、その構成要素として、ECHONET Lite 通信処理部と ECHONET オブジェクトがあります。

3．2．1 ECHONET Lite 通信処理部

ECHONET Lite 通信処理部では、アプリケーションソフトウェアが設備系システムの機器を遠隔制御したり機器の状態をモニタしたりする際の処理を簡単に行えるようにするための通信プロトコル処理や、通信プロトコル処理のための情報の保持や、自機器あるいは他機器の状態などの様々な情報の管理を行います。すなわち、ECHONET Lite 通信処理部では、他の機器の機器オブジェクトなどのオブジェクトに対するアクセスを行うための通信処理を行います。本規格では、この通信プロトコルを規定しています。また、ECHONET Lite 通信処理部が保持する情報のうち他の機器に対して公開する情報およびアクセス手順を、オブジェクトとして表現し、これを ECHONET オブジェクトとして規定します。

3．2．2 ECHONET オブジェクト

ECHONET オブジェクトは、機器オブジェクトと、プロファイルオジェクトに分類さ

れます。

3.2.2.1 ECHONET 機器オブジェクト

機器オブジェクトは、センサやエアコン，冷蔵庫などの設備機器あるいは白物家電機器が保持する情報やリモートで操作可能な制御項目を論理的にモデル化したものであり、遠隔制御のためのインタフェース形式を統一したものです。また、この機器オブジェクトを機器の種別毎に規定するので、異なるメーカの機器であっても同一機器種別であれば、まったく同じ操作で遠隔制御ができるようになります。具体的には、各々の機器が持つ情報や制御対象を機器オブジェクトのプロパティとして規定し、またこれに対する操作方法(設定、参照）を規定します。

なお、機器オブジェクトの定義は、JEM-1439 に規定されている HK(House Keeping)系コマンドを活用して行いました。

また、JEM-1439 では主に家庭内の機器を対象としていますが、本 ECHONET Lite 規格では、さらに、中小ビル、店舗用の機器を対象に規格化を進めています。

3.2.2.2 プロファイルオブジェクト

プロファイルオブジェクトは、ECHONET Lite ノードの動作状態や、メーカ情報、機器オブジェクトリスト等 ECHONET Lite ノードが保持するプロファイルに関する情報を、アプリケーションソフトウェア及び他の ECHONET Lite ノードが操作するためのインタフェース形式を統一したものです。このように、プロファイルオブジェクトを規定するので、異なるメーカの ECHONET Lite ノード（ノード）であっても、まったく同じ操作でプロファイルを制御することができます。具体的には、各々のノードが持つ情報や制御対象をプロファイルオブジェクトのプロパティとして規定し、またこれに対する操作方法(設定、参照）を規定します。

２．２．１０．ECHONET Lite 機器のタイプ（４．２）

ECHONET Lite 機器は、次の２つのタイプがあります。

(1). フル ECHONET Lite 機器（Fuu_Devixe）

ECHONET Lite レディ機器に接続ができる機器です。

(2). ECHONET Lite レディ機器（Ready_Device）

ECHONET Lite ミドルウエアアダプタと接続することで ECHONET Lite につながる機器です。

２．２．１１．ECHONET Lite 接続のためのミドルウエアアダプタ（４．３）

ECHONET Lite レディ機器を、ECHONET Lite ミドルウエアアダプタと接続することで、ECHONET Lite システムに接続できるようにします。

２．２．１２．機器の ECHONET Lite ネットワークへの接続形態（４．４）

図 4-1 に機器の ECHONET Lite ネットワークへの接続形態を示します。

形態１：フル ECHONET Lite 機器が直接ネットワークに接続される場合

形態２：ECHONET Lite レディ機器が ECHONET Lite ミドルウェアアダプタを介してネットワークに接続される場合

形態３：既存機器がアダプタを介してネットワークに接続される場合

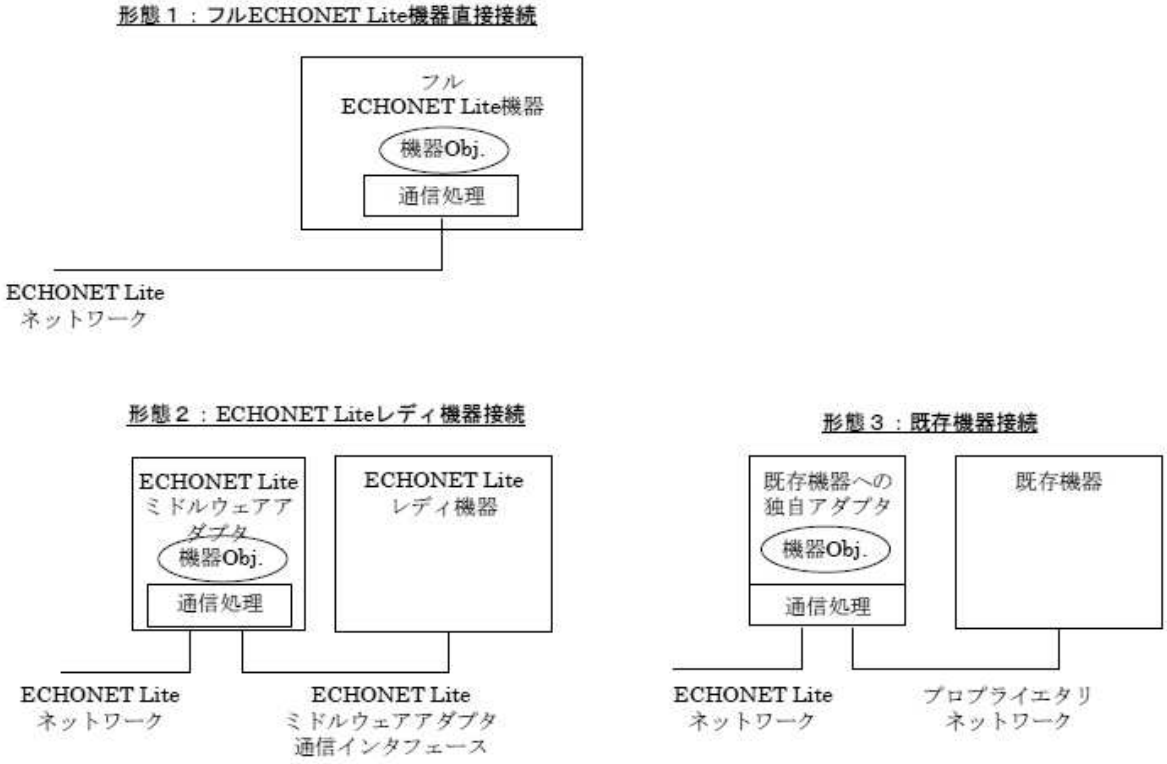


図 4－1　各種アダプタと ECHONET Lite 機器あるいは既存機器との接続の組み合わせ

ECHONET Lite 規格では、図 4－1 で示した接続形態のうち、形態１におけるフル ECHONET Lite 機器と形態２における ECHONET Lite アダプタを規定する。形態３の「既存機器」は、ECHONET Lite 通信機能を備えるアダプタと接続することによってフル ECHONET Lite 機器と同等とみなすことができます。

２．２．１３．対象読者（５．２）

規格書をそのまま掲載します。

５．２ 対象読者

本 ECHONET Lite 仕様規格書の読者として、本節では、ECHONET Lite 機器開発者、ECHONET Lite ミドルウェアアダプタ開発者、アプリケーションソフトウェア開発者、システム開発者・管理者を想定し、それぞれの読者が本仕様規格書のどのパートを中心に読むべきかを説明します。

(1) ECHONET Lite 機器開発者

ECHONET Lite 機器開発者は、基本的には本仕様規格書のすべてのパートを読むべきですが、ECHONET Lite 通信レイヤ構成上の構成要素のうち、各開発者が担当する部分に関するパートを中心に読んでください。

(2) ECHONET Lite ミドルウェアアダプタ開発者

ECHONET Lite ミドルウェアアダプタ開発者は、「第２部　ECHONET Lite 通信ミドルウェア仕様」と「第３部　ECHONET Lite 通信装置仕様」を中心に読んでください。

(3) アプリケーションソフトウェア開発者

アプリケーションソフトウェア開発者は、制御対象機器のプロトコル上の挙動や制御項目を「第２部　ECHONET Lite 通信ミドルウェア仕様」及び「APPENDIX　ECHONET 機器オブジェクト詳細規定」を読んで理解してください。

また、特にコントローラ上のシステムアプリケーションの開発者は、「第 5 部 ECHONET Lite システム設計指針」を参考にしてください。

(4) システム開発者・管理者

システム開発者・管理者は、「第５部　ECHONET Lite システム設計指針」を理解された上で、制御対象機器のプロトコル上の挙動や制御項目を「第２部　ECHONET Lite 通信ミドルウェア仕様」を読んで理解してください。

２章３節　（第２部）ECHONET Lite 通信ミドルウェア仕様

第２部は、ECHONET Lite 通信ミドルウェアの、電文フォーマット、プロトコル処理、機器オブジェクト定義、立ち上げシーケンス等の処理 について規定しています。

この節では、モデルスマート家電を開発するときに重要になる、電文フォーマットについて、抜粋しています。この部分を見ておくことで、マイコンによるモデル機器のプログラムは、理解が容易になります。

２．３．１．通信レイヤ上の位置づけ（１．２）

規格書の抜粋を掲載します。

以後の章で実務上特に重要なことは、次の２点です。

①．UDP を使用できること。

②．ポートは３６１０を使うこと。

規格書の上記事項の解説部分を良く読んでください。

１．２ 通信レイヤ上の位置づけ

通信ミドルウェアは、アプリケーションソフトウェアと、下位通信層の間に位置するものであり、本書（第２部）でその仕様を規定する。本書にて規定する通信ミドルウェア部を、図 1-1 に網掛けにて示した。

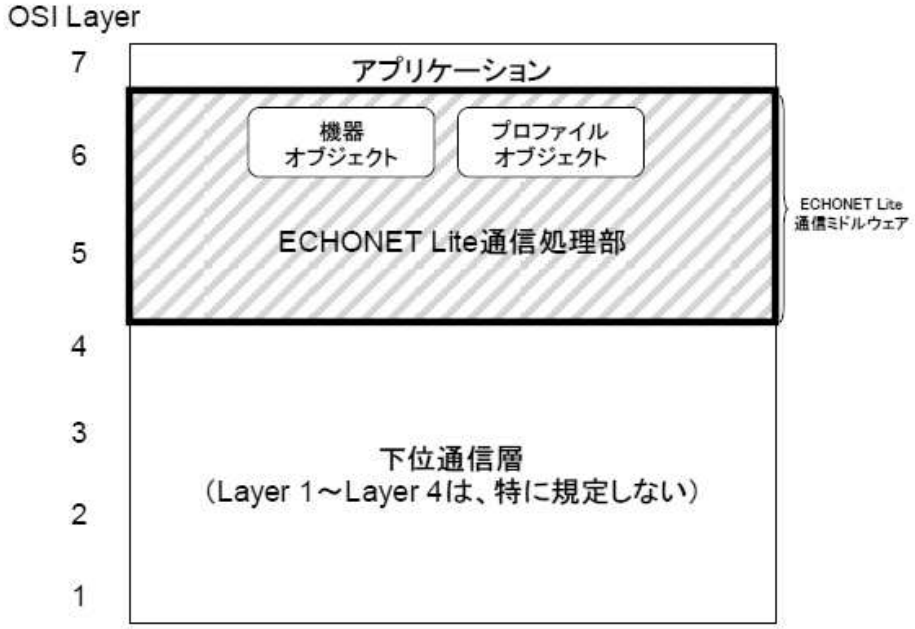


図 1-1 通信ミドルウェア部

図 1-1 に示すように、本書（第 2 部）で規定する通信ミドルウェア部は、ECHONET Lite 通信処理部により構成される。ECHONET Lite 通信処理部は、Layer1～Layer 4 に非依存な機能として規定する。ECHONET Lite 通信処理部は、第 3 章にて規定する ECHONET Lite フレームの送受信を行う。送信形態には、個別送信と一斉同報送信の 2 種類がある。個別送信とは、ECHONET Lite サブネット内において、レイヤ 4 以下のアドレスを用いて宛先を指定し、特定の ECHONET

Lite ノードに対して ECHONET Lite フレームを送信することを言う。一斉同報送信とは、ECHONET Lite サブネット内において、レイヤ 4 以下のアドレスを用いて宛先を指定し、サブネット内の全ての ECHONET Lite ノードに対して ECHONET Lite フレームを送信することを言う。レイヤ 4 以下の下位通信層がマルチキャストやブロードキャストに対応していない場合は、ユニキャストにて、サブネット内に接続する ECHONET Lite 機器に送信することで、ECHONET Lite の一斉同報送信を実現してもよいものとする。ただし、ユニキャストの宛先、及びその設定方法は本規格では規定せず、使用する下位通信層毎に定めるものとする。

セキュリティは ECHONET 通信処理部では規定せず、Layer4 以下で既存の各種セキュリティ標準技術を必要に応じ適用することで、ECHONET Lite からは透過的にセキュリティを確保する。詳細は第 5 部 2．2 に示す。

ただし、Layer4 以下で下記プロトコルを使用する場合は、規定されたアドレスやポートをサポートすることが必須である。

（1）Layer4 で UDP(User Datagram Protocol)、Layer3 で IP(Internet Protocol)、を使用する場合

各 ECHONET Lite ノードは、それぞれ IP アドレスを持つ。IP アドレスの範囲、取得方法は規定しない。1 つの ECHONET Lite フレームは、1 つの UDP パケットにて転送する。UDP パケットにおける送信先 PORT 番号は、要求・応答・通知等の種別に関わらず、常に 3610 とする。送信元 PORT 番号は規定しない。また、ECHONET Lite フレームの一斉同報（一斉送信）は、IP マルチキャストパケットにマッピングして転送する。IPv4 の場合、送信先マルチキャストアドレス値は 224.0.23.0 とする。IPv6 の場合、ff02::1（オールノードマルチキャストアドレス）を用いるものとする。ただしいずれの場合も、OSI 参照モデル 4 層以下の仕様を他の規格団体が定めている仕様に準拠する場合は、該当する規格団体が定めるマルチキャストアドレスを使用する。ECHONET Lite ノードは、ポート 3610 にて、UDP ユニキャスト、および、マルチキャストのパケットを待ち受けるものとする。Layer4(UDP)、Layer3(IP)でのセキュリティ確保が必要な場合、ノードの認証には RFC5191、伝送フレームの Layer4(UDP)での暗号化および改ざん防止には DTLS、Layer3(IP)での暗号化および改ざん防止には IPSec などを用いる。

（2） Layer4 で TCP(Transmission Control Protocol)、Layer3 で IP(Internet Protocol)、を使用する場合

各 ECHONET Lite ノードは、それぞれ IP アドレスを持つ。IP アドレスの範囲、取得方法は規定しない。コネクション確立時は、TCP パケットにおける送信先 PORT 番号は、常に 3610 とする。コネクション確立後の送信先 PORT 番号は規定しない。また、送信元 PORT 番号は規定しない。要求電文に対する応答電文は同一のコネクションで送信するものとする。

ECHONET Lite フレームの一斉同報（一斉送信）は、Layer4 で UDP を使用し IP マルチキャストパケットにマッピングして転送する。IPv4 の場合、送信先マルチキャストアドレス値は 224.0.23.0 とする。IPv6 の場合、ff02::1（オールノードマルチキャストアドレス）を用いるものとする。ただしいずれの場合も、OSI 参照モデル 4 層以下の仕様を他の規格団体が定めている仕様に準拠する場合は、

該当する規格団体が定めるマルチキャストアドレスを使用する。なお、TCP に対応する ECHONET Lite ノードは、ポート番号 3610 にて、UDP ユニキャスト、および、UDP マルチキャストのパケットを待ち受け、必ずメッセージを受信し処理しなければならない。

２．３．２．基本的な考え方（２．１）

ここでは､ECHONET Lite で規定するオブジェクトは､次の２つと説明しています。

①．機器オブジェクト

②．プロファイルオブジェクト

２．３．３．機器オブジェクト（２．２）

規格書を抜粋して掲載します。

機器の持つ「機器としての動作機能」を機器オブジェクトとしてその詳細を規定する。機器オブジェクトは、機器相互で、通信を介しての制御や状態の確認を容易とすることを目的とするものである。機器オブジェクトのデータは通信ミドルウェア上に存在するが､機器としての動作機能本体はアプリケーションソフトウェア部に存在する。通信ミドルウェアでは、インスタンスのプロパティデータが管理され､そのプロパティの通信に関わる動作については ECHONET Lite 通信ミドルウェアにて管理・処理される。本規格においては､「機器オブジェクト」とは「家庭用エアコン」や「冷凍冷蔵庫」等の総称として用いる。また､「家庭用エアコン」や｢冷凍冷蔵庫｣等の各オブジェクトの仕様は､クラスとして別途個々にプロパティを規定する（｢APPENDIX　ECHONET 機器オブジェクト詳細規定」参照)。

機器オブジェクトは各クラスにて利用するプロパティを規定し、その内容およびプロパティに対するサービスを規定する。この関係を、図　2-1 に、具体的な例示と共に示す。

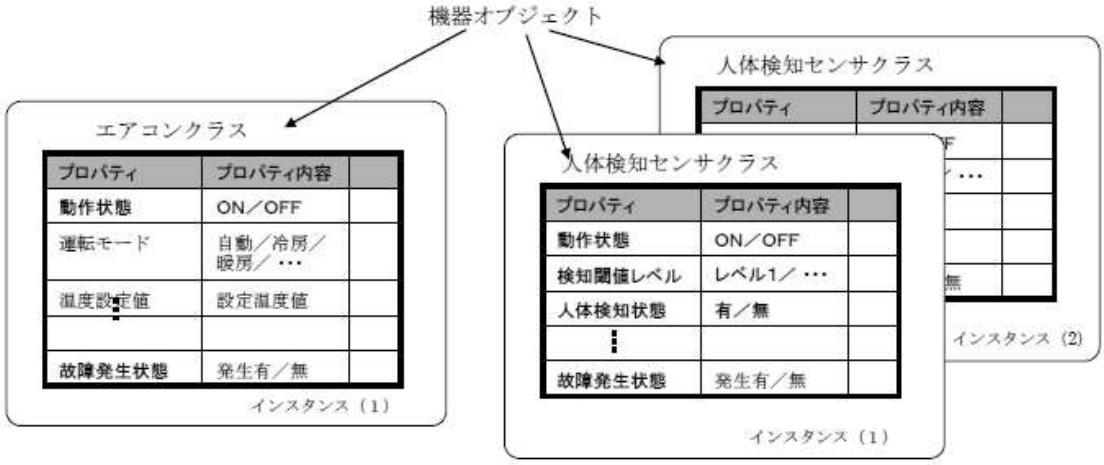


図　2-1　機器オブジェクト構成例図

図 2-1で示した機器オブジェクト（エアコン等）のクラス仕様（プロパティ構成等具体的な定義とコードの規定）については、「APPENDIX　ECHONET 機器オブジェクト詳細規定」にて示す。ECHONET Lite を介してこの ECHONET Lite ノードを制御したい他の ECHONET Lite ノードは、この機器オブジェクトを操作（書き込み／読み出し）することにより、この ECHONET Lite ノードの機能の制御や状態の確認を行うこととなる。

プロパティへの書き込みが行われた場合には、その値がアプリケーションソフトウェアに渡され処理される。実際に処理が実施されるかどうかは、書き込まれたプロパティ値およびアプリケーションの状態に依存する。

また、機器オブジェクトのプロパティ値は、対応するアプリケーションが現在保持する値を、「APPENDIX　ECHONET 機器オブジェクト詳細規定」に記載する各クラスの定義に従って読み出せるものとし、アプリケーションの機能に従い、ユーザの機器操作、機器内部の処理による自動制御、ECHONET Lite 通信による書き込みにより変化するものとする。

※機器オブジェクトは、規格書にあるように、Appendix に詳細な規定があり、収録される機器とそのプロパティは、随時、追加・更新されているので、機器の開発には、最新のものを参照することが必要です。

２．３．４．プロファイルオブジェクト（２．３）

規格書を抜粋して掲載します。

ECHONET Lite ノードの動作状態や、メーカ情報、機器オブジェクトリスト等 ECHONET Lite ノードとしてのプロファイルの情報を、アプリケーションソフトウェア及び他の ECHONET Lite ノードが操作（書き込み／読み出し）することを目的として規定するものである。本規格においては、「プロファイルオブジェクト」とは「ノードプロファイルオブジェクト」のプロファイルクラスの総称として用い、詳細は個々に規定する。プロファイルオブジェクトも、前頁図 2-1の機器オブジェクトと同様に、各クラスにて利用するプロパティを規定し、その内容およびプロパティに対するサービスを規定する（「APPENDIX　ECHONET 機器オブジェクト詳細規定」参照）。このプロファイルオブジェクトを操作（書き込み／読み出し）することにより、ECHONET Lite ノード（ノード）のプロファイルに関する操作を行う。

※プロファイルオブジェクトは、規格書にあるように、Appendix に詳細な規定があり、収録される機器とそのプロパティは、随時、追加・更新されているので、機器の開発には、最新のものを参照してください。

２．３．５．第３章　電文構成（フレームフォーマット）（３．１～３．２）

ECHONET Lite の規格書で最も重要な部分です。規格書該当部分を掲載します。

ここで解説している電文構成に従って、UDP メッセージを発行したり、またはネットワークを伝わる電文を解析したりすることができます。この電文構成を理解することは、ECHONET Lite 機器の開発に必須の条件です。

第３章 電文構成（フレームフォーマット）

３．１ 基本的な考え方

ECHONET Lite では、単純な機器の実装負荷を小さくしたいという状況を鑑み、通信のレイヤ構造上の仕様を満たしつつも、電文サイズを少しでも小さくすることを考慮して、ECHONET Lite 通信ミドルウェア部での電文構成を規定する。

３．２ 電文構成

ECHONET Lite 通信ミドルウェアにおいて処理される ECHONET Lite フレームの電文構成を図 3-1 に示す。電文の各構成要素の詳細仕様については、本節の以下の項で示す。

ECHONET Lite 通信処理部間でやり取りされる電文を、本規格では ECHONET Lite フレームとよぶ。ECHONET Lite フレームは、EHD(３．２．１項参照)の指定により、ECHONET Lite 規定の電文形式と、ユーザ独自の電文形式の 2 種類の形式に区別される。ECHONET Lite フレーム長は下位通信メディアに依存する。

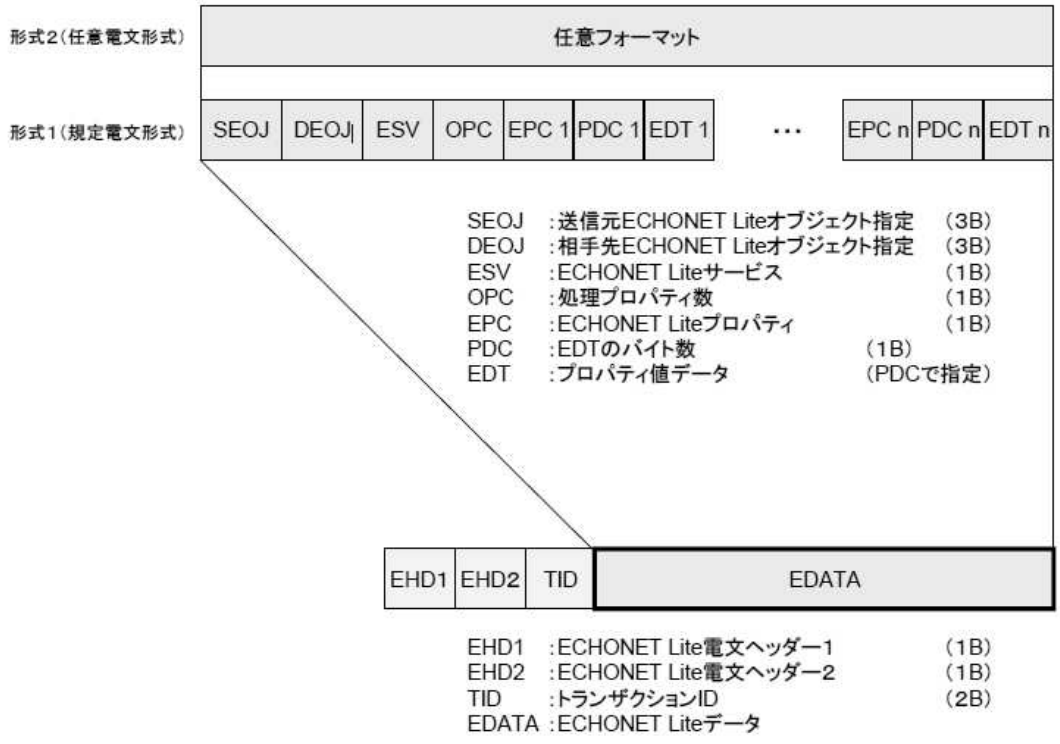


図 3-1　ECHONET Lite フレームの電文構成

３．２．１ ECHONET Lite ヘッダ（EHD）

EHD は ECHONET Lite ヘッダ 1 と ECHONET Lite ヘッダ 2 から構成される。

３.２.１.１ ECHONET Lite ヘッダ１（EHD１）

下図において、図 3-1 で示した ECHONET Lite ヘッダ 1（EHD1）の詳細規定を示す。

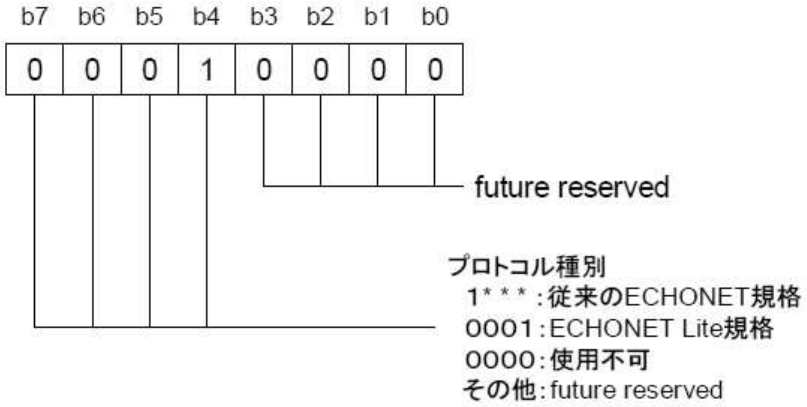


図 3-2 EHD1 詳細規定

b7~b4 の組み合わせは、ECHONET のプロトコル種別を指定する。b7:b6:b5:b4=0:0:0:1 は本仕様にて定義する ECHONET Lite プロトコルであることを示す。なお、b7:b6:b5:b4=0:0:0:0 は従来の ECHONET プロトコルとの共存を可能とするため、使用してはならない。

３.２.１.２ ECHONET Lite ヘッダ２（EHD２）

下図において、図 3-1 で示した ECHONET Lite ヘッダ 2（EHD2）の詳細規定を示す。

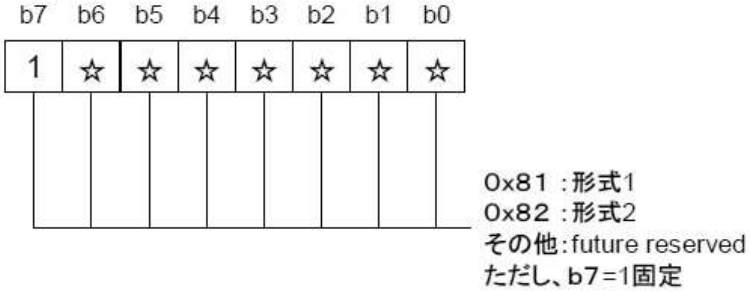


図 3-3 EHD2 詳細規定

EHD2 は、EDATA 部の電文形式を指定する。EHD2 が 0x81 の場合は、EDATA 部の電文形式が本仕様書にて定義する電文形式 1（規定電文形式）であることを示す。EHD2 が 0x82 の場合は、EDATA 部の電文形式が任意の形式となっている電文形式 2（任意電文形式）であることを示す。なお、従来の ECHONET プロトコルとの共存を可能とするため、b7=1

固定とする。

３．２．２ Transaction ID（TID）

ECHONET Lite 通信において、要求送信側が応答受信時に、自己が送信した要求と受信した応答をひも付けするためのパラメータである。応答送信側は、要求メッセージに含まれる値と同じ値を格納すること。プロパティ値通知など、応答受信を必要としないメッセージの TID の値については特に規定しない。

３．２．３ ECHONET Lite データ（EDATA）

ECHONET Lite 通信ミドルウェアにてやり取りされる電文のデータ領域。

３．２．４ ECHONET オブジェクト（EOJ）

図 3-1 で示した ECHONET オブジェクトコードの詳細規定を下図に示す。

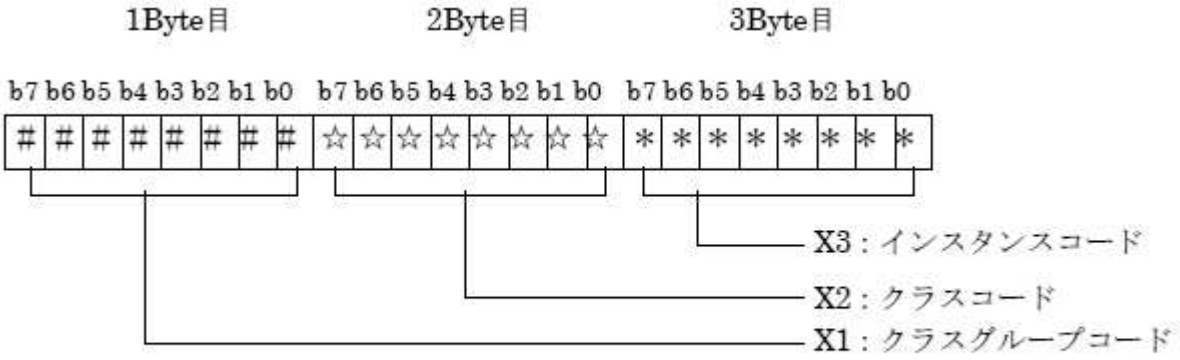


図 3-4　EOJ コードの詳細規定

ECHONET オブジェクトは、[X1. X2] [X3] の形式で表現することとし、それぞれ以下のように規定する。（但し、“.”は、単なる記述上の標記であり、具体的なコードを割り当てるものではない。）すなわち、X1、X2 の組み合わせによりオブジェクトのクラスを示し、X3 はそのクラスのインスタンスを示す。なお、1 つの ECHONET Lite ノードには同一のクラスのインスタンスが複数存在してもよいが、それを個々に識別する際に、この X3 を用いる。

具体的な表 3-2～8 中の項目は、JEM-1439 を活用し、規定した。ここに示すオブジェクトは、今後順次詳細規定を実施していくが、その規定の段階で、オブジェクト自体の規定（存在の有無）については見直しをかけていく。詳細（プロパティ構成まで）規定を実施したオブジェクトについては、備考欄に○をつけ、詳細規定は、「APPENDIX　ECHONET 機器オブジェクト詳細規定」にて示す。

インスタンスコード 0x00 を全インスタンス指定コードとし、指定されたクラスの全インスタンスを指定することを示す。

・X1　：クラスグループコード
　0x00～0xFF。具体的には、表 3-1 参照。
・X2　：クラスコード
　0x00～0xFF。具体的例は、表 3-2～表 3-8 参照。
・X3　：インスタンスコード

0x00～0x7F。[X1．X2]で属性規定されたものと同一のクラスが、複数、一つのノード内に存在する時の識別用コード。
但し、0x00 は、同一クラスのインスタンス全体の指定として使用。

表 3－1　クラスグループコード表

クラスグループコード	クラスグループ名	備考
0x00	センサ関連機器クラスグループ	
0x01	空調関連機器クラスグループ	
0x02	住宅・設備関連機器クラスグループ	
0x03	調理・家事関連機器クラスグループ	
0x04	健康関連機器クラスグループ	
0x05	管理・操作関連機器クラスグループ	
0x06	AV 関連機器クラスグループ	
0x07～0x0D	for future reserved	
0x0E	プロファイルクラスグループ	
0x0F	ユーザ定義クラスグループ	
0x10～0xFF	For future reserved	

表 3-2　クラスグループコード（X1=0x00）の時のクラスコード一覧表
詳細は、「APPENDIX　ECHONET 機器オブジェクト詳細規定」を参照のこと。

表 3-3　クラスグループコード（X1=0x01）の時のクラスコード一覧表
詳細は、「APPENDIX　ECHONET 機器オブジェクト詳細規定」を参照のこと。

表 3-4　クラスグループコード（X1=0x02）の時のクラスコード一覧表
詳細は、「APPENDIX　ECHONET 機器オブジェクト詳細規定」を参照のこと。

表 3-5　クラスグループコード（X1=0x03）の時のクラスコード一覧表
詳細は、「APPENDIX　ECHONET 機器オブジェクト詳細規定」を参照のこと。

表 3-6　クラスグループコード（X1=0x04）の時のクラスコード一覧表
詳細は、「APPENDIX　ECHONET 機器オブジェクト詳細規定」を参照のこと。

表 3-7　クラスグループコード（X1=0x05）の時のクラスコード一覧表

クラスコード	クラス名	詳細規定の有無	備考
0x00～0xFC	For　future　reserved		
0xFD	スイッチ		
0xFE	携帯端末		
0xFF	コントローラ		

表 3-8　クラスグループコード（X1=0x0E）の時のクラスコード一覧表

クラスコード	クラス名	詳細設定の有無	備考
0x00～0xEF	For future reserved		
0xF0	ノードプロファイル	●	本クラスの詳細規定は、第２部、６．１１．１項に記載
0xF1～0xFF	For future reserved		

３．２．５ ECHONET Lite サービス（ESV）

図 3-1 で示した ECHONET Lite サービスコードの詳細規定を示す。

注）b7:b6=0:1以外の時、b0～b5の意味付けは別規定となる。

図 3-5　ESV コードの詳細規定

本コードによるサービスは、EPC で指定されるプロパティに対する操作を指定するものである。ただし、操作の順序を規定するものではなく、どのプロパティから操作されていくかについては実装依存である。

操作として、以下の 3 種類を設ける。さらに「応答」として、EPC により指定された全てのプロパティに対してサービスが受理された場合の「応答」と、指定された複数のプロパティの 1 つ以上が存在しない場合、あるいは 1 つ以上のプロパティで指定のサービスが処理できない場合の「不可応答」を設ける。

「要求」・「応答」（応答／不可応答）・「通知」

「応答」は、応答を必要とする「要求」を受けての返信の位置付けとし、EOJ により指定されたオブジェクトが存在する場合には、「応答」か「不可応答」を返すものとする。EPC により指定された全てのプロパティに対してサービスが受理された場合は「応答」を、指定された 1 つ以上のプロパティで処理を受理できないか、或いは、オブジェクトは存在するが 1 つ以上のプロパティが存在しない場合は「不可応答」を返すものとする。応答不要な「要求」の場合、及び指定されたオブジェクトが存在しない場合には、「応答」は行わないものとする。

「通知」は、自発的に自プロパティの情報を送信するものと、通知要求の応答として送信するものが存在するが、コード上の区別は行わないものとする。

また、操作の具体的な内容として、「書き込み」（応答要求書き込み／応答不用書き込み）・「読み出し」・「書き込み、読み出し」、「通知」（通知／応答要通知）を設け、以下の 6 種類を設定する。

① プロパティ値書き込み（応答不要）
② プロパティ値書き込み（応答要）
③ プロパティ値読み出し
④ プロパティ値書き込み・読み出し
⑤ プロパティ値通知
⑥ プロパティ値通知（応答要）

前記した内容による ESV の具体的コードの割り付けを表 3-9～表 3-11に示す。

表 3-9 要求用 ESV コード一覧表

ｻｰﾋﾞｽｺｰﾄﾞ (ESV)	ECHONET Lite サービス内容	記号	備考
0x60	プロパティ値書き込み要求（応答不要）	SetI	一斉同報可
0x61	プロパティ値書き込み要求（応答要）	SetC	
0x62	プロパティ値読み出し要求	Get	一斉同報可
0x63	プロパティ値通知要求	INF_REQ	一斉同報可
0x64-0x6D	for future reserved		
0x6E	プロパティ値書き込み・読み出し要求	SetGet	一斉同報可
0x6F	for future reserved		

表 3-10 応答・通知用 ESV コード一覧表

ｻｰﾋﾞｽｺｰﾄﾞ (ESV)	ECHONET Lite サービス内容	記号	備考
0x71	プロパティ値書き込み応答	Set_Res	ESV=0x61 の応答、個別応答
0x72	プロパティ値読み出し応答	Get_Res	ESV=0x62 の応答、個別応答
0x73	プロパティ値通知	INF	*1 個別通知、一斉同報通知共に可
0x74	プロパティ値通知（応答要）	INFC	個別通知
0x75-0x79	for future reserved		
0x7A	プロパティ値通知応答	INFC_Res	ESV=0x74 の応答、個別応答
0x7B-0x7D	for future reserved		
0x7E	プロパティ値書き込み・読み出し応答	SetGet_Res	ESV=0x6E の応答、個別応答
0x7F	for future reserved		

注）＊1：自発的なプロパティ値通知、及び、0x63 の応答に使用。

表 3-11　不可応答用 ESV コード一覧表

ｻｰﾋﾞｽｺｰﾄﾞ (ESV)	ECHONET Lite サービス内容	記号	備考
0x50	プロパティ値書き込み要求不可応答	SetI_SNA	ESV=0x60 の不可応答、個別応答
0x51	プロパティ値書き込み要求不可応答	SetC_SNA	ESV=0x61 の不可応答、個別応答
0x52	プロパティ値読み出し不可応答	Get_SNA	ESV=0x62 の不可応答、個別応答
0x53	プロパティ値通知不可応答	INF_SNA	ESV=0x63 の不可応答、個別応答
0x54-0x5D	for future reserved		
0x5E	プロパティ値書き込み・読み出し不可応答	SetGet_SNA	ESV=0x6E の不可応答、個別応答
0x5F	for future reserved		

3．2．6 処理対象プロパティカウンタ（OPC、OPCSet、OPCGet）

処理対象プロパティカウンタは 1 バイトで構成される。ESV によるサービスがプロパティ値書き込み、プロパティ値読み出し、プロパティ値通知サービスの場合は、それぞれ書き込み対象、読み出し対象、通知対象となるプロパティの数を保持する。ESV によるが書き込み・読み出しサービスの場合は、書き込み対象プロパティ数を OPCSet に保持し、読み出し対象プロパティ数を OPCGet に保持する。
処理対象プロパティカウンタが取りうる最小値は 1 であり、最大値は下位通信メディアの送受信可能な電文長により制限される。ただし、SetGet_SNA の場合のみ処理対象プロパティカウンタの値は 0 となることがある。処理対象プロパティカウンタと以降の要求数または応答数が異なる ECHONET Lite フレームを受信したノードは、受信フレームを破棄する。
例として、図 3-6 のように要求が 3 の場合の処理対象プロパティカウンタは 0x03 となる。

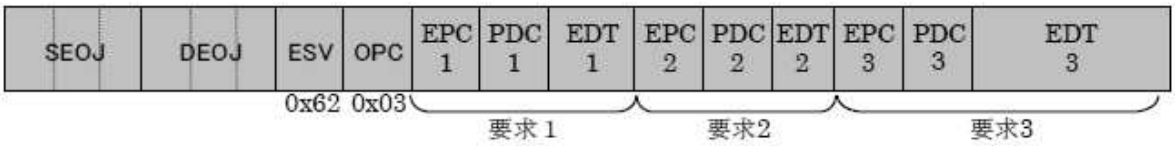


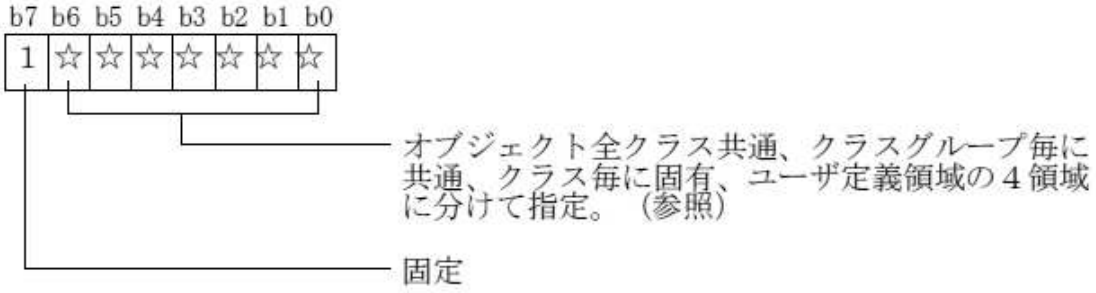
図 3-6　要求数が 3 の場合の処理対象プロパティカウンタ

3．2．7 ECHONET プロパティ（EPC）

図 3-1 で示した ECHONET プロパティ（EPC）コードの詳細規定を示す。EPC は、サービス対象機能を指定する。前項で示した X1（クラスグループコード）と X2（クラスコード）で指定されるオブジェクト毎に規定する。（同一コードでも、指定されるオブジェクトが異なると対象機能も異なることになるが、できる限り同

様の機能のものは、同じコードとなるように詳細は規定する。）オブジェクト毎の具体的なコード値の規定は、「APPENDIX　ECHONET 機器オブジェクト詳細規定」にて規定する。すなわち、本コードは、オブジェクト定義におけるオブジェクトプロパティの識別子に相当するものである。ただし、ECHONET Lite ノードでは、「APPENDIX　ECHONET 機器オブジェクト詳細規定」で規定されている配列要素 EPC をサポートしないものとする。

ここで、電文の構成と EPC、ESV の関連を示す。ECHONET Lite 電文での EPC は、ESV の値によって SEOJ 或いは DEOJ どちらの EOJ によって指定されるオブジェクトに関するものであるかが決まるものとする。ESV が「応答」或いは「通知」である場合には、EPC は SEOJ により指定されるオブジェクトを構成するものとし、DEOJ で指定されたオブジェクト宛ての「応答」或いは「通知」と見なす。ESV が「要求」である場合には、EPC は DEOJ を構成するものと見なし、SEOJ で指定されたオブジェクトからの「要求」と見なす。



注）b7=0 の場合、他のビットの意味付けは別規定となる。

図　3-7　EPC 詳細仕様

表 3-12　EPC コードの領域割り当て表

	8	9	A	B	C	D	E	F
0								
1								
2								
3								
4								
5								
6								
7	オブジェクト全クラスに共通となる領域		クラスグループ毎に共通となる領域*2		クラス毎に固有となる領域*2			ユーザ定義領域*1
8								
9								
A								
B								
C								
D								
E								
F								

←b7～b4 の値（16 進表示）

↑ b3～b0 の値（16 進表示）

注）＊1：ユーザ毎に規定。
ユーザ定義のオブジェクトクラスの場合、
b7～b4（上位 4 ビット）が、0xA～0xF は全てユーザ定義領域となる。
＊2：この二つの領域分けは原則とし、実際は、各クラスグループ毎に境界線の変更はあるものとする。個々の領域については、第 6 章と「APPENDIX　ECHONET 機器オブジェクト詳細規定」の具体的なオブジェクトクラス詳細仕様の中で規定する。

なお、0xF0～0xFF の領域は、各ユーザが独自に利用して良い領域である。

3．2．8 プロパティデータカウンタ（PDC）

ECHONET Lite データ（EDT）のバイト数を保持する。例えば、図 3-8 のように要求 1、要求 2、要求 3 の ECHONET Lite データのサイズがそれぞれ 2Byte、1Byte、5Byte の場合、1 番目のプロパティデータカウンタには 0x02 が、2 番目のプロパティデータカウンタには 0x01 が、3 番目のプロパティデータカウンタには 0x05 が入ることとなる。読み出し要求の場合は PDC の値は 0x00 である。

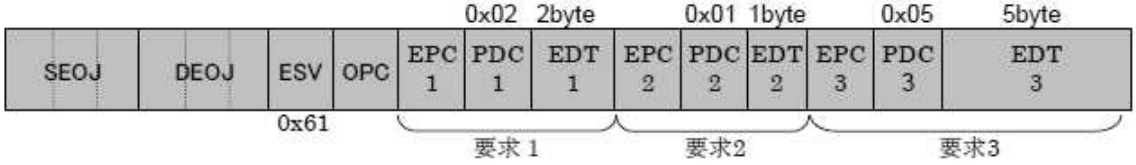


図 3-8　プロパティデータカウンタ

3．2．9　ECHONET プロパティ値データ（EDT）

図 3-1 で示した ECHONET プロパティ値データ（EDT）領域のコードの詳細規定を示す。EDT は、ECHONET Lite サービス（ESV）による具体的設定制御、或いは状態通知等サービス対象となる ECHONET プロパティ（EPC）のデータを示す。EDT は、EPC 毎にサイズ、コードの値等詳細が規定される（「APPENDIX ECHONET 機器オブジェクト詳細規定」参照）。

２章４節　（第５部）ECHONET Lite システム設計指針

第５部では、ECHONET Lite を実装した際に、不具合が出ないようにするための、きめ細かな配慮が盛り込まれています。平易な文章です。第１章を抜粋して掲載します。

２．４．１．第１章　ECHONET Lite の実装に関する指針（１．１～１．７）

第１章 ECHONET Lite の実装に関する指針

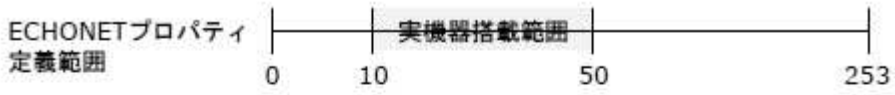
プラグフェストなどを通じて受けた仕様の解釈に関する問合せについて、以下第１章に、指針という形でまとめる。

１．１ プロパティ値の扱いに関する指針

本項では、設定されたプロパティ値が、ECHONET プロパティの定義範囲内であるが、対応する実機器の稼動範囲外である場合のプロパティ値の扱いについて指針を記す。

(1) ECHONET プロパティが対応する実機器の連続値の稼動範囲が、ECHONET プロパティ定義範囲より狭い場合に、ECHONET プロパティに、ECHONET プロパティの上限値および下限値の範囲内で、実機器の上限値および下限値の範囲外の値を設定した時、ECHONET Lite ノード上のアプリケーションは、ECHONET プロパティの上限値と、実機器の上限値との間の値を設定した場合は、実機器の上限値を実機器のプロパティ値及び ECHONET プロパティ値とすることを推奨する。また、ECHONET プロパティの下限値と、実機器の下限値との間の値を設定した場合は、実機器の下限値を実機器のプロパティ値及び ECHONET プロパティ値とすることを推奨する。

例えば、ECHONET プロパティ定義範囲が、0x00～0xFD（0℃～253℃）で、対応する実機器の値の稼動範囲が、0x0A～0x32（10℃～50℃）の場合に、ECHONET プロパティに、実機器の上限値と ECHONET プロパティの上限値との間の値（60℃）を設定した場合には、実機器の稼動範囲の上限値 0x32（50℃）を ECHONET プロパティ値とすることを推奨する。また、実機器の下限値と ECHONET プロパティの下限値との間の値（5℃）を設定した場合には、実機器の稼動範囲の下限値 0x0A（10℃）を ECHONET プロパティ値とすることを推奨する。
参考図を図 1-1 に示す。



•０～10を設定された場合：プロパティ値を10とすることを推奨する。

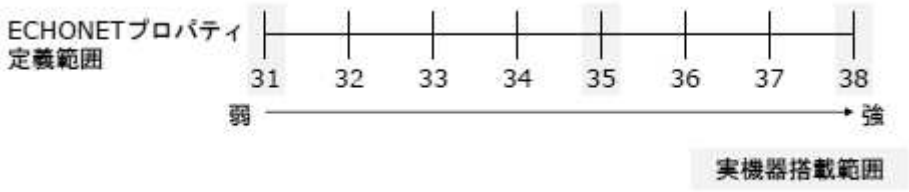
•10～50を設定された場合：プロパティ値は設定値とする。

•50～253を設定された場合：プロパティ値は50とすることを推奨する。

図 1-1　プロパティ値設定例1

(2) ECHONET プロパティが対応する実機器の値の稼動段階が、ECHONET プロパティ定義範囲より少ない場合に、ECHONET プロパティに、ECHONET プロパティの範囲内で、実機器が保持しない段階値を設定した場合に、ECHONET Lite ノード上のアプリケーションは、設定したプロパティ値に近い値を、実機器のプロパティ値およびECHONET プロパティ値とすることを推奨する。

例えば、ECHONET プロパティ定義範囲が、8 段階 0x31～0x38 で、対応する実機器の値の稼動範囲が、3 段階 0x31, 0x35, 0x38 の場合に、ECHONET プロパティに、8 段階のうち、いずれの値を設定した場合でも、ECHONET Lite ノード上のアプリケーションは、ECHONET プロパティ定義範囲が、8 段階 0x31～0x38 と、実機器の稼動範囲、3 段階 0x31, 0x35, 0x38 間のマッピングに従い、設定したプロパティ値に近い値を、実機器のプロパティ値およびECHONET プロパティ値とすることを推奨する。
参考図を図 1-2に示す。



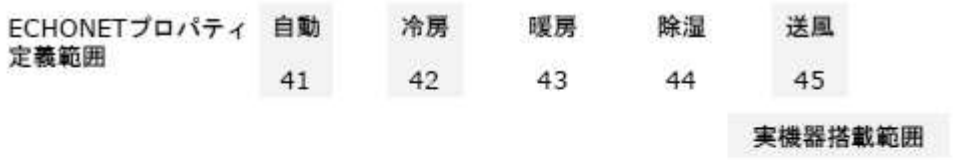
•31,35,38を設定された場合：プロパティ値は設定値とする。

•32,33,34,36,37を設定された場合：設定された値31,35,38の近い値にマッピングを行い、プロパティ値とする。

図 1-2　プロパティ値設定例2

(3) ECHONET プロパティが対応する実機器の値の実装機能が、ECHONET プロパティ定義範囲より少ない場合に、ECHONET プロパティに、ECHONET プロパティの範囲内で、実機器が保持しない段階値を設定した場合に、ECHONET Lite ノード上のアプリケーションは、設定したプロパティ値を無視することを推奨とし、現在の実機器のプロパティ値をECHONET プロパティ値とすることを推奨とする。

参考図を図 1-3に示す。



•41,42,45を設定された場合：プロパティ値は設定値とする。

•43,44を設定された場合：設定した値を無視し、プロパティ値はそのままとする。

図 1-3　プロパティ値設定例3

1．2 応答の扱いに関する指針

プロパティ値書き込み要求(ESV=0x60,61)およびプロパティ値書き込み読み出し要求(ESV=0x6E)の設定値として、以下のような値が指定された場合の応答については、処理を受理したものとして取り扱うことを推奨する。すなわち、ESV=0x60 の場合は、応答を行わない。また、ESV=0x61,0x6E の場合はプロパティ値書き込み応答(ESV=0x71)またはプロパティ値書き込み読み出し応答(ESV=0x7E)を返信する。

ただし、実機器における対応範囲の確認処理、または、実機器への設定処理などを実行してから応答を返信するように実装している場合は不可応答(ESV=0x50,0x51,0x5E)を返してもよいものとする。

- ECHONET プロパティ定義範囲外の値が指定された場合
- ECHONET プロパティが対応する実機器の連続値の稼動範囲が、ECHONET プロパティ定義範囲より狭い場合に、ECHONET プロパティに、ECHONET プロパティの上限値および下限値の範囲内で、実機器の上限値および下限値の範囲外の値が指定された場合
- ECHONET プロパティが対応する実機器の値の稼動段階が、ECHONET プロパティ定義範囲より少ない場合に、ECHONET プロパティに、ECHONET プロパティの範囲内で、実機器が保持しない段階値が指定された場合
- ECHONET プロパティが対応する実機器の値の実装機能が、ECHONET プロパティ定義範囲より少ない場合に、ECHONET プロパティに、ECHONET プロパティの範囲内で、実機器が保持しない段階値が指定された場合

1．3 OPC に関する指針

他の ECHONET Lite 機器に対してプロパティ値書き込み要求(ESV=0x60,0x61)、プロパティ値読み出し要求(ESV=0x62)、プロパティ値書き込み読み出し要求(ESV=0x6E)、プロパティ値通知要求(ESV=0x63)を送信する機器（コントローラなど）は、OPC に 2 以上の値 A を設定した要求送信に対して、A よりも小さい値 B が OPC に設定された不可応答を受信した場合、以降の要求送信では、OPC に値 B（以下）を設定することを推奨する。

なお、このとき制御対象機器の OPC 処理可能数がわからない場合、OPC に 1 を設定して要求電文（ESV=0x6*）を送信することで期待する応答の受信可能性が高まる。

１．４ 一斉同報に関する指針

一斉同報は、その使用方法によっては、ECHONET Lite ノードの処理過負荷やネットワーク輻輳発生を引き起こす可能性がある。一斉同報宛メッセージの送受信に関する指針を示す。

- 宛先が一斉同報宛、かつ、ESV がプロパティ値通知要求（0x63）であるメッセージの送信は、行わないことが望ましい。同メッセージは、それを受信した全てのノードからの、応答による更なる一斉同報宛送信を引き起こす。

- 宛先が一斉同報宛、かつ、ESV がプロパティ値書き込み要求（応答要）（0x61）、読み出し要求（0x62）、通知要求（0x63）、書き込み・読み出し要求（0x6E）のいずれかであるメッセージを受信したノードは、送信元への応答集中を緩和するため、その応答メッセージの送信までに、ノードごとに異なる時間だけ待つのが望ましい。待つ時間は、例えば、ノードごとに異なる固定値、または、ランダム時間を用いる。

- 各ノードのアプリケーションは、一斉同報による通知や、通知要求を使用する場合、システムや通信メディアの特性を考慮した上でトラフィックに大きな影響がないように使用することが望ましい。例えば、マルチホップを行う通信メディアである場合、多量の一斉同報メッセージを送信するとネットワーク負荷が高まり、システム全体の通信信頼性が落ちることがある。このため、予想されるトラフィックを考慮して送信頻度を設計することを推奨する。

１．５ インスタンス数に関する指針

一つの ECHONET Lite ノードが保持する機器オブジェクトのインスタンス数は、原則 84 個以下とする。

実体として 85 個以上のインスタンスを持つノードであっても、84 個以下のインスタンスのみを解釈するノードの存在を考慮し、インスタンスリスト通知(EPC=0xD5)では 84 個までのインスタンスのみを通知するのが望ましい。

85 個以上のインスタンスを含めたインスタンスリスト通知を行いたい、または、それを解釈したい場合、OPC 値を 2 以上、各 EPC 値を 0xD5 とした、一つのメッセージを用いるのが望ましい。ただし、メッセージの送信ノードは、それを解釈しない受信ノードがいることに留意すること。

１．６ プロパティ値書き込み・読み出しサービスに関する指針

プロパティ値書き込み・読み出しサービスの実装指針を示す。

- 他ノードから、プロパティ値書き込み・読み出し要求（ESV=0x6E）を受信するノードは、どのようなプロパティの組み合わせに対しても、まず書き込み要求処理を行い、その完了後の自ノードの状態に基づいた値を、読み出し要求への応答として格納するよう、実装する。任意のプロパティの組み合わせに対し、この順番での処理が保証できない（書き込み処理と機器状態変化とが非同期であるなど）ノードの場合、プロパティ値書き込み・読み出し要求に対して、OPCSet に 0、OPCGet に 0 を格納した不可応答（ESV=0x5E）を返すよう、実装する。

- ある相手ノードに対し、プロパティ値書き込み・読み出し要求（ESV=0x6E）を送信するノードは、その正常応答（ESV=0x7E）を受信した場合、相手ノードへの書き込み要求処理が完了し、かつ、完了後の相手ノードの状態に基づいた値が読み出し要求への応答として得られている、と見なして処理を行うよう、実装する。

１．７ 電文の送信に関する指針

ECHONET Lite 機器には、設備機器、センサ類などのようにメモリ容量が小さく演算処理能力が低い機器が多く存在する。このため、同一の ECHONET Lite 機器に対して短時間で連続して要求電文や通知電文の送信を行う場合や、ECHONET Lite 機器が応答を送信する前に新たな要求電文を同一 ECHONET Lite 機器に送信する場合は、該当する ECHONET Lite 機器からの応答が無くなったり、各電文に対する処理が反映されないなどの恐れがある。ECHONET Lite 機器の中には、秒オーダーかそれ以上の間隔をあける必要がある機器も存在する。同一の ECHONET Lite 機器に対して連続して電文を送信する際、多様な ECHONET Lite 機器の処理能力を考慮して送信間隔を設計することを推奨する。

１．８ ECHONET Lite 機器の管理に関する指針

同一の ECHONET Lite ノード上には、2 つ以上の機器オブジェクトを搭載することが可能である。このとき、ECHONET Lite ノードを一意に特定するだけでなく、同一の ECHONET Lite ノード上に搭載している機器単位で、一意に特定したい場合には、機器オブジェクトに「識別番号プロパティ（0x83）」を搭載し、下位通信層のプロトコル種別を 0xFE とする事を推奨する。ここで、「識別番号プロパティ（0x83）」は機器オブジェクトをドメイン内で一意に識別するための番号であり、「APPENDIX ECHONET 機器オブジェクト詳細規定」の機器オブジェクトスーパークラスに定義している。

２章５節　Appendix

この Appenndix はスマート家電開発に必須のデータブックです。Appendix は、最新の Release G で約 500 ページあります。ここでは、代表的な機器と面白そうなオブジェクトクラスの規定について抜粋して掲載します。この規格書で、ECHONET Lite 通信電文のどのフィールドに、どのようなプロパティ値が入るかが分かります。

後にモデル機器の開発を行おうとする方は、必ず、最新の Appendix をダウンロードして利用してください。

２．５．１．本書の概要（第１章）

本 APPENDIX においては、ECHONET オブジェクトのうち、機器オブジェクトに相当するクラスグループ（クラスグループコード 0x00～0x05）の機器オブジェクト及び、機器オブジェクトスーパークラスのプロパティ構成を詳細に示す。

機器オブジェクトに相当する各クラスは、機器オブジェクトスーパークラスのプロパティを継承する。したがって、各クラスを搭載する機器には、本 APPENDIX の各クラスで規定するプロパティと、機器オブジェクトスーパークラスのプロパティを搭載するものとする。

機器オブジェクト基本規定については「第 2 部 ECHONET 通信ミドルウェア仕様 第 9 章」及び、「第 2 部 ECHONET Lite 通信ミドルウェア仕様　第 7 章」を参照のこと。尚、ECHONET Lite 機器は、配列要素（SetM、GetM）として規定しているプロパティを搭載する事はできない。

プロパティ内容の値域に記載するコード全てに対応する機能を実機器が実装する必要はなく、実機器がその機能として有するコードのみを搭載することとする。

なお、機器オブジェクトの通信上の動作を設定できる「通信定義オブジェクト」が「第 2 部 ECHONET 通信ミドルウェア仕様　第 9 章」に規定されているので、参照のこと。

例えば、リモコンなど 機器側での操作を禁止もしくは制限する場合は、「ローカル変更制限設定 通信定義クラス」を利用する。

ただし、ECHONET Lite では、通信定義オブジェクトを定義していない為、対象外とする。

また、現時点では全ての機器に搭載するのは困難なプロパティであるが、サービス視点で考慮した際、機器への搭載を推奨するプロパティを「オプション必須」プロパティとして規定する。

本書での各クラス規定表内「必須」欄での本プロパティの表記は、表 1 に示した各アプリケーションサービス名の中で本プロパティ搭載によって実現されうるサービス名に対応した表記記号を使用する。

表１アプリケーションサービスと「オプション必須」プロパティ表記記号一覧

アプリケーション サービス名	サービス内容例	表記 記号
モバイルサービス (Mobile services)	・ 宅内機器運転状態遠隔モニター ・ 宅内機器遠隔操作、施錠操作 ・ 訪問者、高齢者生活状況遠隔モニター	Ⓜ
エネルギーサービス (Energy services)	・ 電気使用量、電気料金モニター ・ エアコン・換気扇・照明・ブラインド協調省エネ運転 ・ 契約電力デマンド制御	Ⓔ
快適生活支援サービス (Home amenity services)	・ ブラインド・換気扇・照明の集中操作 ・ 宅内機器スケジュール運転（予冷、予熱）	Ⓗa
ホームヘルスケア サービス (Home health-care services)	・ 健康管理サービス（病院、健康アドバイス会社） ・ 高齢者生活ケアサービス ・ 在宅医療機器監視・制御	Ⓗb
セキュリティサービス (Security services)	・ 防火（火災・ガス漏れ・漏電監視） ・ 防災（漏水検知、地震対応、凍結防止） ・ 防犯（訪問者管理、侵入者防止）	Ⓢ
機器リモート メンテナンスサービス (Remote appliance maintenance services)	・ 宅内機器遠隔故障診断・保守 ・ 宅内機器運転遠隔コンサルタント	Ⓡ

メーカ固有の機能を扱うことができるようにプロパティ内容にメーカオリジナルコードを設ける。但しプロパティ内容の欄にメーカオリジナルコードとして規定されている場合に限り、メーカ固有のコードを割り当てることができる。メーカオリジナルコードは、エコーネットオブジェクト規格のプロパティ内容（コード）として規定されていないプロパティ内容に対して用いることとする。メーカオリジナルコード内のコード設定はメーカごとに個々におこない、メーカオリジナルコード内のコードの追加／削除／変更についてもメーカとの判断でおこなうこととする。なお、メーカオリジナルコードに設定した内容について、公表するか否かについては、メーカごとの判断とする。

２．５．２．各機器抜粋

３．１ センサ関連機器クラスグループ

本節では、機器オブジェクトの内、センサ関連機器クラスグループ（クラスグループコード X1=0x00）に属する ECHONET オブジェクト毎に、コードやプロパティの詳細を規定する。本節で詳細を規定するクラスの一覧を、表５に示し、各クラス毎の詳細を項を設けて規定を示す。クラス規定において、「必須」の記述のあるものは、各クラスを搭載する機器には、そのプロパティとサービスの組み合わせの実装が必須であることを示す。

表５　センサ関連機器クラスグループのオブジェクト一覧表

クラスグループコード	クラスコード	クラス名	詳細規定の有無	備考
0x00	0x00	For future reserved		
	0x01	ガス漏れセンサ	○	
	0x02	防犯センサ	○	
	0x03	非常ボタン	○	
	0x04	救急用センサ	○	
	0x05	地震センサ	○	
	0x06	漏電センサ	○	
	0x07	人体検知センサ	○	
	0x08	来客センサ	○	
	0x09	呼び出しセンサ	○	
	0x0A	結露センサ	○	
	0x0B	空気汚染センサ	○	
	0x0C	酸素センサ	○	
	0x0D	照度センサ	○	
	0x0E	音センサ	○	
	0x0F	投函センサ	○	
	0x10	重荷センサ	○	
	0x11	温度センサ	○	
	0x12	湿度センサ	○	
	0x13	雨センサ	○	
	0x14	水位センサ	○	
	0x15	風呂水位センサ	○	
	0x16	風呂沸き上がりセンサ	○	
	0x17	水漏れセンサ	○	
	0x18	水あふれセンサ	○	
	0x19	火災センサ	○	
	0x1A	タバコ煙センサ	○	

	0x1B	ＣＯ２センサ	○	
	0x1C	ガスセンサ	○	
	0x1D	ＶＯＣセンサ	○	
	0x1E	差圧センサ	○	
	0x1F	風速センサ	○	
	0x20	臭いセンサ	○	
	0x21	炎センサ	○	
	0x22	電力量センサ	○	
	0x23	電流量センサ	○	
	0x24	昼光センサ		
	0x25	水流量センサ	○	
	0x26	微動センサ	○	
	0x27	通貨センサ	○	
	0x28	在床センサ	○	
	0x29	開閉センサ	○	
	0x2A	活動量センサ	○	
	0x2B	人体位置センサ	○	
	0x2C	雪センサ	○	
	0x2D	気圧センサ	○	
	0x2E～0xFF	For future reserved		

注）○：APPENDIX でプロパティ構成を含めた詳細を規定。

３．１．１７ 温度センサクラス規定

クラスグループコード　：　0x00
クラスコード　　：0x11
インスタンスコード　：0x01～0x7F (0x00：全インスタンス指定コード)

プロパティ名称	EPC	プロパティ内容 値域(10 進表記)	データ型	データサイズ	単位	アクセスルール	必須	状変時アナウンス	備考
動作状態	0x80	ON／OFF の状態を示す。 ＯＮ＝0x30，ＯＦＦ＝0x31	unsigned char	1 Byte	－	Set		○	
						Get	○		
温度計測値	0xE0	温度計測値を(0.1℃単位で)示す。 0xF554 ～ 0x7FFE(－ 2732 ～ 32766) (－273.2～3276.6℃)	signed short	2 Byte	0.1 ℃	Get	○		

注1)　状態変化時（状変時）アナウンスの○は、プロパティ実装時には、処理必須を示す。

(1)　動作状態（機器オブジェクトスーパークラスのプロパティを継承）

本クラス固有の機能が、稼動状態であるか否か(ON/OFF)を示す。尚、本クラスを搭載するノードにおいて、ノードの動作開始とともに、本クラスの機能が、稼動を開始する場合は、本プロパティを固定値 0x30（動作状態 ON）で実装することも可能である。

(2)　温度計測値

温度計測値を 0.1℃の単位で示す。プロパティの値域は、0xF554～0x7FFE(－273.2℃～3276.6℃)とし、実機器のプロパティ値が、プロパティの値域を超える場合は、オーバーフローコード 0x7FFF、実機器のプロパティ値が、プロパティの値域未満の場合は、アンダーフローコード 0x8000 を用いるものとする。

３．１．１８ 湿度センサクラス規定

クラスグループコード　：　0x00
クラスコード　　：0x12
インスタンスコード　：0x01～0x7F (0x00：全インスタンス指定コード)

プロパティ名称	EPC	プロパティ内容 値域(10 進表記)	データ型	データサイズ	単位	アクセスルール	必須	状変時アナウンス	備考
動作状態	0x80	ON／OFF の状態を示す。 ＯＮ＝0x30，ＯＦＦ＝0x31	unsigned char	1 Byte	－	Set Get	 ○	○	
相対湿度計測値	0xE0	相対湿度計測値を%単位で示す。 0x00～0x64(0～100%)	unsigned char	1 Byte	%	Get	○		

注1)　状態変化時（状変時）アナウンスの○は、プロパティ実装時には、処理必須を示す。

(1)　動作状態（機器オブジェクトスーパークラスのプロパティを継承）
本クラス固有の機能が、稼動状態であるか否か(ON/OFF)を示す。尚、本クラスを搭載するノードにおいて、ノードの動作開始とともに、本クラスの機能が、稼動を開始する場合は、本プロパティを固定値 0x30（動作状態 ON）で実装することも可能である。

(2)　相対湿度計測値
相対湿度計測値を%の単位で示す。プロパティの値域は、0x00～0x64(0～100%)とし、実機器のプロパティ値が、プロパティの値域を超える場合は、オーバーフローコード 0xFF、実機器のプロパティ値が、プロパティの値域未満の場合は、アンダーフローコード 0xFE を用いるものとする。

３．１．２２ 風呂沸き上がりセンサクラス規定

クラスグループコード　：　0x00
クラスコード　　：0x16
インスタンスコード　：0x01～0x7F (0x00：全インスタンス指定コード)

プロパティ名称	EPC	プロパティ内容 値域(10 進表記)	データ型	データサイズ	単位	アクセスルール	必須	状変時アナウンス	備考
動作状態	0x80	ON／OFF の状態を示す。 ＯＮ＝0x30，ＯＦＦ＝0x31	unsigned char	1 Byte	－	Set Get	 ○	○	
検知閾値レベル設定	0xB0	検知閾値レベルを８段階で指定。 レベル 0x31～0x38	unsigned char	1 Byte	－	Set/ Get			
風呂沸き上がり検知状態	0xB1	風呂沸き上がり検知有無を示す。 風呂沸き上がり有＝0x41 風呂沸き上がり無＝0x42	unsigned char	1 Byte	－	Get	○	○	

注1)　状態変化時（状変時）アナウンスの○は、プロパティ実装時には、処理必須を示す。

(1)　動作状態（機器オブジェクトスーパークラスのプロパティを継承）
本クラス固有の機能が、稼動状態であるか否か(ON/OFF)を示す。尚、本クラスを搭載するノードにおいて、ノードの動作開始とともに、本クラスの機能が、稼動を開始する場合は、本プロパティを固定値 0x30（動作状態 ON）で実装することも可能である。

(2)　検知閾値レベル設定
EPC＝0xB1「風呂沸き上がり検知状態」が有に遷移する閾値を 8 段階のレベルで設定する。0x31 を最小値、0x38 を最大値とするが、各レベルの具体的な値については、規定しない。また、実機器の検知閾値が 8 段階より少ない場合、もしくは、8 段階より多い場合も、必ず、本プロパティで規定する 8 段階のプロパティ値に実機器のプロパティを割り当てるものとする。

(3)　風呂沸き上がり検知状態
風呂沸き上がり検知状態の有、無状態を示す。EPC＝0xB0「検知閾値レベル設定」を実装する際は、検知閾値レベル設定で設定する閾値以上になった場合に、風呂沸き上がり検知状態有りに遷移し、閾値未満になった場合に、風呂沸き上がり検知状態無しに遷移するものとする。また、本プロパティが、風呂沸き上がり検知状態有り＝0x41 に遷移した際には、本プロパティを、定期的にアナウンスするものとする。

３．１．３４ 電力量センサクラス規定

クラスグループコード　：　0x00
クラスコード　　：0x22
インスタンスコード　：0x01～0x7F (0x00：全インスタンス指定コード)

プロパティ名称	EPC	プロパティ内容 値域(10 進表記)	データ型	データサイズ	単位	アクセスルール	必須	状変時ｱﾅｳﾝｽ	備考
動作状態	0x80	ON／OFF の状態を示す。 ＯＮ＝0x30，ＯＦＦ＝0x31	unsigned char	1 Byte	－	Set Get	 ○	○	
積算電力量計測値	0xE0	積算電力量を 0.001kWh で示す。 0x00000000～0x3B9AC9FF (0～999,999.999kWh)	unsigned long	4 Byte	0.001 kWh	Get	○		
中容量センサ瞬時電力値計測値	0xE1	瞬時電力値を W で示す。 0xC4653601 ～ 0x3B9AC9FF (－999,999,999～999,999,999)	signed long	4 Byte	W	Get			
小容量センサ瞬時電力値計測値	0xE2	瞬時電力値を 0.1W で示す。 0x8001～0x7FFE (－3276.7～3276.6)	signed short	2 Byte	0.1 W	Get			
大容量センサ瞬時電力値計測値	0xE3	瞬時電力値を 0.1kW で示す。 0x8001～0x7FFE (－3276.7～3276.6)	signed short	2 Byte	0.1 kW	Get			
積算電力量計測履歴情報	0xE4	積算電力量(0.001kWh)の計測結果履歴を、30 分毎データを過去 24 時間で示す。 0 ～ 0x3B9AC9FF (0 ～ 999,999,999) (0～999,999.999kWh)	unsigned long ×48	192 Byte	0.001 kWh	Get			
実効電圧値計測値	0xE5	実効電圧値を V で示す。 0x0000～0xFFFD (0～65533V)	unsigned short	2 Byte	V	Get			

注1)　状態変化時（状変時）アナウンスの○は、プロパティ実装時には、処理必須を示す。

(1)　動作状態（機器オブジェクトスーパークラスのプロパティを継承）

本クラス固有の機能が、稼動状態であるか否か(ON/OFF)を示す。尚、本クラスを搭載するノードにおいて、ノードの動作開始とともに、本クラスの機能が、稼動を開始する場合は、本プロパティを固定値 0x30（動作状態 ON）で実装することも可能である。

(2)　積算電力量計測値

積算電力量計測値を 0.001kWh の単位で示す。プロパティの値域は、0x00000000～0x3B9AC9FF(0～999,999.999kWh）とし、積算電力量計測値のオーバーフロー時は、0x00000000 から再インクリメントするものとする。

(3) 小容量センサ瞬時電力値計測値

小容量センサの瞬時電力値の計測値を 0.1W の単位で示す。プロパティの値域は、0x8001～0x7FFE(－3276.7～3276.6 W)とし、実機器のプロパティ値が、プロパティの値域を超える場合は、オーバーフローコード 0x7FFF、実機器のプロパティ値が、プロパティの値域未満の場合は、アンダーフローコード 0x8000 を用いるものとする。

(4) 中容量センサ瞬時電力値計測値

中容量センサの瞬時電力値の計測値を W の単位で示す。プロパティの値域は、0xC4653601～0x3B9AC9FF（－999,999,999～999,999,999W)とし、実機器のプロパティ値が、プロパティの値域を超える場合は、オーバーフローコード 0x7FFFFFFF、実機器のプロパティ値が、プロパティの値域未満の場合は、アンダーフローコード 0x80000000 を用いるものとする。

(5) 大容量センサ瞬時電力値計測値

大容量センサの瞬時電力値の計測値を 0.1kW の単位で示す。プロパティの値域は、0x8001～0x7FFE(－3276.7～3276.6 kW)とし、実機器のプロパティ値が、プロパティの値域を超える場合は、オーバーフローコード 0x7FFF、実機器のプロパティ値が、プロパティの値域未満の場合は、アンダーフローコード 0x8000 を用いるものとする。

(6) 積算電力量計測履歴情報

積算電力量(0.001kWh)の計測結果履歴情報の 30 分毎データを過去 24 時間分データで示す。30 分毎の積算電力量計測値は、プロパティ名称「時刻設定値」(EPC=0x97)で設定する時刻に基づき、毎 0 分、30 分の 0.001kWh 単位の計測値を、0x00000000～0x3B9AC9FF(0～999,999.999kWh）のデータとし、時系列に上位バイトから順にプロパティ値とするものとする。

(7) 実効電圧値計測値

電力量センサの実効電圧計測値を V の単位で示す。本プロパティは、計測定格電圧として固定値で実装してもよいものとする。

３．２ 空調関連機器クラスグループ

本節では、機器オブジェクトの内、空調関連機器クラスグループ（クラスグループコード X1=0x01）に属する ECHONET オブジェクト毎に、コードやプロパティの詳細を規定する。本節で詳細を規定するクラスの一覧を、表６に示し、各クラス毎の詳細を項を設けて規定を示す。クラス規定において、「必須」の記述のあるものは、各クラスを搭載する機器には、そのプロパティとサービスの組み合わせの実装が必須であることを示す。

表６　空調関連機器クラスグループのオブジェクト一覧表

クラス グループコード	クラスコード	クラス名	詳細規定の有無	備考
0x01	0x00～0x2F	For　future　reserved		
	0x30	家庭用エアコン	○	
	0x31	冷風機		
	0x32	扇風機		
	0x33	換気扇	○	
	0x34	空調換気扇	○	
	0x35	空気清浄器	○	
	0x36	冷風扇		
	0x37	サーキュレータ		
	0x38	除湿機		
	0x39	加湿器	○	
	0x3A	天井扇		
	0x3B	電気こたつ		
	0x3C	電気あんか		
	0x3D	電気毛布		
	0x3E	ストーブ		
	0x3F	パネルヒータ		
	0x40	電気カーペット		
	0x41	フロアヒータ		
	0x42	電気暖房器	○	
	0x43	ファンヒータ	○	
	0x44	充電器		
	0x45	業務用パッケージエアコン室内機	○	
	0x46	業務用パッケージエアコン室外機	○	
	0x47	業務用パッケージエアコン蓄熱ユニット		
	0x48	業務用ファンコイルユニット		
	0x49	業務用空調冷熱源(チラー)		
	0x50	業務用空調温熱源(ボイラー)		
	0x51	業務用空調 VAV		
	0x52	業務用空調エアハンドリングユニット		
	0x53	ユニットクーラー		
	0x54	業務用コンデンシングユニット		
	0x55	電気蓄熱暖房器	○	
	0x55～0xFF	For　future　reserved		

注）○：APPENDIX でプロパティ構成を含めた詳細を規定。

クラスグループコード　　：0x01
クラスコード　　：0x30
インスタンスコード　：0x01～0x7F (0x00：全インスタンス指定コード)

プロパティ名称	EPC	プロパティ内容 値域(10 進表記)	データ型	データサイズ	単位	アクセスルール	必須	状変時アナウンス	備考
動作状態	0x80	ON／OFF の状態を示す。 ＯＮ＝0x30，ＯＦＦ＝0x31	unsigned char	1 Byte	－	Set Get	○ ○	○	
節電動作設定	0x8F	機器の節電動作を設定し、状態を取得する。 節電動作中=0x41 通常動作中=0x42	unsigned char	1 Byte	－	Set/ Get	○	○	
運転モード設定	0xB0	自動／冷房／暖房／除湿／送風／その他の運転モードを設定し、設定状態を取得する。 順番に以下のコードが対応。 0x41/0x42/0x43/0x44/0x45/0x40	unsigned char	1 Byte	－	Set/ Get	○	○	
温度自動設定	0xB1	AUTO／非 AUTO を設定し、設定状態を取得する。 AUTO＝0x41,非 AUTO＝0x42	unsigned char	1 Byte	－	Set/ Get			
急速動作モード設定	0xB2	通常運転／急速／静音を設定し、設定状態を取得する。 通常運転＝0x41 急速＝0x42，　静音＝0x43	unsigned char	1 Byte	－	Set/ Get			
温度設定値	0xB3	温度設定値を設定し、設定状態を取得する。 0x00～0x32（0～50℃）	unsigned char	1 Byte	℃	Set/ Get	○		
除湿モード時相対湿度設定値	0xB4	除湿モード時相対湿度設定値を設定し、設定状態を取得する。 0x00～0x64, (0～100%)	unsigned char	1 Byte	%	Set/ Get			
冷房モード時温度設定値	0xB5	冷房モード時設定温度値を設定し、設定状態を取得する。 0x00～0x32（0～50℃）	unsigned char	1 Byte	℃	Set/ Get			
暖房モード時温度設定値	0xB6	暖房モード時設定温度値を設定し、設定状態を取得する。 0x00～0x32（0～50℃）	unsigned char	1 Byte	℃	Set/ Get			
除湿モード時温度設定値	0xB7	除湿モード時設定温度値を設定し、設定状態を取得する。 0x00～0x32（0～50℃）	unsigned char	1 Byte	℃	Set/ Get			
定格消費電力値	0xB8	冷房／暖房／除湿／送風の各運転モード時の定格消費電力 0x0000～0xFFFD (0～65533W) 冷房：暖房：除湿：送風	unsigned short ×4	8 Byte	W	Get			
消費電流計測値	0xB9	消費電流計測値 0x0000～0xFFFD (0～6553.3A)	unsigned short	2 Byte	0.1 A	Get			
室内相対湿度計測値	0xBA	室内相対湿度計測値 0x00～0x64 (0～100%)	unsigned char	1 Byte	%	Get			

３．２．１ 家庭用エアコンクラス規定

クラスグループコード　　：0x01
クラスコード　　：0x30
インスタンスコード　：0x01～0x7F (0x00：全インスタンス指定コード)

プロパティ名称	EPC	プロパティ内容 値域(10 進表記)	データ型	データサイズ	単位	アクセスルール	必須	状変時アナウンス	備考
動作状態	0x80	ON／OFF の状態を示す。 ＯＮ＝0x30，ＯＦＦ＝0x31	unsigned char	1 Byte	－	Set Get	○ ○	○	
節電動作設定	0x8F	機器の節電動作を設定し、状態を取得する。 節電動作中=0x41 通常動作中=0x42	unsigned char	1 Byte	－	Set/ Get	○	○	
運転モード設定	0xB0	自動／冷房／暖房／除湿／送風／その他の運転モードを設定し、設定状態を取得する。 順番に以下のコードが対応。 0x41/0x42/0x43/0x44/0x45/0x40	unsigned char	1 Byte	－	Set/ Get	○	○	
温度自動設定	0xB1	AUTO／非 AUTO を設定し、設定状態を取得する。 AUTO＝0x41,非 AUTO＝0x42	unsigned char	1 Byte	－	Set/ Get			
急速動作モード設定	0xB2	通常運転／急速／静音を設定し、設定状態を取得する。 通常運転＝0x41 急速＝0x42，静音＝0x43	unsigned char	1 Byte	－	Set/ Get			
温度設定値	0xB3	温度設定値を設定し、設定状態を取得する。 0x00～0x32 (0～50℃)	unsigned char	1 Byte	℃	Set/ Get	○		
除湿モード時相対湿度設定値	0xB4	除湿モード時相対湿度設定値を設定し、設定状態を取得する。 0x00～0x64, (0～100%)	unsigned char	1 Byte	%	Set/ Get			
冷房モード時温度設定値	0xB5	冷房モード時設定温度値を設定し、設定状態を取得する。 0x00～0x32 (0～50℃)	unsigned char	1 Byte	℃	Set/ Get			
暖房モード時温度設定値	0xB6	暖房モード時設定温度値を設定し、設定状態を取得する。 0x00～0x32 (0～50℃)	unsigned char	1 Byte	℃	Set/ Get			
除湿モード時温度設定値	0xB7	除湿モード時設定温度値を設定し、設定状態を取得する。 0x00～0x32 (0～50℃)	unsigned char	1 Byte	℃	Set/ Get			
定格消費電力値	0xB8	冷房／暖房／除湿／送風の各運転モード時の定格消費電力 0x0000～0xFFFD (0～65533W) 冷房：暖房：除湿：送風	unsigned short ×4	8 Byte	W	Get			
消費電流計測値	0xB9	消費電流計測値 0x0000～0xFFFD (0～6553.3A)	unsigned short	2 Byte	0.1 A	Get			
室内相対湿度計測値	0xBA	室内相対湿度計測値 0x00～0x64 (0～100%)	unsigned char	1 Byte	%	Get			

室内温度計測値	0xBB	室内温度計測値 0x81～0x7D (－127～125℃)	signed char	1 Byte	℃	Get	○		
ユーザリモコン温度設定値	0xBC	ユーザリモコン温度設定値 0x00～0x32 (0～50℃)	unsigned char	1 Byte	℃	Get			
吹き出し温度計測値	0xBD	吹き出し温度計測値 0x81～0x7D (－127～125℃)	signed char	1 Byte	℃	Get			
外気温度計測値	0xBE	外気温度計測値 0x81～0x7D(－127～125℃)	signed char	1 Byte	℃	Get			
相対温度設定値	0xBF	エアコン動作中､動作モードにおける目標温度値に対する相対温度設定値を設定し､設定内容を取得する。 0x81～0x7D(-12.7℃～12.5℃)	signed char	1 Byte	0.1 ℃	Set /Get			
風量設定	0xA0	風量レベルおよび風量自動状態を設定し、設定状態を取得する。風量レベルは８段階で指定。 風量自動設定＝0x41 風量レベル＝0x31～0x38	unsigned char	1 Byte	－	Set /Get	○	○	
風向自動設定	0xA1	風向き上下左右の AUTO／非 AUTO を設定し、設定状態を取得する。 AUTO=0x41、非 AUTO=0x42 上下 AUTO=0x43、左右 AUTO=0x44	unsigned char	1 Byte	－	Set /Get			
風向スイング設定	0xA3	風向スイング OFF／上下／左右／上下左右を設定し､設定状態を取得する。 風向スイング OFF＝0x31、 上下＝0x41、左右＝0x42、 上下左右＝0x43	unsigned char	1 Byte	－	Set /Get			
風向上下設定	0xA4	上下方向の風向きを 5 通りのパターンで設定し､設定状態を取得する。 上=0x41、下=0x42、中央=0x43、上中=0x44、下中=0x45	unsigned char	1 Byte	－	Set /Get			
風向左右設定	0xA5	左右方向の風向きを31通りのパターンで設定し、設定状態を取得する。 右＝0x41、左＝0x42、 中央＝0x43、左右＝0x44 他、プロパティ詳細説明の表に記載のコードで示す。	unsigned char	1 Byte	－	Set /Get			
特殊状態	0xAA	エアコンが特殊状態にあることを示す。 通常状態=0x40、除霜状態=0x41 予熱状態=0x42、排熱状態=0x43	unsigned char	1 Byte	－	Get			
非優先状態	0xAB	エアコンが非優先状態にあることを示す 通常状態=0x40 非優先状態=0x41	unsigned char	1 Byte	－	Get			
換気モード設定	0xC0	換気の動作(方向)を設定し、設定状態を取得する。	unsigned char	1 Byte	－	Set /Get			

		換気 ON（排気方向）=0x41、 換気 OFF=0x42、 換気 ON（吸気方向）=0x43							
加湿モード設定	0xC1	加湿のモード設定 ON/OFF を設定し、設定状態を取得する。 加湿 ON＝0x41,OFF＝0x42	unsigned char	1 Byte	—	Set /Get			
換気風量設定	0xC2	換気風量レベルを設定し、設定状態を取得する。 換気風量自動=0x41 換気風量レベル=0x31～0x38	unsigned char	1 Byte	—	Set /Get			
加湿量設定	0xC4	加湿量レベルを設定し、設定状態を取得する。 加湿量自動=0x41 加湿量レベル=0x31～0x38	unsigned char	1 Byte	—	Set /Get			
搭載空気清浄方法	0xC6	空気清浄機能を実現するために搭載されている方法をビットマップで示す。 ビット 0：電気集塵方式搭載情報 0 非搭載 1 搭載 ビット 1：クラスタイオン方式搭載情報 0 非搭載 1 搭載	unsigned char	1 Byte	—	Get			
空気清浄機能モード設定	0xC7	8 バイトの配列で、実現方法ごとの空気清浄機能の ON/OFF およびその制御レベルを設定し、設定状態を取得する。 第 0 要素： 電気集塵方式による空気清浄機能の設定状態 第 1 要素： クラスタイオン方式による空気清浄機能の設定状態 第 2 要素～第 7 要素： for future reserved	unsigned char ×8	1Byte × 8	—	SetM /GetM Set /Get			
搭載リフレッシュ方法	0xC8	リフレッシュ機能を実現するために搭載されている方法をビットマップで示す。 ビット 0：マイナスイオン方式搭載情報 0 非搭載 1 搭載 ビット 1：クラスタイオン方式搭載情報 0 非搭載 1 搭載	unsigned char	1 Byte	—	Get			
リフレッシュ機能モード設定	0xC9	8 バイトの配列で、実現方法ごとのリフレッシュ機能の ON/OFF およびその制御レベルを設定し、設定状態を取得する。	unsigned char ×8	1 Byte ×	—	SetM /GetM Set			

		第0要素： マイナスイオン方式によるリフレッシュ機能の設定状態 第1要素： クラスタイオン方式によるリフレッシュ機能の設定状態 第2要素～第7要素： for future reserved		8		/Get			
搭載自己洗浄方法	0xCA	自己洗浄機能を実現するために搭載されている方法をビットマップで示す。 ﾋﾞｯﾄ0：オゾン洗浄方式搭載情報 0 非搭載 1 搭載 ﾋﾞｯﾄ1：乾燥方式搭載情報 0 非搭載 1 搭載	unsigned char	1 Byte	－	Get			
自己洗浄機能モード設定	0xCB	8バイトの配列で、実現方法ごとの自己洗浄機能のON/OFFおよびその制御レベルを設定し、設定状態を取得する。 第0要素： オゾン洗浄方式による自己洗浄機能の設定状態 第1要素： 乾燥方式による自己洗浄機能の設定状態 第2要素～第7要素： for future reserved	unsigned char ×8	1Byte × 8	－	SetM /GetM Set /Get			
特別運転モード設定	0xCC	特別運転モードを設定し、設定状態を取得する。 設定なし：0x40 衣類乾燥：0x41 結露抑制：0x42 ダニカビ抑制：0x43 強制除霜 0x44 0x45～ for future reserved	unsigned char	1 Byte	－	Set /Get			
内部動作状態	0xCD	エアコンの内部動作状態をビットマップで表現する。 ﾋﾞｯﾄ0：コンプレッサ動作状態 0 停止中 1 動作中 ﾋﾞｯﾄ1：サーモON/OFF状態 0 サーモＯＦＦ状態 1 サーモＯＮ状態 ビット2～ビット7： for future reserved	unsigned char	1 Byte	－	Get			
強制サーモモード設定	0xCE	エアコンのサーモ設定を無視して運転するか否かを設定する。 通常設定=0x40, 強制サーモON=0x41, 強制サーモOFF=0x42	unsigned char	1 Byte	－	Set / Get			
空気清浄モード設定	0xCF	空気清浄のモード設定 ON/OFFを設定し、設定状態を取得する。	unsigned char	1 Byte	－	Set /Get			

		第0要素： マイナスイオン方式によるリフレッシュ機能の設定状態 第1要素： クラスタイオン方式によるリフレッシュ機能の設定状態 第2要素～第7要素： for future reserved		8		/Get			
搭載自己洗浄方法	0xCA	自己洗浄機能を実現するために搭載されている方法をビットマップで示す。 ﾋﾞｯﾄ0：オゾン洗浄方式搭載情報 0 非搭載 1 搭載 ﾋﾞｯﾄ1：乾燥方式搭載情報 0 非搭載 1 搭載	unsigned char	1 Byte	－	Get			
自己洗浄機能モード設定	0xCB	8バイトの配列で、実現方法ごとの自己洗浄機能のON/OFFおよびその制御レベルを設定し、設定状態を取得する。 第0要素： オゾン洗浄方式による自己洗浄機能の設定状態 第1要素： 乾燥方式による自己洗浄機能の設定状態 第2要素～第7要素： for future reserved	unsigned char ×8	1Byte × 8	－	SetM /GetM Set /Get			
特別運転モード設定	0xCC	特別運転モードを設定し、設定状態を取得する。 設定なし：0x40 衣類乾燥：0x41 結露抑制：0x42 ダニカビ抑制：0x43 強制除霜 0x44 0x45～ for future reserved	unsigned char	1 Byte	－	Set /Get			
内部動作状態	0xCD	エアコンの内部動作状態をビットマップで表現する。 ﾋﾞｯﾄ0：コンプレッサ動作状態 0 停止中 1 動作中 ﾋﾞｯﾄ1：サーモON/OFF状態 0 サーモＯＦＦ状態 1 サーモＯＮ状態 ビット2～ビット7： for future reserved	unsigned char	1 Byte	－	Get			
強制サーモモード設定	0xCE	エアコンのサーモ設定を無視して運転するか否かを設定する。 通常設定=0x40, 強制サーモON=0x41, 強制サーモOFF=0x42	unsigned char	1 Byte	－	Set / Get			
空気清浄モード設定	0xCF	空気清浄のモード設定 ON/OFFを設定し、設定状態を取得する。	unsigned char	1 Byte	－	Set /Get			

		空気清浄 ON＝0x41,OFF＝0x42							
ON タイマ 予約設定	0x90	予約入／予約切を設定し、設定状態を取得する。 時刻予約,相対時間予約共に入＝0x41,予約切＝0x42, 時刻予約のみ入り＝0x43, 相対時刻予約のみ入り＝0x44	unsigned char	1 Byte	－	Set /Get			
ON タイマ 時刻設定値	0x91	タイマ値 HH:MM を設定し、設定状態を取得する。 0～0x17: 0～0x3B (=0～23):(=0～59)	unsigned char ×2	2 Byte	－	Set /Get			
ON タイマ相対時間設定値	0x92	タイマ値 HH:MM を設定し、更新された時間を取得する。 0～0xFF: 0～0x3B (=0～255):(=0～59)	unsigned char ×2	2 Byte	－	Set /Get			
OFF タイマ 予約設定	0x94	予約入／予約切を設定し、設定内容を取得する。 時刻予約,相対時間予約共に入＝0x41,予約切＝0x42, 時刻予約のみ入り＝0x43, 相対時刻予約のみ入り＝0x44	unsigned char	1 Byte	－	Set /Get			
OFF タイマ 時刻設定値	0x95	タイマ値 HH:MM を設定し、設定状態を取得する。 0～0x17: 0～0x3B (=0～23):(=0～59)	unsigned char ×2	2 Byte	－	Set /Get			
OFF タイマ 相対時間設定値	0x96	タイマ値 HH:MM を設定し、更新された時間を取得する。 0～0xFF: 0～0x3B (=0～255):(=0～59)	unsigned char ×2	2 Byte	－	Set /Get			

注1)　状態変化時（状変時）アナウンスの○は、プロパティ実装時には、処理必須を示す。

(1)　動作状態（機器オブジェクトスーパークラスのプロパティを継承）

家庭用エアコンの運転／停止を設定し、動作状態を取得する。運転／停止にそれぞれ、0x30/0x31 のプロパティ値が対応するものとする。

(2)　節電動作設定(機器オブジェクトスーパークラスのプロパティを継承)

家庭用エアコンの通常モード(非節電モード)／節電動作モードを設定し動作状態を取得する。節電動作モード／通常動作モード(非節電モード)に、それぞれ 0x41/0x42 のプロパティ値が対応する。

生活者の見守りサービスのために、状変時アナウンスを必須とする。

(3)　運転モード設定

家庭用エアコンの自動／冷房／暖房／除湿／送風／その他の各運転モードを設定し、設定状態を取得する。「その他」というモードは、他のいずれの運転モードにも該当しない運転モードである。それぞれの運転モードにはそれぞれ、順に 0x41/0x42/0x43/0x44/0x45/0x40 の

プロパティ値が対応するものとする。プロパティ値のとる値については、本クラスを実装する実機器が、その機能として取りうるプロパティ値のみを実装すればよいものとする。例えば、本クラスを搭載する実機器が送風機能をその機能として搭載していない場合は、送風に対する 0x45 を実装する必要はない。
動作状態プロパティ（0x80）が OFF(０ｘ３１)の場合であっても、本プロパティの有効性は保証されるものとする。

(4) 温度自動設定

家庭用エアコンが、「温度設定値」(EPC＝0xB3，0xB5，0xB6，0xB7)を目標値とせず、家庭用エアコン本体の自動温度設定値算出アルゴリズム等により、動作している状態の ON/OFF を設定し、設定状態を取得する。
自動状態 ON の場合 0x41 とし、自動状態 OFF の場合は 0x42 のプロパティ値を取るものとする。
動作状態プロパティ（0x80）が OFF(０ｘ３１)の場合であっても、本プロパティの有効性は保証されるものとする。

(5) 急速動作モード設定

通常運転／急速／静音のモードを設定し、設定状態を取得する。それぞれに、順に 0x41/0x42/0x43 のプロパティ値が対応するものとする。本プロパティは、「運転モード設定」(EPC=0xB0)を修飾して設定することが可能であるプロパティで、急速設定は、急速冷房、急速暖房もしくは、パワフル動作状態等に対応する。また、本プロパティの通常運転／急速／静音状態は、それぞれ排他的関係にある設定状態である。

(6) 温度設定値

エアコンの現在の「運転モード設定」における温度設定値を℃の単位で設定し、設定状態を取得する。本プロパティで示す温度設定値は、エアコンに「温度自動設定」の機能が実装されていない場合、または実装されているが「温度自動設定」が非 AUTO の状態を取る場合に、エアコンが目標とする温度値である。また、「温度自動設定」が AUTO 状態であるために、本プロパティで表す温度設定目標値が不明となってしまう場合に本プロパティがとる値は０ｘFD（温度設定値不明）とする。
動作状態プロパティ（0x80）が OFF(０ｘ３１)の場合であっても、本プロパティの有効性は保証されるものとする。

(7) 除湿モード時相対湿度設定値

「運転モード設定」(EPC=0xB0)が、除湿モードの場合の相対湿度設定値を%の単位で設定し、設定状態を取得する。本プロパティを実装する場合は、「運転モード設定」(EPC=0xB0)

の現在の設定が、除湿モード以外の場合も設定／取得が可能である。
動作状態プロパティ（0x80）が OFF(０ｘ３１)の場合であっても、本プロパティの有効性は保証されるものとする。

(8) 冷房モード時温度設定値

「運転モード設定」(EPC=0xB0)が、冷房モードの場合の温度設定値を℃の単位で設定し、設定状態を取得する。本プロパティを実装する場合は、「運転モード設定」(EPC=0xB0)の現在の設定が、冷房モード以外の場合も設定／参照が可能である。
本プロパティで示す冷房モード時温度設定値は、エアコンに「温度自動設定」の機能が実装されていない場合、または実装されているが「温度自動設定」が非 AUTO の状態を取る場合に、エアコンが目標とする温度値である。
本プロパティを実装する場合は、温度設定値プロパティ（EPC:0xB3）のプロパティ内容と一致しなければならない。
動作状態プロパティ（0x80）が OFF(０ｘ３１)の場合であっても、本プロパティの有効性は保証されるものとする。

(9) 暖房モード時温度設定値

「運転モード設定」(EPC=0xB0)が、暖房モードの場合の温度設定値を℃の単位で設定し、設定状態を取得する。本プロパティを実装する場合は、「運転モード設定」(EPC=0xB0)の現在の設定が、暖房モード以外の場合も設定／参照が可能である。
本プロパティで示す暖房モード時温度設定値は、エアコンに「温度自動設定」の機能が実装されていない場合、または実装されているが「温度自動設定」が非 AUTO の状態を取る場合に、エアコンが目標とする温度値である。
本プロパティを実装する場合は、温度設定値プロパティ（EPC:0xB3）のプロパティ内容と一致しなければならない。
動作状態プロパティ（0x80）が OFF(０ｘ３１)の場合であっても、本プロパティの有効性は保証されるものとする。

(10) 除湿モード時温度設定値

「運転モード設定」(EPC=0xB0)が、除湿モードの場合の温度設定値を℃の単位で設定し、設定状態を取得する。本プロパティを実装する場合は、「運転モード設定」(EPC=0xB0)の現在の設定が、除湿モード以外の場合も設定／参照が可能である。
本プロパティで示す除湿モード時温度設定値は、エアコンに「温度自動設定」の機能が実装されていない場合、または実装されているが「温度自動設定」が非 AUTO の状態を取る場合に、エアコンが目標とする温度値である。
本プロパティを実装する場合は、温度設定値プロパティ（EPC:0xB3）のプロパティ内容と

一致しなければならない。
動作状態プロパティ（0x80）が OFF(0x31)の場合であっても、本プロパティの有効性は保証されるものとする。

(11) 定格消費電力値

冷房／暖房／除湿／送風の各運転モード時の定格消費電力（カタログ値)を W の単位で示す。各モード毎の消費電力は、0x0000～0xFFFD(0～65533W)とし、冷房／暖房／除湿／送風の順に、上位 Byte からプロパティ値とする。実機器が、その機能として、いずれかの運転モードをサポートしない場合は、アンダーフローコード 0xFFFE を用いるものとする。
動作状態プロパティ（0x80）が OFF(0x31)の場合であっても、本プロパティの有効性は保証されるものとする。

(12) 消費電流計測値

エアコンの現在の消費電流を 0.1A の単位で示す。対象となる電流が交流の場合は、実効値を示すものとする。プロパティ値は、0x0000～0xFFFD(0～6553.3A)とし、実機器のプロパティ値が、プロパティの値域を超える場合は、オーバーフローコード 0xFFFF、実機器のプロパティ値が、プロパティの値域未満の場合は、アンダーフローコード 0xFFFE を用いるものとする。

(13) 室内相対湿度計測値

室内相対湿度計測値を%の単位で示す。プロパティの値域は、0x00～0x64(0～100%)とし、実機器のプロパティ値が、プロパティの値域を超える場合は、オーバーフローコード 0xFF、実機器のプロパティ値が、プロパティの値域未満の場合は、アンダーフローコード 0xFE を用いるものとする。また、計測値を返せない場合は、0xFD を用いるものとする。
動作状態プロパティ（0x80）が OFF(0x31)の場合であっても、本プロパティの有効性は保証されるものとする。

(14) 室内温度計測値

室内温度計測値を℃の単位で示す。プロパティの値域は、0x81～0x7D(－127～125℃)とし、実機器のプロパティ値が、プロパティの値域を超える場合は、オーバーフローコード 0x7F、実機器のプロパティ値が、プロパティの値域未満の場合は、アンダーフローコード 0x80 を用いるものとする。また、計測値を返せない場合は、0x7E を用いるものとする。
動作状態プロパティ（0x80）が OFF(0x31)の場合であっても、本プロパティの有効性は保証されるものとする。

(15) ユーザリモコン温度設定値

家庭用エアコンのユーザのリモコンによって、設定された最新の温度設定値を℃の単位で示す。本プロパティの使用例として、コントローラ等から家庭用エアコンの設定温度を変更した場合に、ユーザのリモコン等による設定温度を参照するために用いる。
動作状態プロパティ（0x80）が OFF(0x31)の場合であっても、本プロパティの有効性は保証されるものとする。

(16) 吹き出し温度計測値

吹き出し温度計測値を℃の単位で示す。プロパティの値域は、0x81～0x7D(－127～125℃)とし、実機器のプロパティ値が、プロパティの値域を超える場合は、オーバーフローコード 0x7F、実機器のプロパティ値が、プロパティの値域未満の場合は、アンダーフローコード 0x80 を用いるものとする。また、計測値を返せない場合は、０ｘ7E を用いるものとする。
動作状態プロパティ（0x80）が OFF(0x31)の場合であっても、本プロパティの有効性は保証されるものとする。

(17) 外気温度計測値

室外機が設置されている場所の温度計測値(外気温度)を℃の単位で示す。プロパティの値域は、0x81～0x7D(－127～125℃)とし、実機器のプロパティ値が、プロパティの値域を超える場合は、オーバーフローコード 0x7F、実機器のプロパティ値が、プロパティの値域未満の場合は、アンダーフローコード 0x80 を用いるものとする。また、計測値を返せない場合は、０ｘ7E を用いるものとする。
動作状態プロパティ（0x80）が OFF(0x31)の場合であっても、本プロパティの有効性は保証されるものとする。

(18) 相対温度設定値

家庭用エアコンが運転中の動作モードにおいて目標としている温度（目標温度）に対し、「高め」「低め」を相対値で設定し、設定状態を取得する。また、絶対値での温度設定を目標値としない運転モード、例えば自動等においても、「高め」「低め」を相対値で設定する。本プロパティ値は、0.1℃刻みで－12.7℃～12.5℃（0x81～0x7D）の範囲を取る。1.0℃低めの場合は 0xF6 を指定し、1.0℃高めの場合は 0x0A を指定する。
実機器のプロパティ値が、プロパティ値の上限を超える場合には、オーバーフローコード 0x7F、実機器のプロパティ値が、プロパティ値の下限未満の場合は、アンダーフローコード 0x80 を用いるものとする。また、設定値を返せない場合は、0x7E を用いるものとする。
動作状態プロパティ（0x80）が OFF(0x31)の場合であっても、本プロパティの有効性は保証されるものとする。

(19) 風量設定

風量レベルおよび、風量自動状態を設定し、設定状態を取得する。風量自動状態のプロパティの値は、0x41 とする。風量レベルを 8 段階で設定し、0x31～0x38 のプロパティ値を取るものとする。各風量レベルの具体的な値は、規定しないが、0x31 を風量最小、0x38 を風量最大とする。

動作状態プロパティ（0x80）が OFF(０ｘ３１)の場合であっても、本プロパティの有効性は保証されるものとする。

生活者の見守りサービスのために、状変時アナウンスを必須とする。

(20) 風向自動設定

風向き（上下・左右）の、AUTO／非 AUTO を設定し、設定状態を取得する。

全 AUTO=0x41、非 AUTO=0x42、上下 AUTO=0x43、左右 AUTO=0x44

プロパティ値のとる値については、本クラスを実装する実機器が、その機能として取りうるプロパティ値のみを実装すればよいものとする。

動作状態プロパティ（0x80）が OFF(０ｘ３１)の場合であっても、本プロパティの有効性は保証されるものとする。

(21) 風向スイング設定

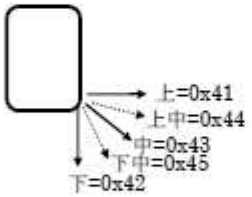
風向スイングの方向を設定し､設定状態を取得する｡風向スイング OFF＝0x31, 上下＝0x41, 左右＝0x42, 上下左右＝0x43

プロパティ値のとる値については、本クラスを実装する実機器が、その機能として取りうるプロパティ値のみを実装すればよいものとする。

(22) 風向上下設定

風向の上下角度を設定し、設定状態を取得する。上下角度を 5 段階で表示する。

上＝0x41、下＝0x42、中央＝0x43、上中=0x44、下中=0x45



（横から見た図）

プロパティ値のとる値については、本クラスを実装する実機器が、その機能として取りうるプロパティ値のみを実装すればよいものとする。また、スイング動作中に対する本プロパティの有効性は機器依存とする。

(23) 風向左右設定

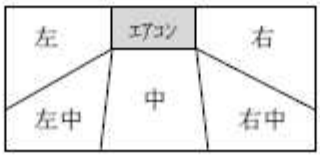
左右方向の風向きを設定し、設定状態を取得する。下表の通り風向きの方向を左、左中、中、右中、右の５つの方向に分けて、どの方向に風を吹くかをコードで示す。（○：吹く方向）
プロパティ値のとる値については、本クラスを実装する実機器が、その機能として取りうるプロパティ値のみを実装すればよいものとする。また、スイング動作中に対する本プロパティの有効性は機器依存とする。

（Version 2.01 Release a 以前での元　右＝0x41、左＝0x42、中央＝0x43、左右＝0x44 は下表の通りに割り当てる。）

ｺｰﾄﾞ	左	左中	中	右中	右	備考
0x41	×	×	×	○	○	元「右」
42	○	○	×	×	×	元「左」
43	×	○	○	○	×	元「中央」
44	○	○	×	○	○	元「左右」
51	×	×	×	×	○	
52	×	×	×	○	×	
0x53 は無し（0x41＝元「右」があるため）						
54	×	×	○	×	×	
55	×	×	○	×	○	
56	×	×	○	○	×	
57	×	×	○	○	○	
58	×	○	×	×	×	
59	×	○	×	×	○	
5A	×	○	×	○	×	
5B	×	○	×	○	○	
5C	×	○	○	×	×	
5D	×	○	○	×	○	
0x5E は無し（0x43＝元「中央」があるため）						
5F	×	○	○	○	○	

ｺｰﾄﾞ	左	左中	中	右中	右	備考
0x60	○	×	×	×	×	
61	○	×	×	×	○	
62	○	×	×	○	×	
63	○	×	×	○	○	
64	○	×	○	×	×	
65	○	×	○	×	○	
66	○	×	○	○	×	
67	○	×	○	○	○	
0x68 は無し（0x42＝元「左」があるため）						
69	○	○	×	×	○	
6A	○	○	×	○	×	
0x6B は無し（0x44＝元「左右」があるため）						
6C	○	○	○	×	×	
6D	○	○	○	×	○	
6E	○	○	○	○	×	
6F	○	○	○	○	○	

５つの方向は右図の通り。



（上から見た図）

(24)　特殊状態

家庭用エアコンが、除霜状態、予熱状態、排熱状態にあることを示す。

除霜状態にある場合のプロパティ値を 0x41、予熱状態にある場合のプロパティ値を 0x42、排熱状態にある場合のプロパティ値を 0x43 とする。なお、いずれの状態でもない場合のプロパティ値を 0x40 とする。

予熱状態とは、暖房運転起動直後、あるいは除霜復帰直後から暖風温度が上昇し吹き出すまでの室内ファン停止状態または室内ファンが低回転状態などを言う。

排熱状態とは、運転停止(特に暖房運転)した際に、エアコンの機器内に残っている熱を排出するために、ファン(通常室内ファン)および冷凍サイクルを回している状態などを言う。

動作状態プロパティ（0x80）がOFF(０ｘ３１)の場合であっても、本プロパティの有効性は保証されるものとする。

(25)　非優先状態

家庭用エアコンが、非優先状態にあることを示す。非優先状態の一例として、１つの室外機に接続されている複数の室内機において、対象となるエアコンの取れる動作モードが、他のエアコンの動作モードによって制限されている状態などを表す。

通常状態の場合のプロパティ値を 0x40、非優先状態の場合のプロパティ値 0x41 とする。

動作状態プロパティ（0x80）が OFF(０ｘ３１)の場合であっても、本プロパティの有効性は保証されるものとする。

(25)　換気モード設定

家庭用エアコンに搭載されている換気機能のモードの ON/OFF を設定し、設定状態を取得する。

換気 ON（排気方向）=0x41、換気 OFF=0x42、換気 ON（吸気方向）=0x43、

換気 ON（吸排気方向）=0x44

動作状態プロパティ（0x80）が OFF(０ｘ３１)の場合であっても、本プロパティの有効性は保証されるものとする。

(26)　加湿モード設定

家庭用エアコンに搭載されている加湿機能のモードの ON/OFF を設定し、設定状態を取得する。

加湿 ON＝0x41,OFF＝0x42

動作状態プロパティ（0x80）が OFF(０ｘ３１)の場合であっても、本プロパティの有効性は保証されるものとする。

(27)　換気風量設定

換気風量レベルおよび、換気風量自動状態を設定し、設定内容を取得する。換気風量自動状

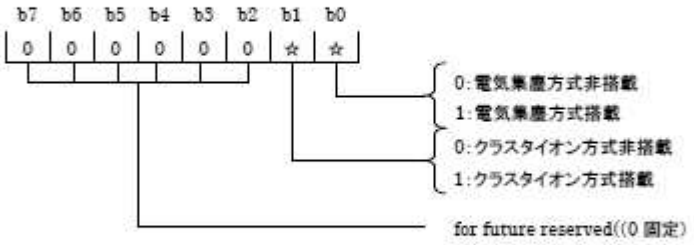
態のプロパティの値は、0x41 とする。換気風量レベルを 8 段階で設定し、0x31～0x38 のプロパティ値を取るものとする。各換気風量レベルの具体的な値は、規定しないが、0x31 を換気風量最小、0x38 を換気風量最大とする。
動作状態プロパティ（0x80）が OFF(０ｘ３１)の場合であっても、本プロパティの有効性は保証されるものとする。

(28) 加湿量設定

加湿量レベルおよび、加湿量自動状態を設定し、設定内容を取得する。加湿量自動状態のプロパティの値は、0x41 とする。加湿量レベルを 8 段階で設定し、0x31～0x38 のプロパティ値を取るものとする。各加湿量レベルの具体的な値は、規定しないが、0x31 を加湿量最小、0x38 を加湿量最大とする。
動作状態プロパティ（0x80）が OFF(０ｘ３１)の場合であっても、本プロパティの有効性は保証されるものとする。

(29) 搭載空気清浄方法

どのような実現方法の空気清浄機能が搭載されているかの一覧をビットマップで示す。実現方法として、電気集塵方式とクラスタイオン方式の 2 方式を規定する。以下に詳細を示す。ビットが 0 の場合は当該実現方法が搭載されていないことを、1 の場合は搭載されていることを示す。



動作状態プロパティ（0x80）が OFF(０ｘ３１)の場合であっても、本プロパティの有効性は保証されるものとする。

(30) 空気清浄機能モード設定

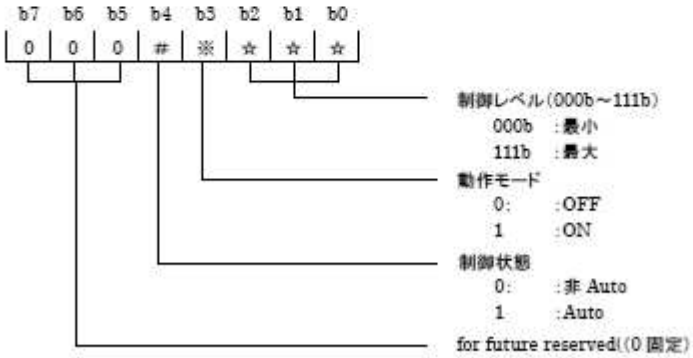
空気清浄機能の動作モード（ON／OFF）と制御状態の AUTO／非 AUTO 状態、および制御状態が非 AUTO である場合の制御レベルを、空気清浄機能の実現方法ごとに設定し、設定状態を取得する。本プロパティは 8 要素からなる配列で、要素番号と空気清浄機能の実現方法が 1 対 1 に対応する。要素番号と実現方法の対応は以下のとおりである。

第 0 要素：電気集塵方式

第 1 要素：クラスタイオン方式

第 2 要素～第 7 要素：for future reserved

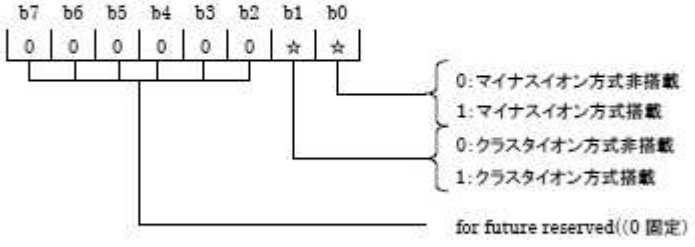
各要素のサイズは 1 バイトである。b0～b2 で対応する方式の制御レベルを 000b～111b の 8 段階で示す。各制御レベルの具体値は規定しないが、000b が制御レベル最小、111b が制御レベル最大を示すものとする。b3 は対応する方式の ON／OFF を示す。b3=0 が動作モード OFF、b3=1 が動作モード ON を示す。b4 は対応する方式の制御状態が自動（Auto）であるか否（非 Auto）かを示す。b4=0 が非 Auto、b4=1 が Auto であることを示す。b4=1（Auto）の場合、b0～b2 による制御レベルの設定は無効となる。下図に詳細を示す。



動作状態プロパティ（0x80）が OFF(０ｘ３１)の場合であっても、本プロパティの有効性は保証されるものとする。

(31) 搭載リフレッシュ方法

どのような実現方法のリフレッシュ機能が搭載されているかの一覧をビットマップで示す。実現方法として、マイナスイオン方式とクラスタイオン方式の 2 方式を規定する。以下に詳細を示す。ビットが 0 の場合は当該実現方法が搭載されていないことを、1 の場合は搭載されていることを示す。



動作状態プロパティ（0x80）が OFF(０ｘ３１)の場合であっても、本プロパティの有効性は保証されるものとする。

(32) リフレッシュ機能モード設定

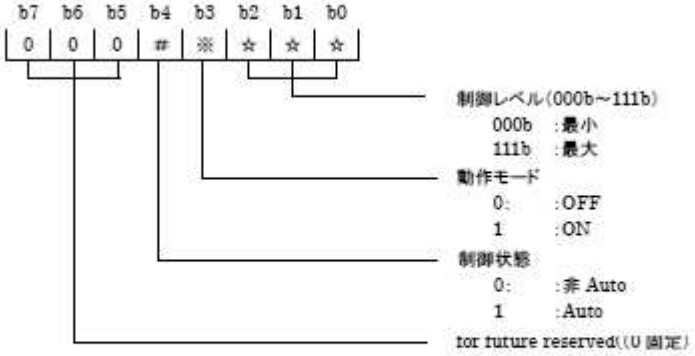
リフレッシュ機能の動作モード（ON／OFF）と制御状態の AUTO／非 AUTO 状態、および制御状態が非 AUTO である場合の制御レベルを、リフレッシュ機能の実現方法ごとに設定し、設定状態を取得する。本プロパティは 8 要素からなる配列で、要素番号とリフレッシュ機能の実現方法が 1 対 1 に対応する。要素番号と実現方法の対応は以下のとおりである。

第 0 要素：マイナスイオン方式

第 1 要素：クラスタイオン方式

第 2 要素～第 7 要素：for future reserved

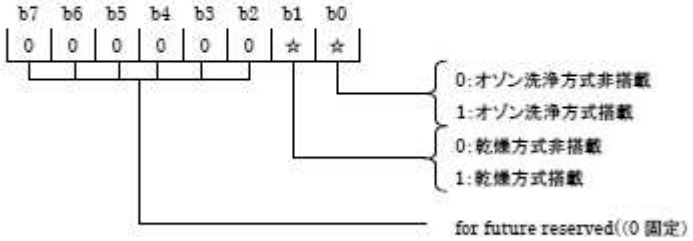
各要素のサイズは 1 バイトである。b0～b2 で対応する方式の制御レベルを 000b～111b の 8 段階で示す。各制御レベルの具体値は規定しないが、000b が制御レベル最小、111b が制御レベルを示すものとする。b3 は対応する方式の ON／OFF を示す。b3=0 が動作モード OFF、b3=1 が動作モード ON を示す。b4 は対応する方式の制御状態が自動（Auto）であるか否（非 Auto）かを示す。b4=0 が非 Auto、b4=1 が Auto であることを示す。b4=1（Auto）の場合、b0～b2 による制御レベルの設定は無効となる。下図に詳細を示す。



動作状態プロパティ（0x80）が OFF(０ｘ３１)の場合であっても、本プロパティの有効性は保証されるものとする。

(33) 搭載自己洗浄方法

どのような実現方法の自己洗浄機能が搭載されているかの一覧をビットマップで示す。実現方法として、オゾン洗浄方式と乾燥方式の 2 方式を規定する。以下に詳細を示す。ビットが 0 の場合は当該実現方法が搭載されていないことを、1 の場合は搭載されていることを示す。



動作状態プロパティ（0x80）が OFF(０ｘ３１)の場合であっても、本プロパティの有効性は保証されるものとする。

(34) 自己洗浄機能モード設定

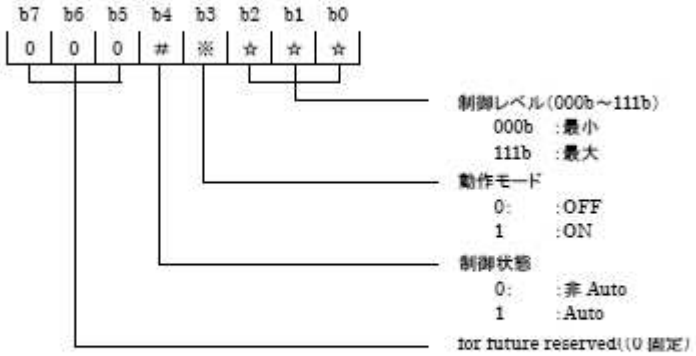
自己洗浄機能の動作モード（ON／OFF）と制御状態の AUTO／非 AUTO 状態、および制御状態が非 AUTO である場合の制御レベルを、自己洗浄機能の実現方法ごとに設定し、設定状態を取得する。本プロパティは 8 要素からなる配列で、要素番号と自己洗浄機能の実現方法が 1 対 1 に対応する。要素番号と実現方法の対応は以下のとおりである。

第 0 要素：オゾン洗浄方式

第 1 要素：乾燥方式

第 2 要素～第 7 要素：for future reserved

各要素のサイズは 1 バイトである。b0～b2 で対応する方式の制御レベルを 000b～111b の 8 段階で示す。各制御レベルの具体値は規定しないが、000b が制御レベル最小、111b が制御レベル最大を示すものとする。b3 は対応する方式の ON／OFF を示す。b3=0 が動作モード OFF、b3=1 が動作モード ON を示す。b4 は対応する方式の制御状態が自動（Auto）であるか否（非 Auto）かを示す。b4=0 が非 Auto、b4=1 が Auto であることを示す。b4=1（Auto）の場合、b0～b2 による制御レベルの設定は無効となる。下図に詳細を示す。



動作状態プロパティ（0x80）が OFF(０ｘ３１)の場合であっても、本プロパティの有効性は保証されるものとする。

(35) 特別運転モード設定

「運転モード設定」プロパティ(0xB0)で設定されるモードのより詳細なモードを設定し、設定状態を取得する。プロパティ値は、衣類乾燥を 0x41、結露抑制を 0x42、ダニカビ抑制を 0x43、強制除霜を 0x44 で示す。また、設定なしの場合は 0x40 とする。0x45 以降は for future reserved とする。

「特別運転モード設定」の設定が「運転モード設定」(0xB0)プロパティに影響する場合、その結果を「運転モード設定」(0xB0)プロパティに反映されなければならない。例えば、衣類乾燥機能が暖房モードで動作する場合、「特別運転モード設定」プロパティの設定により「運転モード設定」プロパティ(0xB0)の状態は「暖房」になるが、双方のプロパティの関係については実装依存とし特に規定はしない。

プロパティ値のとる値については、本クラスを実装する実機器が、その機能として取りうるプロパティ値のみを実装すればよいものとする。

動作状態プロパティ（0x80）が OFF(０ｘ３１)の場合であっても、本プロパティの有効性は保証されるものとする。

(36) 内部動作状態

エアコンの内部動作をビットマップで表現する。

ﾋﾞｯﾄ0：コンプレッサ動作状態　　0　停止中　　1　動作中

ﾋﾞｯﾄ1：サーモＯＮ／ＯＦＦ状態　　0　サーモＯＦＦ状態　　1　サーモＯＮ状態

（ビット２～７　for future reserved）

ここでいう“サーモＯＦＦ状態”とは、“室温が設定温度(目標温度)に達しており熱交換が停止(エアコンとしては運転状態)している状態”という意味であり、“サーモON状態”は、“室温が設定温度からあるレベル以上はなれており、熱交換を行っている状態”と言う意味である。

動作状態プロパティ（0x80）が OFF(0x31)の場合であっても、本プロパティの有効性は保証されるものとする。

(37) 強制サーモモード設定

家庭エアコンにおいて、サーモ設定を無視して運転するか否かを設定する。

通常設定＝0x40、強制サーモＯＮ=0x41、強制サーモＯＦＦ=0x42

通常設定とは、室内外温度の変化に応じて運転／停止(エアコンとしては運転状態)する設定を指し、”強制サーモＯＮ”は、設定温度を無視して熱交換をしつづける運転設定であり、”強制サーモＯＦＦ”は、設定温度に関わらず熱交換を停止する運転設定を指している。

動作状態プロパティ（0x80）が OFF(０ｘ３１)の場合であっても、本プロパティの有効性は保証されるものとする。

(38) 空気清浄モード設定

家庭用エアコンに搭載されている空気清浄機能のモードの ON（0x41）/OFF（0x42）を設定し、設定状態を取得する。

動作状態プロパティ（0x80）が OFF(0x31)の場合であっても、本プロパティの有効性は保証されるものとする。

(39) ON タイマ予約設定

ON タイマの予約の入、切を設定し、設定状態を取得する。本プロパティは、「ON タイマ時刻設定値」もしくは、「ON タイマ相対時間設定値」と関連する。

時刻予約、相対時間予約共に入＝0x41、予約切＝0x42、時刻予約のみ入り＝0x43、

相対時間予約のみ入り＝0x44

動作状態プロパティ（0x80）が OFF(0x31)の場合であっても、本プロパティの有効性は保証されるものとする。

(40) ON タイマ時刻設定値

「ON タイマ予約設定」が入の場合に、エアコンが ON になる時刻を時：0x00～0x17(0～23)、分：0x00～0x3B(0～59)で設定し、設定状態を取得する。時、分の順に上位バイトからプロパティ値とする。

動作状態プロパティ（0x80）が OFF(0x31)の場合であっても、本プロパティの有効性は保証されるものとする。

(41) ON タイマ相対時間設定値

「ON タイマ予約設定」が入の場合に、エアコンが ON になる時間を現在時刻からの相対時間で設定し、更新された時間を取得する。データ形式は、時：0x00～0xFF(0～255)、分：0x00～0x3B(0～59)とし、時、分の順に上位バイトからプロパティ値とする。

動作状態プロパティ（0x80）が OFF(0x31)の場合であっても、本プロパティの有効性は保証されるものとする。

(42) OFF タイマ予約設定

OFF タイマの予約の入、切を設定し、設定状態を取得する。本プロパティは、「OFF タイマ時刻設定値」もしくは、「OFF タイマ相対時間設定値」と関連する。

時刻予約、相対時間予約共に入＝0x41、予約切＝0x42、時刻予約のみ入り＝0x43、

相対時間予約のみ入り＝0x44

動作状態プロパティ（0x80）が OFF(0x31)の場合であっても、本プロパティの有効性は保証されるものとする。

(43) OFF タイマ時刻設定値

「OFF タイマ予約設定」が入の場合に、エアコンが OFF になる時刻を時：0x00～0x17(0～23)、分：0x00～0x3B(0～59)で設定し、設定状態を取得する。時、分の順に上位バイトからプロパティ値とする。

動作状態プロパティ（0x80）が OFF(０ｘ３１)の場合であっても、本プロパティの有効性は保証されるものとする。

(44) OFF タイマ相対時間設定値

「OFF タイマ予約設定」が入の場合に、エアコンが OFF になる時間を現在時刻からの相対時間で設定し、更新された時間を取得する。データ形式は、時：0x00～0xFF(0～255)、分：0x00～0x3B(0～59)とし、時、分の順に上位バイトからプロパティ値とする。

動作状態プロパティ（0x80）が OFF(０ｘ３１)の場合であっても、本プロパティの有効性は保証されるものとする。

３．３ 住宅・設備関連機器クラスグループ

本節では、機器オブジェクトの内、住宅・設備関連機器クラスグループ（クラスグループコード X1=0x02）に属する ECHONET オブジェクト毎に、コードやプロパティの詳細を規定する。本節で詳細を規定するオブジェクトクラスの一覧を、表７に示し、各クラス毎の詳細を項を設けて規定を示す。クラス規定において、「必須」の記述のあるものは、各クラスを搭載する機器には、そのプロパティとサービスの組み合わせの実装が必須であることを示す。

表７　住宅・設備関連機器クラスグループのオブジェクト一覧表（１／２）

クラスグループコード	クラスコード	クラス名	詳細規定の有無	備考
0x02	0x00～0x5F	For future reserved		
	0x60	電動ブラインド･日よけ	○	
	0x61	電動シャッター	○	
	0x62	電動カーテン		
	0x63	電動雨戸・シャッター	○	
	0x64	電動ゲート	○	
	0x65	電動窓	○	
	0x66	電動玄関ドア・引戸	○	
	0x67	散水器（庭用）	○	
	0x68	散水器（火災用）		
	0x69	噴水		
	0x6A	瞬間湯沸器		
	0x6B	電気温水器	○	
	0x6C	太陽熱温水器		
	0x6D	循環ポンプ		
	0x6E	電気便座（温水洗浄便座、暖房便座など）	○	
	0x6F	電気錠	○	
	0x70	ガス元弁		
	0x71	ホームサウナ		
	0x72	瞬間式給湯器	○	
	0x73	浴室暖房乾燥機	○	
	0x74	ホームエレベータ		
	0x75	電動間仕切り		
	0x76	水平トランスファ		
	0x77	電動物干し		
	0x78	浄化槽		
	0x79	住宅用太陽光発電	○	
	0x7A	冷温水熱源機	○	
	0x7B	床暖房	○	
	0x7C	燃料電池	○	

	0x7D	蓄電池	○	
	0x7E	電気自動車充放電器	○	
	0x7F	エンジンコージェネレーション	○	

住宅・設備関連機器クラスグループのオブジェクト一覧表（2／2）

0x02	0x80	電力量メータ	○	
	0x81	水流量メータ	○	
	0x82	ガスメータ	○	
	0x83	LP ガスメータ	○	
	0x84	時計		
	0x85	自動ドア		
	0x86	業務用エレベータ		
	0x87	分電盤メータリング	○	
	0x88	低圧スマート電力量メータ	○	
	0x89	スマートガスメータ	○	
	0x8A	高圧スマート電力量メータ	○	
	0x8B～0x8F	For future reserved		
	0x90※1)	一般照明	○	シャンデリア、スタンド、ブランケット、ダウンライト、スポット照明、ペンダント照明、シーリングライト、ウォールライト等を含む
	0x91	単機能照明	○	
	0x92～0x98※1)	For future reserved		
	0x99※2)	非常照明		誘導灯、非常灯、保安灯、防犯灯等を含む
	0x9A～0x9C※2)	For future reserved		
	0x9D	設備照明		
	0xA0	ブザー	○	
	0x9E～0x9F 0xA1～0xFF	For future reserved		

注）○：APPENDIX でプロパティ構成を含めた詳細を規定。

※1）Version 2.10 以前においては、シャンデリア、スタンド、ブランケット、ダウンライト、スポット照明、ペンダント照明、シーリングライト、ウォールライトに個別クラスコードを割り当てていたが、Version 2.11 以降は全て一般照明に統合する。

※2）Version 2.10 以前においては、誘導灯、非常灯、保安灯、防犯灯には、個別にクラスコードを割り当てていたが、Version 2.11 以降は全て非常照明に統合する。

３．３．１３ 住宅用太陽光発電クラス規定

クラスグループコード　：0x02
クラスコード　　：0x79
インスタンスコード　：0x01～0x7F (0x00：全インスタンス指定コード)

プロパティ名称	EPC	プロパティ内容 値域(10 進表記)	データ型	データサイズ	単位	アクセスルール	必須	状変時アナウンス	備考
動作状態	0x80	ON／OFF の状態を示す。 ＯＮ＝0x30，ＯＦＦ＝0x31	unsigned char	1 Byte	—	Set Get	 ○	○	
系統連系状態	0xD0	系統連系状態のタイプを示す。 系統連系（逆潮流可）＝0x00 独立＝0x01 系統連系（逆潮流不可）＝0x02	unsigned char	1 Byte	—	Get			
瞬時発電電力計測値	0xE0	瞬時発電電力を W で示す。 0x0000～0xFFFD（0～65533）	unsigned short	2 Byte	W	Get	○		
積算発電電力量計測値	0xE1	積算電力量を 0.001kWh で示す。 0x00000000～0x3B9AC9FF (0～999,999.999kWh)	unsigned long	4 Byte	0.001 kWh	Get	○		
積算発電電力量リセット設定	0xE2	0x00 を書き込むことにより積算発電電力量をリセットする リセット＝0x00	unsigned char	1 Byte	—	Set			
積算売電電力量計測値	0xE3	売電電力の積算値を 0.001kWh で示す。 0x00000000～0x3B9AC9FF (0～999,999.999kWh)	unsigned long	4 Byte	0.001 kWh	Get			
積算売電電力量リセット設定	0xE4	0x00 を書き込むことにより積算買電電力量をリセットする リセット＝0x00	unsigned char	1 Byte	—	Set			
発電電力制限設定１	0xE5	発電電力制限値を定格発電電力値の%で設定し、設定状態を取得する。 0x00～0x64（0～100%）	unsigned char	1 Byte	%	Set/ Get			
発電電力制限設定２	0xE6	発電電力制限値を W で設定し、設定状態を取得する 0x0000～0xFFFD（0～65533）	unsigned short	2 Byte	W	Set/ Get			
売電電力制限設定	0xE7	売電電力制限値を W で設定し、設定状態を取得する 0x0000～0xFFFD（0～65533）	unsigned short	2 Byte	W	Set/ Get			
定格発電電力値（系統連系時）	0xE8	系統連系時の定格発電電力値を W で示す 0x0000～0xFFFD（0～65533）	unsigned short	2 Byte	W	Set/ Get			
定格発電電力値（独立時）	0xE9	独立時の定格発電電力値を W で示す 0x0000～0xFFFD（0～65533）	unsigned short	2 Byte	W	Set/ Get			

注1)　状態変化時（状変時）アナウンスの○は、プロパティ実装時には、処理必須を示す。

(1) 動作状態

本クラス固有の機能が、稼動状態であるか否か(ON/OFF)を示す。尚、本クラスを搭載するノードにおいて、ノードの動作開始とともに、本クラスの機能が、稼動を開始する場合は、本プロパティを固定値 0x30（動作状態 ON）で実装することも可能である。

(2) 系統連系状態

現在の系統との接続状態(系統連系状態)を示す。
系統連系(逆潮流可)＝0x00, 独立＝0x01, 系統連系(逆潮流不可)=0x02

(3) 瞬時発電電力計測値

瞬時発電力を W の単位で示す。プロパティの値域は、0x0000～0xFFFD とし、実機器のプロパティ値が、プロパティの値域を超える場合は、オーバーフローコード 0xFFFF、実機器のプロパティ値が、プロパティの値域未満の場合は、アンダーフローコード 0xFFFE を用いるものとする。

(4) 積算発電電力量計測値

積算発電電力量を 0.001kWh の単位で示す。プロパティの値域は、0x00000000～0x3B9AC9FF(0～999,999.999kWh）とし、積算電力量のオーバーフロー時は、0x00000000 から再インクリメントするものとする。

(5) 積算発電電力量リセット設定

0x00 をセットすることにより、積算発電電力量をゼロにリセットする。

(6) 積算売電電力量計測値

積算売電電力量を 0.001kWh の単位で示す。プロパティの値域は、0x00000000～0x3B9AC9FF(0～999,999.999kWh）とし、積算電力量のオーバーフロー時は、0x00000000 から再インクリメントするものとする。

(7) 積算売電電力量リセット設定

0x00 をセットすることにより、積算売電電力量をゼロにリセットする。

(8) 発電電力制限設定 1

発電電力を定格発電電力値の％で設定し、設定状態を取得する。プロパティの値域は、0～100（0x00～0x64）とし、単位は%とする。本プロパティ値が 100 の場合は、制限なしを示す。本プロパティに設定された値での発電電力値制限が不可能な場合は、設定された値を

越えない範囲で最も近い値での制限を行う。

(9) 発電電力制限設定２

発電電力を W の単位で設定し、設定状態を取得する。プロパティの値域は、0x0000～0xFFFD（0～65533）とする。本プロパティに設定された値での発電電力値制限が不可能な場合は、設定された値を越えない範囲で最も近い値での制限を行う。

(10) 売電電力制限設定

売電電力を W の単位で設定し、設定状態を取得する。プロパティの値域は、0x0000～0xFFFD（0～65533）とする。本プロパティに設定された値での発電電力値制限が不可能な場合は、設定された値を越えない範囲で最も近い値での制限を行う。

(11) 定格発電電力値（系統連系時）

系統連系（逆潮流可・逆潮流不可）時の定格発電電力（カタログ値）を W の単位で示す。プロパティの値域は、0x0000～0xFFFD（0～65533）とする。

(12) 定格発電電力値（独立時）

独立時の定格発電電力（カタログ値）を W の単位で示す。プロパティの値域は、0x0000～0xFFFD（0～65533）とする。

３．３．１８ 電気自動車充放電器クラス規定

クラスグループコード　：　0x02

クラスコード　　：0x7E

インスタンスコード　：0x01～0x7F (0x00：全インスタンス指定コード)

プロパティ名称	EPC	プロパティ内容 値域(10 進表記)	データ型	データサイズ	単位	アクセスルール	必須	状変時アナウンス	備考
動作状態	0x80	ON／OFF の状態を示す	unsigned char	1 Byte	－	Set		○	
		ON＝0x30，OFF＝0x31				Get	○		
車載電池の放電可能容量値 1	0xC0	電気自動車充放電器に接続された電気自動車の車載電池の放電可能容量を Wh で示す	unsigned long	4 Byte	Wh	Get	○ *1		
		0x00000000～0x3B9AC9FF（0～999,999,999Wh）							
車載電池の放電可能容量値 2	0xC1	電気自動車充放電器に接続された電気自動車の車載電池の放電可能容量を 0.1Ah で示す	unsigned short	2 Byte	0.1Ah	Get			
		0x0000～0x7FFE（0～3,276.6Ah）							
車載電池の放電可能残容量 1	0xC2	電気自動車充放電器に接続された電気自動車の車載電池の放電可能残容量を Wh で示す	unsigned long	4 Byte	Wh	Get	○ *2		
		0x00000000～0x3B9AC9FF（0～999,999,999Wh）							
車載電池の放電可能残容量 2	0xC3	電気自動車充放電器に接続された電気自動車の車載電池の放電可能残容量を 0.1Ah で示す	unsigned short	2 Byte	0.1Ah	Get			
		0x0000～0x7FFE（0～3,276.6Ah）							
車載電池の放電可能残容量 3	0xC4	電気自動車充放電器に接続された電気自動車の車載電池の放電可能残容量（%）を示す	unsigned char	1 Byte	%	Get	○ *2		
		0x00～0x64（0～100%）							
定格充電能力	0xC5	電気自動車充放電器における定格充電能力を W で示す	unsigned long	4 Byte	W	Get	○		
		0x00000000～0x3B9AC9FF（0～999,999,999W）							
定格放電能力	0xC6	電気自動車充放電器における定格放電能力を W で示す	unsigned long	4 Byte	W	Get	○		
		0x00000000～0x3B9AC9FF（0～999,999,999W）							
車両接続・充放電可否状態	0xC7	電気自動車充放電器における充放電の可否を示す	unsigned char	1 Byte	－	Get	○	○	
		不定=0xFF 車両未接続=0x30 車両接続・充電不可・放電不可=0x40 車両接続・充電可・放電不可=0x41 車両接続・充電不可・放電可=0x42 車両接続・充電可・放電可=0x43							

最小最大充電電力値	0xC8	電気自動車充放電器への充電電力の最小値および最大値を、それぞれWで示す 0x00000000～0x3B9AC9FF（0～999,999,999W） 最小充電電力値：最大充電電力値	unsigned long ×2	8 Byte	W	Get	○		
最小最大放電電力値	0xC9	電気自動車充放電器からの放電電力の最小値および最大値を、それぞれWで示す 0x00000000～0x3B9AC9FF（0～999,999,999W） 最小放電電力値：最大放電電力値	unsigned long ×2	8 Byte	W	Get	○		
最小最大充電電流値	0xCA	電気自動車充放電器への充電電流の最小値および最大値を、それぞれ0.1Aで示す 0x0000～0x7FFE（0～3,276.6A） 最小充電電流値：最大充電電流値	unsigned short ×2	4 Byte	0.1A	Get	○		
最小最大放電電流値	0xCB	電気自動車充放電器からの放電電流の最小値および最大値を、それぞれ0.1Aで示す 0x0000～0x7FFE（0～3,276.6A） 最小放電電流値：最大放電電流値	unsigned short ×2	4 Byte	0.1A	Get	○		
充放電器タイプ	0xCC	電気自動車充放電器のタイプを示す AC_CPLT=0x11 AC_HLC（充電のみ）=0x12 AC_HLC（充放電可）=0x13 DC_タイプAA（充電のみ）=0x21 DC_タイプAA（充放電可）=0x22 DC_タイプAA（放電のみ）=0x23 DC_タイプBB（充電のみ）=0x31 DC_タイプBB（充放電可）=0x32 DC_タイプBB（放電のみ）=0x33 DC_タイプEE（充電のみ）=0x41 DC_タイプEE（充放電可）=0x42 DC_タイプEE（放電のみ）=0x43 DC_タイプFF（充電のみ）=0x51 DC_タイプFF（充放電可）=0x52 DC_タイプFF（放電のみ）=0x53 *6	unsigned char	1 Byte	－	Get	○		
車両接続確認	0xCD	電気自動車充放電器と車両の接続状態を確認する 接続確認=0x10	unsigned char	1 Byte	－	Set	○ *7		
車載電池の使用容量値1	0xD0	電気自動車充放電器に接続された電気自動車の車載電池の容量をWhで示す 0x00000000～0x3B9AC9FF（0～999,999,999Wh）	unsigned long	4 Byte	Wh	Get	○ *3		
車載電池の使用容量値2	0xD1	電気自動車充放電器に接続された電気自動車の車載電池の容量を0.1Ahで示す 0x0000～0x7FFE（0～3,276.6Ah）	unsigned short	2 Byte	0.1Ah	Get			
定格電圧	0xD2	通常時の電気自動車充放電器の定格電圧をVで示す	unsigned short	2 Byte	V	Get			

		0x0000～0x7FFE（0～32,766V）							
瞬時充放電電力計測値	0xD3	瞬時充放電電力を±W で示す 0x00000001～0x3B9AC9FF（1～999,999,999W）：充電時（プラス値）、0xFFFFFFFF～0xC4653601（－1～－999,999,999W）：放電時（マイナス値）	signed long	4 Byte	W	Get			
瞬時充放電電流計測値	0xD4	瞬時充放電電流を±0.1A で示す 0x0001 ～ 0x7FFE （ 0.1 ～ 3,276.6A）：充電時（プラス値）、0xFFFF ～ 0x8001 （ － 0.1 ～ － 3,276.7A）：放電時（マイナス値）	signed short	2 Byte	0.1A	Get			
瞬時充放電電圧計測値	0xD5	瞬時充放電電圧を±V で示す 0x0001～0x7FFE（1～32,766V）：充電時（プラス値）、0xFFFF～0x8001（－1～－32,767V）：放電時（マイナス値）	signed short	2 Byte	V	Get			
積算放電電力量計測値	0xD6	積算放電電力量を 0.001kWh で示す 0x00000000～0x3B9AC9FF（0～999,999.999kWh	unsigned long	4 Byte	0.001 kWh	Get			
積算放電電力量リセット設定	0xD7	積算放電電力量をリセットする リセット=0x00	unsigned char	1 Byte	－	Set			
積算充電電力量計測値	0xD8	積算充電電力量を 0.001kWh で示す 0x00000000～0x3B9AC9FF（0～999,999.999kWh	unsigned long	4 Byte	0.001 kWh	Get			
積算充電電力量リセット設定	0xD9	積算充電電力量をリセットする リセット=0x00	unsigned char	1 Byte	－	Set			
運転モード設定	0xDA	充電/放電/待機/停止/その他の運転モードを設定する 充電＝0x42, 放電＝0x43, 待機＝0x44, 停止=0x47, その他＝0x40	unsigned char	1 Byte	－	Set /Get	○	○	
系統連系状態	0xDB	電気自動車充放電器の系統連系状態を示す。 系統連系（逆潮流可）＝0x00 独立＝0x01 系統連系（逆潮流不可）＝0x02	unsigned char	1 Byte	－	Get			
車載電池の電池残容量 1	0xE2	電気自動車充放電器に接続された電気自動車の車載電池の残容量を Wh で示す 0x00000000～0x3B9AC9FF（0～999,999,999Wh）	unsigned long	4 Byte	Wh	Get	○ *4		
車載電池の電池残容量 2	0xE3	電気自動車充放電器に接続された電気自動車の車載電池の残容量を 0.1Ah で示す 0x0000～0x7FFE（0～3,276.6Ah）	unsigned short	2 Byte	0.1Ah	Get			
車載電池の電池残容量 3	0xE4	電気自動車充放電器に接続された電気自動車の車載電池残容量（%）を示す 0x00～0x64（0～100%）	unsigned char	1 Byte	%	Get	○ *4		
充電量設定値 1	0xE7	充電の電力量を Wh で指定する 0x00000000～0x3B9AC9FF（0～999,999,999Wh）	unsigned long	4 Byte	Wh	Set/ Get			*5

充電量設定値 2	0xE9	充電の容量を 0.1Ah で指定する 0x0000～0x7FFE (0～3,276.6Ah)	unsigned short	2 Byte	0.1Ah	Set/ Get			*5
充電電力設定値	0xEB	充電の電力を W で指定する 0x00000000～0x3B9AC9FF (0～999,999,999W)	unsigned long	4 Byte	W	Set/ Get			
放電電力設定値	0xEC	放電の電力を W で指定する 0x00000000～0x3B9AC9FF (0～999,999,999W)	unsigned long	4 Byte	W	Set/ Get			
充電電流設定値	0xED	充電の電流を 0.1A で指定する 0x0000～0xFFFD (0～6,553.3A)	unsigned short	2 Byte	0.1A	Set/ Get			
放電電流設定値	0xEE	放電の電流を 0.1A で指定する 0x0000～0xFFFD (0～6,553.3A)	unsigned short	2 Byte	0.1A	Set/ Get			
定格電圧 (独立時)	0xEF	独立時の電気自動車充放電器の定格電圧を V で示す 0x0000～0x7FFE (0～32,766V)	unsigned short	2 Byte	V	Get			

注1)　状態変化時 (状変時) アナウンスの○は、プロパティ実装時には、処理必須を示す。

注 2)

*1：車載電池の放電可能容量値 1 は、電気自動車充放電器に接続された電気自動車から出力される場合、搭載を必須とする。応答できない状況にある場合、不可応答を返す。

*2：車載電池の放電可能残容量 1、車載電池の放電可能残容量 3 は、電気自動車充放電器に接続された電気自動車から出力される場合、いずれかの搭載を必須とする。応答できない状況にある場合、不可応答を返す。

*3：車載電池の使用容量値 1 は、電気自動車充放電器に接続された電気自動車から出力される場合、搭載を必須とする。応答できない状況にある場合、不可応答を返す。

*4：車載電池の電池残容量 1、車載電池の電池残容量 3 は、電気自動車充放電器に接続された電気自動車から出力される場合、いずれかの搭載を必須とする。応答できない状況にある場合、不可応答を返す。

*5：充電量設定値 1 (または 2) を使用する場合は充電量設定値 2 (または 1) を使用しない

*6：充放電器タイプの DC_タイプ AA、DC_タイプ BB、DC_タイプ EE 及び DC_タイプ FF は、IEC 62196-3 にて規定されている Configuration AA、Configuration BB、Configuration EE 及び Configuration FF に記載の形状のコネクタを有する電気自動車充放電器とする。

*7：車両接続確認は充放電器タイプが DC_タイプ AA の場合のみ必須とする。

※電気自動車充放電器は、電気自動車と接続される EVPS (Electric Vehicle Power System) を含む充放電器である。電気自動車充放電器に接続される電気自動車は変更可能であるため、電気自動車充放電器に接続される電気自動車に応じて、電気自動車充放電器の各プロパティ値も変化する。そのため、車両接続・充放電可否状態が車両未接続から車両接続に変化した際に、接続する電気自動車が変わっている可能性があるため、電気自動車と EVPS により定まる諸元に関するプロパティ値を再取得することが望ましい。

例：車載電池の使用容量値 1、車載電池の放電可能可能容量値 1、車載電池の放電可能残容量値 1、車載電池の放電可能残容量値 3

なお、本クラスで取り扱う電流、電圧、電力の値はACとして取り扱う事とする。

(1) 動作状態（機器オブジェクトスーパークラスのプロパティを継承）
電気自動車充放電器が、状態取得および設定受付が可能な状態（ON 状態）であるか否か（OFF 状態）を示す。ON 状態には 0x30 を、OFF 状態には 0x31 を対応させる。

(2) 車載電池の放電可能容量値 1
電気自動車充放電器に接続された電気自動車の車載電池の放電可能容量を Wh 単位で示す。プロパティの値域は、0x00000000～0x3B9AC9FF（0～999,999,999Wh）とする。

(3) 車載電池の放電可能容量値 2
電気自動車充放電器に接続された電気自動車の車載電池の放電可能容量を 0.1Ah 単位で示す。プロパティの値域は、0x0000～0x7FFE（0～3,276.6Ah）とする。

(4) 車載電池の放電可能残容量 1
電気自動車充放電器に接続された電気自動車の車載電池の放電可能残容量を Wh で示す。プロパティの値域は、0x00000000～0x3B9AC9FF（0～999,999,999Wh）とする。

(5) 車載電池の放電可能残容量 2
電気自動車充放電器に接続された電気自動車の車載電池の放電可能残容量を 0.1Ah で示す。プロパティの値域は、0x0000～0x7FFE（0～3,276.6Ah）とする。

(6) 車載電池の放電可能残容量 3
電気自動車充放電器に接続された電気自動車の車載電池の放電可能残容量を%で示す。プロパティの値域は、0x00～0x64（0～100%）とする。

(7) 定格充電能力
電気自動車充放電器の定格充電能力を W 単位で示す。プロパティの値域は、0x00000000～0x3B9AC9FF（0～999,999,999W）とする。充電機能が存在しない場合は 0W とする。

(8) 定格放電能力
電気自動車充放電器の定格放電能力を W 単位で示す。プロパティの値域は、0x00000000～0x3B9AC9FF（0～999,999,999W）とする。放電機能が存在しない場合は 0W とする。

(9) 車両接続・充放電可否状態
電気自動車充放電器における充放電の可否を示す。車両と接続されていない場合を 0x30 で

示す。車両と接続され、充電不可かつ放電不可の場合を 0x40、充電可かつ放電不可の場合を 0x41、充電不可かつ放電可の場合を 0x42、充電可かつ放電可の場合を 0x43 で示す。
充放電を開始しないと車両状態がわからない場合、充電可及び放電可の状態については、充電開始または放電開始指示を与えられたときに出力するものとする。車両の未接続、接続状態を判別できない場合は不定状態として 0xFF で示す。

(10) 最小最大充電電力値

電気自動車充放電器への充電電力の最小値および最大値を W の単位で示す。それぞれの値域は、0x00000000～0x3B9AC9FF（0～999,999,999W）とし、最小/最大の順に上位 Byte からプロパティ値とする。実機器のプロパティ値が、プロパティの値域を超える場合は、オーバーフローコード 0xFFFFFFFF を用いるものとする。充電機能がない場合は 0 とする。

(11) 最小最大放電電力値

電気自動車充放電器への放電電力の最小値および最大値を W の単位で示す。それぞれの値域は、0x00000000～0x3B9AC9FF（0～999,999,999W）とし、最小/最大の順に上位 Byte からプロパティ値とする。実機器のプロパティ値が、プロパティの値域を超える場合は、オーバーフローコード 0xFFFFFFFF を用いるものとする。放電機能がない場合は 0 とする。

(12) 最小最大充電電流値

電気自動車充放電器への充電電流の最小値および最大値を 0.1A の単位で示す。それぞれの値域は、0x0000～0x7FFE（0～3,276.6A）とし、最小/最大の順に上位 Byte からプロパティ値とする。実機器のプロパティ値が、プロパティの値域を超える場合は、オーバーフローコード 0xFFFF を用いるものとする。充電機能がない場合は 0 とする。

(13) 最小最大放電電流値

電気自動車充放電器からの放電電流の最小値および最大値を 0.1A の単位で示す。それぞれの値域は、0x0000～0x7FFE（0～3,276.6A）とし、最小/最大の順に上位 Byte からプロパティ値とする。実機器のプロパティ値が、プロパティの値域を超える場合は、オーバーフローコード 0xFFFF を用いるものとする。放電機能がない場合は 0 とする。

(14) 充放電器タイプ

電気自動車充放電器のタイプを示す。電気自動車充放電器のタイプとして AC_CPLT (0x11)、AC_HLC (充電のみ) (0x12)、AC_HLC(充放電) (0x13)、DC_タイプ AA (充電のみ) (0x21)、DC_タイプ AA (充放電) (0x22)、DC_タイプ AA (放電のみ) (0x23)、DC_タイプ BB (充電のみ) (0x31)、DC_タイプ BB (充放電) (0x32)、DC_タイプ BB (放電のみ) (0x33)、DC_タイプ EE(充電のみ) (0x41)、DC_タイプ EE (充放電) (0x42)、DC_タイプ EE (放電のみ)

（0x43）、DC_タイプ FF（充電のみ）（0x51）、DC_タイプ FF（充放電）（0x52）、DC_タイプ FF（放電のみ）（0x53）のいずれかを示す。

充放電器タイプの説明

(1)AC_CPLT（0x11）：電気自動車へ交流で充電し、電気自動車とはＣＰＬＴ信号にて通信する。

・ AC_HLC（充電のみ）（0x12）：電気自動車へ交流で充電し、電気自動車とはＣＰＬＴ信号及びＨＬＣ信号にて通信する。
・ AC_HLC(充放電)（0x13）：電気自動車へ交流で充電、及び電気自動車から電気自動車充放電器へ交流で放電し、電気自動車とはＣＰＬＴ信号及びＨＬＣ信号にて通信する。
・ DC_タイプ AA（充電のみ）（0x21）：電気自動車へ直流で充電し、電気自動車とはタイプ AA 方式信号通信する。
・ DC_タイプ AA（充放電）（0x22）：電気自動車へ直流で充電、及び電気自動車から電気自動車充放電器へ直流で放電し、電気自動車とはタイプ AA 方式信号通信する。
・ DC_タイプ AA（放電のみ）（0x23）：電気自動車から電気自動車充放電器へ直流で放電し、電気自動車とはタイプ AA 方式信号通信する。
・ DC_タイプ BB（充電のみ）（0x31）：電気自動車へ直流で充電し、電気自動車とはタイプ BB 方式信号通信する。
・ DC_タイプ BB(充放電)（0x32）：電気自動車へ直流で充電、及び電気自動車から電気自動車充放電器へ直流で放電し、電気自動車とはタイプ BB 方式信号通信する。
・ DC_タイプ BB（放電のみ）（0x33）：電気自動車から電気自動車充放電器へ直流で放電し、電気自動車とはタイプ BB 方式信号通信する。
・ DC_タイプ EE（充電のみ）（0x41）：電気自動車へ直流で充電し、電気自動車とはタイプ EE 方式信号通信する。
・ DC_タイプ EE（充放電）（0x42）：電気自動車へ直流で充電、及び電気自動車から電気自動車充放電器へ直流で放電し、電気自動車とはタイプ EE 方式信号通信する。
・ DC_タイプ EE（放電のみ）（0x43）：電気自動車から電気自動車充放電器へ直流で放電し、電気自動車とはタイプ EE 方式信号通信する。
・ DC_タイプ FF（充電のみ）（0x51）：電気自動車へ直流で充電し、電気自動車とはタイプ FF 方式信号通信する。
・ DC_タイプ FF（充放電）（0x52）：電気自動車へ直流で充電、及び電気自動車から電気自動車充放電器へ直流で放電し、電気自動車とはタイプ FF 方式信号通信する。
・ DC_タイプ FF（放電のみ）（0x53）：電気自動車から電気自動車充放電器へ直流で放電し、電気自動車とはタイプ FF 方式信号通信する。

充放電器タイプと車両接続・充放電可否状態（0xC7）との関係についての説明

・ AC_CPLT（0x11）：常に不定（0xFF）となる。
・ AC_HLC（充電のみ）（0x12）又はAC_HLC(充放電)（0x13）：接続された車両がCPLT機能のみ搭載した車両の場合、不定（0xFF）となる。CPLT及びHLC機能を搭載した車両の場合、車両接続・充放電可否状態（0xC7）で示した内容となる。
・ DC_タイプAA（充電のみ）（0x21）、DC_タイプAA（充放電）（0x22）又はDC_タイプAA（放電のみ）（0x23）：車両接続確認（0xCD）にて情報を取得するまでは不定（0xFF）となる。情報取得後は車両接続・充放電可否状態（0xC7）で示した内容となる。
・ DC_タイプBB(充電のみ)（0x31）、DC_タイプBB（充放電）（0x32）、DC_タイプBB（放電のみ）（0x33）、DC_タイプEE(充電のみ)（0x41）、DC_タイプEE（充放電）（0x42）、DC_タイプEE(放電のみ)（0x43）、DC_タイプFF(充電のみ)（0x51）、DC_タイプFF（充放電）(0x52)又はDC_タイプFF（放電のみ）(0x53)、:車両接続・充放電可否状態（0xC7）で示した内容となる。

(15) 車両接続確認

電気自動車充放電器と電気自動車との接続状態及び充放電可否状態の確認を行う。これにより車両接続・充放電可否状態[0xC7]情報を取得することができる。
本プロパティは電気自動車充放電器タイプがDC_タイプAA（充電のみ）（0x21）、DC_タイプAA（充放電）（0x22）、DC_タイプAA（放電のみ）（0x23）のいずれかの場合にのみ必須である。

(16) 車載電池の使用容量値1

電気自動車充放電器に接続された電気自動車に搭載された電池の容量をWhの単位で示す。プロパティの値域は、0x00000000～0x3B9AC9FF（0～999,999,999Wh）とする。

(17) 車載電池の使用容量値2

電気自動車充放電器に接続された電気自動車に搭載された電池の容量を0.1Ahの単位で示す。プロパティの値域は、0x0000～0x7FFE（0～3,276.6Ah）とする。

(18) 定格電圧

通常時の電気自動車充放電器の定格電圧をVの単位で示す。プロパティの値域は、0x0000～0x7FFE（0～32,766V）とする。

(19) 瞬時充放電電力計測値

電気自動車充放電器の充放電時の瞬時電力をWの単位で示す。プロパティの値域は、充電時には、0x00000001～0x3B9AC9FF（1～999,999,999W）とし、放電時には、0xFFFFFFFF～0xC4653601（－1～－999,999,999W）とする。実機器のプロパティ値が、プロパティの

値域を超える場合は、オーバーフローコード 0x7FFFFFFF、実機器のプロパティ値が、プロパティの値域未満の場合は、アンダーフローコード 0x80000000 を用いるものとする。充放電をしていないときは 0。

(20) 瞬時充放電電流計測値

電気自動車充放電器の充放電時の瞬時電流を 0.1A の単位で示す。プロパティの値域は、充電時には、0x0001～0x7FFE（0.1～3,276.6A）とし、放電時には、0xFFFF～0x8001（－0.1～－3,276.7A）とする。実機器のプロパティ値が、プロパティの値域を超える場合は、オーバーフローコード 0x7FFF、実機器のプロパティ値が、プロパティの値域未満の場合は、アンダーフローコード 0x8000 を用いるものとする。充放電をしていないときは 0。

(21) 瞬時充放電電圧計測値

電気自動車充放電器の充放電時の瞬時電圧を V の単位で示す。プロパティの値域は、充電時には、0x0001～0x7FFE（1～32,766V）とし、放電時には、0xFFFF～0x8001（－1～－32,767V）とする。実機器のプロパティ値が、プロパティの値域を超える場合は、オーバーフローコード 0x7FFF、実機器のプロパティ値が、プロパティの値域未満の場合は、アンダーフローコード 0x8000 を用いるものとする。充放電をしていないときは 0。

(22) 積算放電電力量計測値

電気自動車充放電器の放電時の積算電力量を 0.001kWh の単位で示す。プロパティの値域は、0x00000000～0x3B9AC9FF（0～999,999.999kWh）とし、積算電力量のオーバーフロー時は、0x00000000 から再インクリメントするものとする。

(23) 積算放電電力量リセット設定

0x00 をセットすることにより、積算放電電力量計測値をゼロにリセットを行う。

(24) 積算充電電力量計測値

電気自動車充放電器の充電時の積算電力量を 0.001kWh の単位で示す。プロパティの値域は、0x00000000～0x3B9AC9FF（0～999,999.999kWh）とし、積算電力量のオーバーフロー時は、0x00000000 から再インクリメントするものとする。

(25) 積算充電電力量リセット設定

0x00 をセットすることにより、積算充電電力量計測値をゼロにリセットを行う。

(26) 運転モード設定

電気自動車充放電器の運転モードを示す。運転モードとして、充電（0x42）、放電（0x43）、

待機（0x44）、停止（0x47）、その他（0x40）のいずれかを示す。
但し、運転モード（その他）は充放電器が充電、放電、待機、停止のいずれのモードで無い場合を示す。

(27) 系統連系状態
現在の系統と電気自動車充放電器の接続状態（系統連系状態）を示す。
系統連系（逆潮流可）＝0x00, 独立＝0x01,系統連系（逆潮流不可）=0x02

(28) 車載電池の電池残容量 1
電気自動車充放電器に接続された電気自動車に搭載された電池の残容量を Wh で示す。プロパティの値域は、0x00000000～0x3B9AC9FF（0～999,999,999Wh）とする。

(29) 車載電池の電池残容量 2
電気自動車充放電器に接続された電気自動車に搭載された電池の残容量を 0.1Ah で示す。プロパティの値域は、0x0000～0x7FFE（0～3,276.6Ah）とする。

(30) 車載電池の電池残容量 3
電気自動車充放電器に接続された電気自動車に搭載された電池の残容量（SOC: State of Charge）を%で示す。プロパティの値域は、0x00～0x64（0～100%）とする。

(31) 充電量設定値 1
充電する際の電力量を Wh の単位で指定する。プロパティの値域は、0x00000000～0x3B9AC9FF（0～999,999,999Wh）とする。充電をしていない時は 0。

(32) 充電量設定値 2
充電する際の容量を 0.1Ah の単位で指定する。プロパティの値域は 0x0000～0x7FFE（0～3,276.6Ah）とする。充電をしていない時は 0。

(33) 充電電力設定値
充電する際の電力を W の単位で指定する。プロパティの値域は、0x00000000～0x3B9AC9FF（0～999,999,999W）とする。

(34) 放電電力設定値
系統連系の場合に、放電する際の電力を W の単位で指定する。プロパティの値域は、0x00000000～0x3B9AC9FF（0～999,999,999W）とする。放電をしていない時は 0。独立時の場合には、不可応答を返す。

(35) 充電電流設定値

充電する際の電流を 0.1A の単位で指定する。プロパティの値域は、0x0000～0xFFFD（0～6,553.3A）とする。充電をしていない時は 0。

(36) 放電電流設定値

系統連系の場合に、放電する際の電流を 0.1A の単位で指定する。プロパティの値域は、0x0000～0xFFFD（0～6,553.3A）とする。放電をしていない時は 0。独立時の場合には、不可応答を返す。

(37) 定格電圧（独立時）

系統連系状態が「独立」時の定格電圧（カタログ値）をＶの単位で示す。プロパティの値域は、0x0000～0x7FFE（0～32,766V）とする。本プロパティを使用しない場合は定格電圧(EPC=0xD2)が独立時の値を兼ねても良い。

３．３．２０ 電力量メータクラス規定

クラスグループコード　：　0x02

クラスコード　　：0x80

インスタンスコード　：0x01～0x7F (0x00：全インスタンス指定コード)

プロパティ名称	EPC	プロパティ内容 値域(10 進表記)	データ型	データサイズ	単位	アクセスルール	必須	状変時アナウンス	備考
動作状態	0x80	ON／OFF の状態を示す。 ＯＮ＝0x30，ＯＦＦ＝0x31	unsigned char	1 Byte	－	Set Get	 ○	○	
積算電力量計測値	0xE0	積算電力量を 10 進表記において、8 桁で示す。 0x00000000～0x05F5E0FF (0～99,999,999)	unsigned long	4 Byte	0.1 or 0.01 kWh	Get	○		
積算電力量単位	0xE2	積算電力量計測値(0xE0)の単位を示す。 0x01：0.1kWh 0x02：0.01kWh	unsigned char	1 Byte	－	Get	○		
積算電力量計測値履歴 1	0xE3	積算電力量(8 桁)の計測結果履歴を、30 分毎データを過去 24 時間で示す。 0x00000000～0x05F5E0FF (0～99,999,999)	unsigned long ×48	192 Byte	0.1 or 0.01 kWh	Get			
積算電力量計測値履歴 2	0xE4	積算電力量(8 桁)の計測結果履歴を、30 分毎データの 1 日単位データを、過去 45 日で示す。 0x00000000～0x05F5E0FF (0～99,999,999)	unsigned long ×48 ×45	192 Byte × 45	0.1 or 0.01 kWh	GetM			

注1)　状態変化時（状変時）アナウンスの○は、プロパティ実装時には、処理必須を示す。

(1)　動作状態（機器オブジェクトスーパークラスのプロパティを継承）

本クラス固有の機能が、稼動状態であるか否か(ON/OFF)を示す。本クラスを搭載するノードにおいて、ノードの動作開始とともに、本クラスの機能が、稼動を開始する場合は、本プロパティを固定値 0x30（動作状態 ON）で実装することも可能である。

(2)　積算電力量計測値

積算電力量計測値を 10 進表記において 8 桁で示す。積算電力量単位(EPC＝0xE2)のプロパティ値により単位を示す。積算電力量単位(EPC＝0xE2)が、0x01 の場合 0.1kWh の単位、0x02 の場合 0.01kWh の単位を取るものとする。プロパティの値域は、0x00000000～05F5E0FF(0～99,999,999)とし、積算電力量のオーバーフロー時は、0x00000000 から再インクリメントするものとする。

(3)　積算電力量単位

積算電力量計測値(EPC＝0xE0)の単位を示す。プロパティ値が 0x01 の場合、積算電力量計測値(EPC＝0xE0)が 0.1kWh の単位、プロパティ値が 0x02 の場合、積算電力量計測値(EPC＝0xE0)が 0.01kWh の単位を取るものとする。

(4)　積算電力量計測値履歴 1

積算電力量計測値(EPC＝0xE0)の計測結果履歴の 30 分毎データを過去 24 時間分データで示す。積算電力量単位(EPC＝0xE2)のプロパティ値により単位を示し、積算電力量単位(EPC＝0xE2)が、0x01 の場合 0.1kWh の単位、積算電力量単位(EPC＝0xE2)が、0x02 の場合 0.01kWh の単位を取るものとする。30 分毎の積算電力量計測値は、プロパティ名称「現在時刻設定値」(EPC=0x97)で設定する時刻に基づき、毎 0 分、30 分の 8 桁単位の計測値を、0x00000000～05F5E0FF(0～99,999,999)のデータとし、時系列に上位バイトから順にプロパティ値とするものとする。計測値履歴の未計測の時刻のデータに関しては、0xFFFFFFFE を用いるものとする。

(5)　積算電力量計測値履歴 2

積算電力量計測値(EPC＝0xE0)の計測結果履歴を、30 分毎データの 1 日単位データ(4 Byte×48)を 1 配列要素とし、過去 45 日で示す。積算電力量単位(EPC＝0xE2)のプロパティ値により単位を示し、積算電力量単位(EPC＝0xE2)が、0x01 の場合 0.1kWh の単位、積算電力量単位(EPC＝0xE2)が、0x02 の場合 0.01kWh の単位を取るものとする。30 分毎の積算電力量計測値は、プロパティ名称「現在時刻設定値」(EPC=0x97)で設定する時刻に基づき、毎 0 分、30 分の 8 桁単位の計測値を、0x00000000～05F5E0FF(0～99,999,999)のデータとし、45 日前の 0 時 0 分の積算電力量計測値のデータから、時系列に上位バイトから順に 1 日単位データ(4 Byte×48)を 1 配列要素の 45 日分データをプロパティ値とするものとする。計測履歴の未計測の時刻のデータに関しては、0xFFFFFFFE を用いるものとする。

３．３．２８ 一般照明クラス規定

クラスグループコード　：　0x02
クラスコード　　：0x90
インスタンスコード　：0x01～0x7F (0x00：全インスタンス指定コード)

プロパティ名称	EPC	プロパティ内容 値域(10 進表記)	データ型	データサイズ	単位	アクセスルール	必須	状変時ｱﾅｳﾝｽ	備考
動作状態	0x80	ON／OFF の状態を示す。 ＯＮ＝0x30，ＯＦＦ＝0x31	unsigned char	1 Byte	－	Set Get	○ ○	○	
照度レベル設定	0xB0	照度レベルを%で示す。 0x00～0x64(0～100%)	unsigned char	1 Byte	%	Set/ Get			
光色設定	0xB1	光色を設定する 電球色＝0x41，白色＝0x42，昼白色＝0x43，昼光色＝0x44,その他=0x40	unsigned char	1 Byte	－	Set/ Get			
照度レベル段数設定	0xB2	照度レベルを段数で設定し、設定状態を取得する。 0x01～設定可能照度レベル最大値 (暗～明)	unsigned char	1 Byte	－	Set/ Get			
光色レベル段数設定	0xB3	光色レベルを段数で設定し、設定状態を取得する。 0x01～設定可能光色レベル最大値 (電球色～白色)	unsigned char	1 Byte	－	Set/ Get			
設定可能レベル最大値	0xB4	通常灯モード時の照度及び光色設定可能レベル最大値を取得する。 1Byte 目:照度 2Byte 目:光色 0x01～0xFF(1～255 段階) 0x00(機能を搭載していない場合)	unsigned char ×2	2 Byte	－	Get			
常夜灯設定可能レベル最大値	0xB5	常夜灯モード時の照度及び光色設定可能レベル最大値を取得する。 1Byte 目:照度 2Byte 目:光色 0x01～0xFF(1～255 段階) 0x00(機能を搭載していない場合)	unsigned char ×2	2 Byte	－	Get			
点灯モード設定	0xB6	自動／通常灯／常夜灯／カラー灯 自動=0x41，通常灯=0x42，常夜灯=0x43，カラー灯=0x45	unsigned char	1 Byte	－	Set/ Get	○		
通常灯モード時照度レベル設定	0xB7	通常灯モード時の照度レベルを%で示す。 0x00～0x64(0～100%)	unsigned char	1 Byte	%	Set/ Get			
通常灯モード時照度レベル段数設定	0xB8	通常灯モード時照度レベルを段数で設定し、設定状態を取得する。	unsigned char	1 Byte	－	Set/ Get			

		0x01～設定可能照度レベル最大値 （暗～明）							
常夜灯モード時照度レベル設定	0xB9	常夜灯モード時の照度レベルを%で示す。 0x00～0x64(0～100%)	unsigned char	1 Byte	%	Set/ Get			
常夜灯モード時照度レベル段数設定	0xBA	常夜灯モード時照度レベルを段数で設定し、設定状態を取得する。 0x01～設定可能照度レベル最大値 （暗～明）	unsigned char	1 Byte	－	Set/ Get			
通常灯モード時光色設定	0xBB	通常灯モード時光色を設定する 電球色＝0x41, 白色＝0x42, 昼白色＝0x43, 昼光色＝0x44,その他=0x40	unsigned char	1 Byte	－	Set/ Get			
通常灯モード時光色レベル段数設定	0xBC	通常灯モード時光色レベルを段数で設定し、設定状態を取得する。 0x01～設定可能光色レベル最大値 （電球色～白色）	unsigned char	1 Byte	－	Set/ Get			
常夜灯モード時光色設定	0xBD	常夜灯モード時光色を設定する 電球色＝0x41, 白色＝0x42, 昼白色＝0x43, 昼光色＝0x44,その他=0x40	unsigned char	1 Byte	－	Set/ Get			
常夜灯モード時光色レベル段数設定	0xBE	常夜灯モード時光色レベルを段数で設定し、設定状態を取得する。 0x01～設定可能光色レベル最大値 （電球色～白色）	unsigned char	1 Byte	－	Set/ Get			
自動モード時点灯モード状態	0xBF	自動モード時の点灯モード状態を取得する 通常灯=0x42, 常夜灯=0x43,消灯=0x44, カラー灯=0x45	unsigned char	1 Byte	－	Get			
カラー灯モード時RGB設定	0xC0	カラー灯モード時のRGB値を設定し、設定状態を取得する。 1Byte 目:R 2Byte 目:G 3Byte 目:B 0x00～0xFF(0～255) 最低輝度=0x00,最高輝度=0xFF	unsigned char×3	3 Byte	－	Set/Get			
ON タイマ予約設定	0x90	予約入／予約切 予約入＝0x41,予約切＝0x42	unsigned char	1 Byte	－	Set/ Get			
ON タイマ時刻設定値	0x91	タイマ値 HH:MM 0～0x17: 0～0x3B (=0～23):(=0～59)	unsigned char ×2	2 Byte	－	Set/ Get			
OFF タイマ予約設定	0x94	予約入／予約切 予約入＝0x41,予約切＝0x42	unsigned char	1 Byte	－	Set/ Get			
OFF タイマ時刻設定値	0x95	タイマ値 HH:MM 0～0x17: 0～0x3B (=0～23):(=0～59)	unsigned char ×2	2 Byte	－	Set/ Get			

注1） 状態変化時（状変時）アナウンスの○は、プロパティ実装時には、処理必須を示す。

(1) 動作状態（機器オブジェクトスーパークラスのプロパティを継承）

動作状態 ON

照度レベルプロパティ（0xB0、0xB2）が存在しない場合　　：点灯状態

照度レベルプロパティ（0xB0、0xB2）が存在する場合 ：照度レベルが点灯照度に反映されている状態

尚、点灯モードプロパティ（0xB6）の自動モード(0x41）が存在する場合は、自動モードで消灯状態の場合もある。

動作状態 OFF

消灯状態

(2) 照度レベル設定

照明の現在の「点灯モード設定」における照度レベルを%で示す。照度レベルを設定し、設定状態を取得する。実機器の照度レベル設定が%単位より少ない場合、もしくは、多い場合も、必ず、本プロパティで規定する%単位のプロパティ値に実機器のプロパティを割り当てるものとする。

本プロパティと「照度レベル段数設定」(0xB2)とを実装する場合は、両者の値を互いに関連付けしなければならない。

本プロパティで示す照度レベル設定値は、点灯モード設定(0xB6)の自動(0x41)の機能が実装されていない場合、または実装されているが点灯モード設定が通常(0x42)／常夜灯(0x43)の状態をとる場合に、実機器の照度レベル設定値である。また、点灯モード設定が自動(0x41)の状態の時は、取得した時点の照度レベル設定値であることを推奨する。ただし、自動状態であるために、本プロパティで表す照度レベル設定値が不明となってしまう場合に本プロパティが取る値は 0xFD(設定値不明)とする。

動作状態プロパティ(0x80)が OFF(0x31)の場合であっても、本プロパティの有効性は保証されるものとする。

(3) 光色設定

本プロパティの設定により照明の現在の「点灯モード設定」における光色(電球色/白色/昼白色/昼光色/その他 0x40)の設定し、設定状態を取得する。「その他」とは、他のいずれの光色にも該当しない光色である。プロパティ値の取る値については、本クラスを実装する機器が、その機能として取りうるプロパティ値のみを実装すればよいものとする。例えば、本クラスを搭載する実機器が昼白色をその機能として搭載していない場合は、昼白色に対する 0x43 を実装する必要はない。また、本プロパティと「光色レベル段数設定」（EPC=0xB3）とを実装する場合は、両者の値を互いに関連付けなければならない。

本プロパティで示す光色設定値は、点灯モード設定(0xB6)の自動(0x41)の機能が実装されていない場合、または実装されているが点灯モード設定が通常(0x42)／常夜灯(0x43)の状態をとる場合に、実機器の光色設定値である。また、点灯モード設定が自動(0x41)の状態の時は、取得した時点の光色設定値であることを推奨する。ただし、自動状態であるために、本プロパティで表す光色設定値が不明となってしまう場合に本プロパティが取る値は0xFD(設定値不明)とする。
動作状態プロパティ(0x80)がOFF(0x31)の場合であっても、本プロパティの有効性は保証されるものとする。

(4) 照度レベル段数設定

照明の現在の「点灯モード設定」における照度レベルを段数で示す。照度レベルを設定し、設定状態を取得する。設定可能な照度レベルの最大値はEPC=0xB4「設定可能レベル最大値」及びEPC=0xB5「常夜灯設定可能レベル最大値」で取得する。照度レベルの具体的な値は規定しないが、レベル設定値が小さいほど、明るさが暗い状態となり、大きいほど明るい状態となる。プロパティ値のとる値については、本クラスを実装する実機器が、その機能として取りうるプロパティ値のみを実装すればよいものとする。本プロパティと「照度レベル設定」（EPC=0xB0）とを実装する場合は、両者の値を互いに関連付けなければならない。また、本プロパティを実装する場合は、設定可能レベル最大値（0xB4）を実装することを必須とする。また、点灯モード設定(0xB6)の常夜灯(0x43)機能を実装するときは、常夜灯設定可能レベル最大値(0xB5)も実装することを必須とする。
本プロパティで示す照度レベル段数設定値は、点灯モード設定(0xB6)の自動(0x41)の機能が実装されていない場合、または実装されているが点灯モード設定が通常(0x42)／常夜灯(0x43)の状態をとる場合に、実機器の照度レベル段数設定値である。また、点灯モード設定が自動(0x41)の状態の時は、取得した時点の照度レベル段数設定値であることを推奨する。ただし、自動状態であるために、本プロパティで表す照度レベル段数設定値が不明となってしまう場合に本プロパティが取る値は0x00(設定値不明)とする。
動作状態プロパティ(0x80)がOFF(0x31)の場合であっても、本プロパティの有効性は保証されるものとする。

(5) 光色レベル段数設定

照明の現在の「点灯モード設定」における光色レベルを段数で示す。光色レベルを設定し、設定状態を取得する。設定可能な光色レベルの最大値はEPC=0xB4「設定可能レベル最大値」及びEPC=0xB5「常夜灯設定可能レベル最大値」で取得する。照度レベルの具体的な値は規定しないが、レベル設定値が小さいほど、電球色状態となり、大きいほど白色状態となる。プロパティ値のとる値については、本クラスを実装する実機器が、その機能として取りうるプロパティ値のみを実装すればよいものとする。本プロパティと「光色設定」（EPC=0xB1）

とを実装する場合は、両者の値を互いに関連付けなければならない。また、本プロパティを実装する場合は、設定可能レベル最大値（0xB4）を実装することを必須とする。また、点灯モード設定(0xB6)の常夜灯(0x43) 機能を実装するときは、常夜灯設定可能レベル最大値(0xB5)も実装することを必須とする。

本プロパティで示す光色レベル段数設定値は、点灯モード設定(0xB6)の自動(0x41)の機能が実装されていない場合、または実装されているが点灯モード設定が通常(0x42)／常夜灯(0x43)の状態をとる場合に、実機器の光色レベル段数設定値である。また、点灯モード設定が自動(0x41)の状態の時は、取得した時点の光色レベル段数設定値であることを推奨する。ただし、自動状態であるために、本プロパティで表す光色レベル段数設定値が不明となってしまう場合に本プロパティが取る値は 0x00(設定値不明)とする。

動作状態プロパティ(0x80)が OFF(0x31)の場合であっても、本プロパティの有効性は保証されるものとする。

(6) 設定可能レベル最大値

通常灯の設定可能レベル最大値を取得する。本プロパティのデータサイズは 2Byte で、設定可能レベル最大値をそれぞれ 1Byte、255 段階（0x01～0xFF）で表現する。2Byte のうち、1Byte 目は照度レベル段数設定、2Byte 目は光色レベル段数設定を表す。なお、搭載していない機能の設定可能レベル最大値は、0x00 とする。

(7) 常夜灯設定可能レベル最大値

常夜灯の設定可能レベル最大値を取得する。本プロパティのデータサイズは 2Byte で、設定可能レベル最大値をそれぞれ 1Byte、255 段階（0x01～0xFF）で表現する。2Byte のうち、1Byte 目は照度レベル段数設定、2Byte 目は光色レベル段数設定を表す。なお、搭載していない機能の設定可能レベル最大値は、0x00 とする。

(8) 点灯モード設定

照明の自動／通常灯／常夜灯／カラー灯の各点灯モードを設定し、設定状態を取得する。

自動(0x41) ：照度センサや自動調光アルゴリズム等により、照明器具が自動的に
通常灯／常夜灯の選択、照度レベル設定、または照度レベル段数設定、
光色設定、または光色レベル段数設定を制御している状態

通常灯(0x42)：メイン光源が点灯するモード

常夜灯(0x43)：メイン光源ではなく、常夜灯（豆電球など）が点灯するモード

カラー灯(0x45)：カラー灯が点灯するモード

本プロパティを実装する実機器が、その機能として取りうるプロパティ値のみを実装すれば良いものとする。例えば、自動機能を搭載していない場合は、自動に対する 0x41 を実装する必要は無い。

(9) 通常灯モード時照度レベル設定

「点灯モード設定」(EPC=0xB6)が、通常灯モードの場合の照度レベルを%で示す。照度レベルを設定し、設定状態を取得する。実機器の照度レベル設定が%単位より少ない場合、もしくは、多い場合も、必ず、本プロパティで規定する%単位のプロパティ値に実機器のプロパティを割り当てるものとする。

本プロパティを実装する場合は、点灯モード設定(0xB6)の現在の設定が、通常灯モード以外の場合も設定／取得が可能である。また、本プロパティと「通常灯モード時照度レベル段数設定」(0xB8)とを実装する場合は、両者の値を互いに関連付けしなければならない。

動作状態プロパティ(0x80)が OFF(0x31)の場合であっても、本プロパティの有効性は保証されるものとする。

(10) 通常灯モード時照度レベル段数設定

「点灯モード設定」(EPC=0xB6)が、通常灯モードの場合の照度レベルを段数で示す。照度レベルを設定し、設定状態を取得する。設定可能な照度レベルの最大値は EPC=0xB4「設定可能レベル最大値」で取得する。照度レベルの具体的な値は規定しないが、レベル設定値が小さいほど、明るさが暗い状態となり、大きいほど明るい状態となる。プロパティ値のとる値については、本クラスを実装する実機器が、その機能として取りうるプロパティ値のみを実装すればよいものとする。本プロパティと「通常灯モード時照度レベル設定」(EPC=0xB7)とを実装する場合は、両者の値を互いに関連付けなければならない。また、本プロパティを実装する場合は、設定可能レベル最大値 (0xB4) を実装することを必須とする。

本プロパティを実装する場合は、点灯モード設定(0xB6)の現在の設定が、通常灯モード以外の場合も設定／取得が可能である。

動作状態プロパティ(0x80)が OFF(0x31)の場合であっても、本プロパティの有効性は保証されるものとする。

(11) 常夜灯モード時照度レベル設定

「点灯モード設定」(EPC=0xB6)が、常夜灯モードの場合の照度レベルを%で示す。照度レベルを設定し、設定状態を取得する。実機器の照度レベル設定が%単位より少ない場合、もしくは、多い場合も、必ず、本プロパティで規定する%単位のプロパティ値に実機器のプロパティを割り当てるものとする。

本プロパティを実装する場合は、点灯モード設定(0xB6)の現在の設定が、常夜灯モード以外の場合も設定／取得が可能である。また、本プロパティと「常夜灯モード時照度レベル段数設定」(0xBA)とを実装する場合は、両者の値を互いに関連付けしなければならない。

動作状態プロパティ(0x80)が OFF(0x31)の場合であっても、本プロパティの有効性は保証されるものとする。

(12) 常夜灯モード時照度レベル段数設定

「点灯モード設定」(EPC=0xB6)が、常夜灯モードの場合の照度レベルを段数で示す。照度レベルを設定し、設定状態を取得する。設定可能な照度レベルの最大値は EPC=0xB5「常夜灯設定可能レベル最大値」で取得する。照度レベルの具体的な値は規定しないが、レベル設定値が小さいほど、明るさが暗い状態となり、大きいほど明るい状態となる。プロパティ値のとる値については、本クラスを実装する実機器が、その機能として取りうるプロパティ値のみを実装すればよいものとする。本プロパティと「常夜灯モード時照度レベル設定」(EPC=0xB9) とを実装する場合は、両者の値を互いに関連付けなければならない。また、本プロパティを実装する場合は、常夜灯設定可能レベル最大値 (0xB5) を実装することを必須とする。

本プロパティを実装する場合は、点灯モード設定(0xB6)の現在の設定が、常夜灯モード以外の場合も設定／取得が可能である。

動作状態プロパティ(0x80)が OFF(0x31)の場合であっても、本プロパティの有効性は保証されるものとする。

(13) 通常灯モード時光色設定

「点灯モード設定」(EPC=0xB6)が、通常灯モードの場合の光色(電球色 0x41/白色 0x42/昼白色 0x43/昼光色 0x44/その他 0x40)を設定し、設定状態を取得する。「その他」とは、他のいずれの光色にも該当しない光色である。プロパティ値の取る値については、本クラスを実装する機器が、その機能として取りうるプロパティ値のみを実装すればよいものとする。例えば、本クラスを搭載する実機器が昼白色をその機能として搭載していない場合は、昼白色に対する 0x43 を実装する必要はない。

本プロパティを実装する場合は、点灯モード設定(0xB6)の現在の設定が、通常モード以外の場合も設定／取得が可能である。また、本プロパティと「通常灯モード時光色レベル段数設定」(0xBC)とを実装する場合は、両者の値を互いに関連付けしなければならない。

動作状態プロパティ(0x80)が OFF(0x31)の場合であっても、本プロパティの有効性は保証されるものとする。

(14) 通常灯モード時光色レベル段数設定

「点灯モード設定」(EPC=0xB6)が、通常灯モードの場合の光色レベルを段数で示す。光色レベルを設定し、設定状態を取得する。設定可能な光色レベルの最大値は EPC=0xB4「設定可能レベル最大値」で取得する。光色レベルの具体的な値は規定しないが、レベル設定値が小さいほど、電球色状態となり、大きいほど白色状態となる。プロパティ値のとる値については、本クラスを実装する実機器が、その機能として取りうるプロパティ値のみを実装すればよいものとする。また、本プロパティを実装する場合は、設定可能レベル最大値 (0xB4)

を実装することを必須とする。
本プロパティを実装する場合は、点灯モード設定(0xB6)の現在の設定が、通常モード以外の場合も設定／取得が可能である。また、本プロパティと「通常灯モード時光色設定」(0xBB)とを実装する場合は、両者の値を互いに関連付けしなければならない。
動作状態プロパティ(0x80)が OFF(0x31)の場合であっても、本プロパティの有効性は保証されるものとする。

(15) 常夜灯モード時光色設定

「点灯モード設定」(EPC=0xB6)が、常夜灯モードの場合の光色(電球色 0x41/白色 0x42/昼白色 0x43/昼光色 0x44/その他 0x40)を設定し、設定状態を取得する。「その他」とは、他のいずれの光色にも該当しない光色である。プロパティ値の取る値については、本クラスを実装する機器が、その機能として取りうるプロパティ値のみを実装すればよいものとする。例えば、本クラスを搭載する実機器が昼白色をその機能として搭載していない場合は、昼白色に対する 0x43 を実装する必要はない。
本プロパティを実装する場合は、点灯モード設定(0xB6)の現在の設定が、常夜灯モード以外の場合も設定／取得が可能である。また、本プロパティと「常夜灯モード時光色レベル段数設定」(0xBE)とを実装する場合は、両者の値を互いに関連付けしなければならない。
動作状態プロパティ(0x80)が OFF(0x31)の場合であっても、本プロパティの有効性は保証されるものとする。

(16) 常夜灯モード時光色レベル段数設定

「点灯モード設定」(EPC=0xB6)が、常夜灯モードの場合の光色レベルを段数で示す。光色レベルを設定し、設定状態を取得する。設定可能な光色レベルの最大値は EPC=0xB5「常夜灯設定可能レベル最大値」で取得する。光色レベルの具体的な値は規定しないが、レベル設定値が小さいほど、電球色状態となり、大きいほど白色状態となる。プロパティ値のとる値については、本クラスを実装する実機器が、その機能として取りうるプロパティ値のみを実装すればよいものとする。また、本プロパティを実装する場合は、常夜灯設定可能レベル最大値（0xB5）を実装することを必須とする。
本プロパティを実装する場合は、点灯モード設定(0xB6)の現在の設定が、常夜灯モード以外の場合も設定／取得が可能である。また、本プロパティと「常夜灯モード時光色設定」(0xBD)とを実装する場合は、両者の値を互いに関連付けしなければならない。
動作状態プロパティ(0x80)が OFF(0x31)の場合であっても、本プロパティの有効性は保証されるものとする。

(17) 自動モード時点灯モード状態

「点灯モード設定」(EPC=0xB6)が、自動モード(0x41)の場合の、実機器の点灯状態を取得

する。通常灯=0x42、常夜灯=0x43 、消灯=0x44、カラー灯=0x45 のプロパティ値が対応する。

プロパティ値のとる値については、本クラスを実装する実機器が、その機能として取りうるプロパティ値のみを実装すればよいものとする。例えば、本クラスを搭載する実機器が常夜灯をその機能として搭載していない場合は、常夜灯に対する 0x43 を実装する必要はない。

(18) カラー灯モード時 RGB 設定

「点灯モード設定」(EPC=0xB6)が、カラー灯モード(0x45)の場合の、RGB 値を設定し、状態を取得する。RGB 値の具体的な値は規定しない。プロパティ値のとる値については、本クラスを実装する実機器が、その機能として取りうるプロパティ値のみを実装すればよいものとする。本プロパティに設定された値での RGB 設定が不可能な場合は、設定可能な最も近い値が設定される。本プロパティを実装する場合は、点灯モード設定(EPC=0xB6)の現在の設定が、カラー灯モード(0x45)以外の場合も設定／取得が可能である。

(19) ON タイマ予約設定

ON タイマの予約の入、切を設定する。本プロパティは、「ON タイマ時刻設定」と関連する。

予約入＝0x41,予約切＝0x42

動作状態プロパティ(0x80)が OFF(0x31)の場合であっても、本プロパティの有効性は保証されるものとする。

(20) ON タイマ時刻設定値

「ON タイマ予約設定」が入の場合に、ON になる時刻を時 : 0x00～0x17(0～23)、分 : 0x00～0x3B(0～59)で示す。時、分の順に上位バイトからプロパティ値とする。

動作状態プロパティ(0x80)が OFF(0x31)の場合であっても、本プロパティの有効性は保証されるものとする。

(21) OFF タイマ予約設定

OFF タイマの予約の入、切を設定する。本プロパティは、「OFF タイマ時刻設定値」と関連する。

予約入＝0x41,予約切＝0x42

動作状態プロパティ(0x80)が OFF(0x31)の場合であっても、本プロパティの有効性は保証されるものとする。

(22) OFF タイマ時刻設定値

「OFF タイマ予約設定」が入の場合に、OFF になる時刻を時 : 0x00～0x17(0～23)、分 : 0x00～0x3B(0～59)で示す。時、分の順に上位バイトからプロパティ値とする。

動作状態プロパティ(0x80)が OFF(0x31)の場合であっても、本プロパティの有効性は保証されるものとする。

３．３．２９ 単機能照明クラス規定

クラスグループコード　：　0x02
クラスコード　　：0x91
インスタンスコード　：0x01～0x7F (0x00：全インスタンス指定コード)

プロパティ名称	EPC	プロパティ内容 値域(10進表記)	データ型	データサイズ	単位	アクセスルール	必須	状変時ｱﾅｳﾝｽ	備考
動作状態	0x80	ON／OFF の状態を示す。 ON=0x30，OFF=0x31	unsigned char	1 Byte	－	Set Get	○ ○	○	
照度レベル設定	0xB0	照度レベルを%で示す。 0x00～0x64(0～100%)	unsigned char	1 Byte	%	Set/ Get			

注1)　状態変化時（状変時）アナウンスの○は、プロパティ実装時には、処理必須を示す。

本クラスは、一般照明クラス規定でサポートされない照明の場合に使用する。
点灯モードに限らず、動作状態は、ON（点灯時）/ OFF（消灯時）の設定が可能である。

(1)　動作状態（機器オブジェクトスーパークラスのプロパティを継承）
照明機器が ON 状態（点灯状態）であるか、OFF 状態（消灯状態）にあるかを示す。

(2)　照度レベル設定
照明の現在の照度レベルを%で示す。照度レベルを設定し、設定状態を取得する。実機器の照度レベル設定が%単位より少ない場合、もしくは、多い場合も、必ず、本プロパティで規定する%単位のプロパティ値に実機器のプロパティを割り当てるものとする。
動作状態プロパティ(0x80)が OFF(0x31)の場合であっても、本プロパティの有効性は保証されるものとする。

３．３．３１ 電気自動車充電器クラス規定

クラスグループコード　：　0x02
クラスコード　　：0xA1
インスタンスコード　：0x01～0x7F (0x00：全インスタンス指定コード)

プロパティ名称	EPC	プロパティ内容 値域(10進表記)	データ型	データサイズ	単位	アクセスルール	必須	状変時アナウンス	備考
動作状態	0x80	ON／OFFの状態を示す	unsigned char	1 Byte	－	Set		○	
		ON＝0x30，OFF＝0x31				Get	○		
定格充電能力	0xC5	電気自動車充電器における定格充電能力をWで示す	unsigned long	4 Byte	W	Get	○		
		0x00000000～0x3B9AC9FF（0～999,999,999W）							
車両接続・充電可否状態	0xC7	電気自動車充電器における充電の可否を示す	unsigned char	1 Byte	－	Get	○	○	
		不定=0xFF 車両未接続=0x30 車両接続・充電不可=0x40 車両接続・充電可=0x41							
最小最大充電電力値	0xC8	電気自動車充電器への充電電力の最小値および最大値を、それぞれWで示す	unsigned long ×2	8 Byte	W	Get			
		0x00000000～0x3B9AC9FF（0～999,999,999W） 最小充電電力値：最大充電電力値							
最小最大充電電流値	0xCA	電気自動車充電器への充電電流の最小値および最大値を、それぞれ0.1Aで示す	unsigned short ×2	4 Byte	0.1A	Get			
		0x0000～0x7FFE (0～3,276.6A) 最小充電電流値：最大充電電流値							
充電器タイプ	0xCC	電気自動車充電器のタイプを示す	unsigned char	1 Byte	－	Get	○		
		AC_CPLT=0x11 AC_HLC（充電のみ）=0x12 DC_タイプAA（充電のみ）=0x21 DC_タイプBB（充電のみ）=0x31 DC_タイプEE（充電のみ）=0x41 DC_タイプFF（充電のみ）=0x51 *4							
車両接続確認	0xCD	電気自動車充電器と車両との接続状態を確認する	unsigned char	1 Byte	－	Set	○ *1		
		接続確認=0x10							
車載電池の使用容量値1	0xD0	電気自動車充電器に接続された電気自動車の車載電池の容量をWhで示す	unsigned long	4 Byte	Wh	Get	○ *2		
		0x00000000～0x3B9AC9FF（0～999,999,999Wh）							
定格電圧	0xD2	通常時の電気自動車充電器の定格電圧をVで示す	unsigned short	2 Byte	V	Get			

		0x0000～0x7FFE（0～32,766V）							
瞬時充電電力計測値	0xD3	瞬時充電電力をWで示す 0x00000000～0x3B9AC9FF（0～999,999,999W）	signed long	4 Byte	W	Get			
積算充電電力量計測値	0xD8	積算充電電力量を0.001kWhで示す 0x00000000～0x3B9AC9FF（0～999,999.999kWh	unsigned long	4 Byte	0.001 kWh	Get			
積算充電電力量リセット設定	0xD9	積算充電電力量をリセットする リセット=0x00	unsigned char	1 Byte	－	Set			
運転モード設定	0xDA	充電/待機/停止/その他の運転モードを設定する 充電＝0x42，待機＝0x44，停止=0x47，その他=0x40	unsigned char	1 Byte	－	Set /Get	○	○	
車載電池の電池残容量1	0xE2	電気自動車充電器に接続された電気自動車の車載電池の残容量をWhで示す 0x00000000～0x3B9AC9FF（0～999,999,999Wh）	unsigned long	4 Byte	Wh	Get	○ *3		
車載電池の電池残容量3	0xE4	電気自動車充電器に接続された電気自動車の車載電池残容量（%）を示す 0x00～0x64（0～100%）	unsigned char	1 Byte	%	Get	○ *3		
充電電力設定値	0xEB	充電の電力をWで指定する 0x00000000～0x3B9AC9FF（0～999,999,999W）	unsigned long	4 Byte	W	Set/ Get			
充電電流設定値	0xED	充電の電流を0.1Aで指定する 0x0000 ～ 0xFFFD （0 ～ 6,553.3A）	unsigned short	2 Byte	0.1A	Set/ Get			

注1） 状態変化時（状変時）アナウンスの○は、プロパティ実装時には、処理必須を示す。

注2)

*1：車両接続確認は充電器タイプがDC_タイプAAの場合のみ必須とする。

*2：車載電池の使用容量値1は、電気自動車充電器に接続された電気自動車から出力される場合、必須とする。応答できない状況にある場合、不可応答を返す。

*3：車載電池の電池残容量1、車載電池の電池残容量3は、電気自動車充電器に接続された電気自動車から出力される場合、いずれかの搭載を必須とする。応答できない状況にある場合、不可応答を返す。

*4：充電器タイプのDC_タイプAA、DC_タイプBB、DC_タイプEE及びDC_タイプFFは、IEC 62196-3にて規定されているConfiguration AA、Configuration BB、Configuration EE及びConfiguration FFに記載の形状のコネクタを有する電気自動車充電器とする。

※電気自動車充電器は、電気自動車と接続されるEVPS（Electric Vehicle Power System）を含む充電器である。電気自動車充電器に接続される電気自動車は変更可能であるため、電気自動車充電器に接続される電気自動車に応じて、電気自動車充電器の各プロパティ値も変化する。そのため、車両接続・充電可否状態が車両未接続から車両接続に変化した際に、接続する電気

自動車が変わっている可能性があるため、電気自動車と EVPS により定まる諸元に関するプロパティ値を再取得することが望ましい。

例：車載電池の使用容量値 1

なお、本クラスで取り扱う電流、電圧、電力の値は AC として取り扱う事とする。

(1) 動作状態（機器オブジェクトスーパークラスのプロパティを継承）

電気自動車充電器が、状態取得および設定受付が可能な状態（ON 状態）であるか否か（OFF 状態）を示す。ON 状態には 0x30 を、OFF 状態には 0x31 を対応させる。

(2) 定格充電能力

電気自動車充電器の定格充電能力をW単位で示す。プロパティの値域は、0x00000000～0x3B9AC9FF（0～999,999,999W）とする。

(3) 車両接続・充電可否状態

電気自動車充電器における充電の可否を示す。車両と接続されていない場合を 0x30 で示す。車両と接続され、充電不可の場合を 0x40、充電可の場合を 0x41 で示す。

充電を開始しないと車両状態がわからない場合、充電可の状態については、充電開始指示を与えられたときに出力するものとする。車両の未接続、接続状態を判別できない場合は不定状態として 0xFF で示す。

(4) 最小最大充電電力値

電気自動車充電器への充電電力の最小値および最大値を W の単位で示す。それぞれの値域は、0x00000000～0x3B9AC9FF（0～999,999,999W）とし、最小/最大の順に上位 Byte からプロパティ値とする。実機器のプロパティ値が、プロパティの値域を超える場合は、オーバーフローコード 0xFFFFFFFF を用いるものとする。

(5) 最小最大充電電流値

電気自動車充電器への充電電流の最小値および最大値を 0.1A の単位で示す。それぞれの値域は、0x0000～0x7FFE（0～3,276.6A）とし、最小/最大の順に上位 Byte からプロパティ値とする。実機器のプロパティ値が、プロパティの値域を超える場合は、オーバーフローコード 0xFFFF を用いるものとする。

(6) 充電器タイプ

電気自動車充電器のタイプを示す。電気自動車充電器のタイプとして AC_CPLT（0x11）、AC_HLC（充電のみ）（0x12）、DC_タイプ AA（充電のみ）（0x21）、DC_タイプ BB（充電のみ）（0x31）、DC_タイプ EE（充電のみ）（0x41）、DC_タイプ FF（充電のみ）（0x51）のいずれかを示す。

充電器タイプの説明

・AC_CPLT（0x11）：電気自動車へ交流で充電し、電気自動車とはＣＰＬＴ信号にて通信する。
・AC_HLC（充電のみ）（0x12）：電気自動車へ交流で充電し、電気自動車とはＣＰＬＴ信号及びＨＬＣ信号にて通信する。
・DC_タイプ AA（充電のみ）（0x21）：電気自動車へ直流で充電し、電気自動車とはタイプ AA 方式信号通信する。
・DC_タイプ BB（充電のみ）（0x31）：電気自動車へ直流で充電し、電気自動車とはタイプ BB 方式信号通信する。
・DC_タイプ EE（充電のみ）（0x41）：電気自動車へ直流で充電し、電気自動車とはタイプ EE 方式信号通信する。
・DC_タイプ FF（充電のみ）（0x51）：電気自動車へ直流で充電し、電気自動車とはタイプ FF 方式信号通信する。

充電器タイプと車両接続・充電可否状態（0xC7）との関係についての説明

・AC_CPLT（0x11）：常に不定（0xFF）となる。
・AC_HLC（充電のみ）（0x12）：接続された車両が CPLT 機能のみ搭載した車両の場合、不定（0xFF）となる。CPLT 及び HLC 機能を搭載した車両の場合、車両接続・充電可否状態（0xC7）で示した内容となる。
・DC_タイプ AA（充電のみ）（0x21）：車両接続確認（0xCD）にて情報を取得するまでは不定（0xFF）となる。情報取得後は車両接続・充電可否状態（0xC7）で示した内容となる。
・DC_タイプ BB（充電のみ）（0x31）、DC_タイプ EE（充電のみ）（0x41）及び DC_タイプ FF（充電のみ）（0x51）：車両接続・充電可否状態（0xC7）で示した内容となる。

(7)　車両接続確認

電気自動車充電器と電気自動車との接続状態及び充電可否状態の確認を行う。これにより車両接続・充電可否状態[0xC7]情報を取得することができる。
本プロパティは電気自動車充電器タイプが DC_タイプ AA（充電のみ）（0x21）の場合にのみ必須である。

(8)　車載電池の使用容量値 1

電気自動車充電器に接続された電気自動車に搭載された電池の容量を Wh の単位で示す。プロパティの値域は、0x00000000～0x3B9AC9FF（0～999,999,999Wh）とする。

(9)　定格電圧

通常時の電気自動車充電器の定格電圧を V の単位で示す。プロパティの値域は、0x0000～0x7FFE（0～32,766V）とする。

(10) 瞬時充電電力計測値

電気自動車充電器の充電時の瞬時電力を W の単位で示す。プロパティの値域は、充電時には、0x00000000～0x3B9AC9FF（0～999,999,999W）とする。実機器のプロパティ値が、プロパティの値域を超える場合は、オーバーフローコード 0x7FFFFFFF を用いるものとする。充電をしていないときは 0。

(11) 積算充電電力量計測値

電気自動車充電器の充電時の積算電力量を 0.001kWh の単位で示す。プロパティの値域は、0x00000000～0x3B9AC9FF（0～999,999.999kWh）とし、積算電力量のオーバーフロー時は、0x00000000 から再インクリメントするものとする。

(12) 積算充電電力量リセット設定

0x00 をセットすることにより、積算充電電力量計測値をゼロにリセットを行う。

(13) 運転モード設定

電気自動車充電器の運転モードを示す。運転モードとして、充電（0x42）、待機（0x44）、停止（0x47）、その他（0x40）のいずれかを示す。

但し、運転モード（その他）は充電器が充電、待機、停止のいずれのモードで無い場合を示す。

(14) 車載電池の電池残容量 1

電気自動車充電器に接続された電気自動車に搭載された電池の残容量を Wh で示す。プロパティの値域は、0x00000000～0x3B9AC9FF（0～999,999,999Wh）とする。

(15) 車載電池の電池残容量 3

電気自動車充電器に接続された電気自動車に搭載された電池の残容量（SOC: State of Charge）を%で示す。プロパティの値域は、0x00～0x64（0～100%）とする。

(16) 充電電力設定値

充電する際の電気自動車充電器への電力を W の単位で指定する。プロパティの値域は、0x00000000～0x3B9AC9FF（0～999,999,999W）とする。

(17) 充電電流設定値

充電する際の電気自動車充電器への電流を 0.1A の単位で指定する。プロパティの値域は、0x0000～0xFFFD（0～6,553.3A）とする。

３．４ 調理・家事関連機器クラスグループ

本節では、機器オブジェクトの内、調理・家事関連機器クラスグループ（クラスグループ指定コード X1=0x03）に属する ECHONET オブジェクト毎に、コードやプロパティの詳細を規定する。本節で詳細を規定するクラスの一覧を、表８に示し、各クラス毎の詳細を項を設けて規定を示す。クラス規定において、「必須」の記述のあるものは、各クラスを搭載する機器には、そのプロパティとサービスの組み合わせの実装が必須であることを示す。

表８　調理・家事関連機器クラスグループのオブジェクト一覧表

クラス グループコード	クラスコード	クラス名	詳細規定の有無	備考
0x03	0x00～0xAF	For future reserved		
	0xB0	コーヒーメーカ		
	0xB1	コーヒーミル		
	0xB2	電気ポット	○	
	0xB3	電気こんろ		
	0xB4	トースタ		
	0xB5	ジューサ・ミキサ		
	0xB6	フードプロセッサ		
	0xB7	冷凍冷蔵庫	○	
	0xB8	オーブンレンジ	○	
	0xB9	クッキングヒータ	○	
	0xBA	オーブン		
	0xBB	炊飯器	○	
	0xBC	電子ジャー		
	0xBD	食器洗い機		
	0xBE	食器乾燥機		
	0xBF	電気もちつき機		
	0xC0	保温機		
	0xC1	精米機		
	0xC2	自動製パン機		
	0xC3	スロークッカ		
	0xC4	電気漬物機		
	0xC5	洗濯機	○	
	0xC6	衣類乾燥機	○	
	0xC7	電気アイロン		
	0xC8	ズボンプレッサ		
	0xC9	ふとん乾燥機		
	0xCA	小物・くつ乾燥機		
	0xCB	電気掃除機(セントラルクリーナ含む)		
	0xCC	ディスポーザ		
	0xCD	電気蚊取り機		
	0xCE	業務用ショーケース	○	
	0xCF	業務用冷蔵庫		
	0xD0	業務用ホットケース		
	0xD1	業務用フライヤー		
	0xD2	業務用電子レンジ		
	0xD3	洗濯乾燥機	○	
	0xD4	業務用ショーケース向け室外機	○	
	0xD5～0xFF	For future reserved		

注）○：APPENDIX でプロパティ構成を含めた詳細を規定。

３．４．１ 電気ポットクラス規定

クラスグループコード　：　0x03
クラスコード　　：0xB2
インスタンスコード　：0x01～0x7F (0x00：全インスタンス指定コード)

プロパティ名称	EPC	プロパティ内容 値域(10 進表記)	データ型	データサイズ	単位	アクセスルール	必須	状変時ｱﾅｳﾝｽ	備考
動作状態	0x80	ON／OFF の状態を示す。 ＯＮ＝0x30，ＯＦＦ＝0x31	unsigned char	1 Byte	－	Set Get	 ○	○	
蓋開閉状態	0xB0	蓋開／閉状態 蓋開＝0x41，蓋閉＝0x42	unsigned char	1 Byte	－	Get			
湯切れ警告状態	0xB1	電気ポットの湯切れ状態を通知する 湯切れ有り＝0x41 湯切れなし＝0x40	unsigned char	1 Byte	－	Get		○	
沸騰設定	0xB2	沸騰設定 沸騰開始＝0x41 沸騰停止／保温＝0x42	unsigned char	1 Byte	－	Set/ Get			
沸騰／保温モード設定	0xE0	クエン酸洗浄／通常保温／省エネ保温を示す。 クエン酸洗浄＝0x41, 通常保温＝0x42,省エネ保温＝0x43	unsigned char	1 Byte	－	Set/ Get			
保温温度設定値	0xE1	保温温度設定値を℃で示す。 0x00～0x64(0～100)	unsigned char	1 Byte	℃	Set/ Get			
出湯状態	0xE2	出湯状態 出湯有＝0x41，出湯無＝0x42	unsigned char	1 Byte	－	Get		○	
ロック状態	0xE3	出湯ロック状態 ロック有＝0x41，ロック無＝0x42	unsigned char	1 Byte	－	Get			

注）状態変化時（状変時）アナウンスの○は、プロパティ実装時には、処理必須を示す。

(1) 動作状態（機器オブジェクトスーパークラスのプロパティを継承）

本クラス固有の機能が、稼動状態であるか否か(ON/OFF)を示す。尚、本クラスを搭載するノードにおいて、ノードの動作開始とともに、本クラスの機能が、稼動を開始する場合は、本プロパティを固定値 0x30（動作状態 ON）で実装することも可能である。

(2) 蓋開閉状態

電気ポット（ジャーポット）の蓋の開閉状態を示す。
蓋開状態は、0x41。蓋閉状態は、0x42 を用いるものとする。

(3) 湯切れ警告状態

本プロパティの値は、電気ポットが湯切れ状態になった場合に値が 0x40 から 0x41 に遷移

するものとする。また電気ポットに水、あるいはお湯が入れられたときに、0x41 から 0x40 へ復帰するものとする。

(4) 沸騰設定

電気ポットの沸騰開始、沸騰停止／保温状態を設定する。プロパティ値としては、それぞれ 0x41,、0x42 が対応するものとする。なお、本プロパティは、電気ポットが沸騰動作を終了すると、その値は自動的に 0x42 に遷移しなくてはならない。

(5) 沸騰／保温モード

電気ポット(ジャーポット)のクエン酸洗浄／通常保温／省エネ保温の設定を示す。
それぞれの運転モードにそれぞれ、順に 0x41/0x42/0x43 のプロパティ値が対応するものとする。
プロパティ値のとる値については、本クラスを搭載する実機器が、その機能として持つ機能に対応するプロパティ値のみを実装すればよいものとする。

(6) 保温温度設定値

保温温度設定値を℃の単位で示す。
プロパティ値の範囲は、0x00～0x64 (0～100℃)とする。

(7) 出湯状態

出湯操作の状態を、出湯操作有:0x41、出湯操作無:0x42 で示す。具体的には、人の操作により湯が出ている状態が出湯操作有であり、通常何も湯の出ていない状態は出湯操作無となる。

(8) ロック状態

電気ポット（ジャーポット）操作のロック状態を、ロック有：0x41、ロック無：0x42 で示す。

３．７　ＡＶ関連機器クラスグループ

本節では、機器オブジェクトの内、ＡＶ関連機器クラスグループ（クラスグループ指定コード X1=0x06）に属する ECHONET オブジェクト毎に、コードやプロパティの詳細を規定する。本節で詳細を規定するクラスの一覧を、表１１に示し、各クラス毎の詳細を項を設けて規定を示す。クラス規定において、「必須」の記述のあるものは、各クラスを搭載する機器には、そのプロパティとサービスの組み合わせの実装が必須であることを示す。

表１１　ＡＶ関連機器クラスグループのオブジェクト一覧表

クラスグループコード	クラスコード	クラス名	詳細規定の有無	備考
0x06	0x00	For future reserved		
	0x01	ディスプレー	○	
	0x02	テレビ	○	
	0x03	オーディオ	○	
	0x04	ネットワークカメラ	○	
	0x05～0xFF	For future reserved		

注）○：APPENDIX でプロパティ構成を含めた詳細を規定。

３．７．１ ディスプレークラス規定

クラスグループコード　　：0x06
クラスコード　　：0x01

本クラスは AV 関連機器クラスグループとしてディスプレー機器の文字ディスプレー機能部(表示、表示制御、表示データバッファリング等々)、または任意のクラスグループに属しディスプレー機能を有する機器上の文字ディスプレー機能部に適用する。他の文字ディスプレー機能用プロパティ(表示文字属性<書体、サイズ、色等>,ディスプレー上での表示方法、表示位置等々)は必要に応じて追加する。

本クラスは、具体的には文字表示用専用ディスプレーやクラス全般の文字ディスプレー部(液晶表示部)などに適用する。

プロパティ名称	EPC	プロパティ内容 値域(10 進表記)	データ型	データサイズ	単位	アクセスルール	必須	状変時アナウンス	備考
動作状態	0x80	ON/OFF 状態を表す ON=0x30, OFF=0x31	unsigned char	1 Byte	－	Set Get	Ⓜ ○	○	
表示制御設定	0xB0	文字表示・非表示を設定し、設定状態を取得する 表示=0x30, 非表示=0x31	unsigned char	1 Byte	－	Set/Get			
文字列設定受付可能状態	0xB1	伝達文字列を受付けられる状態にあるか否かを示す レディ=0x30, ビジー=0x31	unsigned char	1 Byte	－	Get	○	○	
表示可能文字コード	0xB2	表示可能文字コードをビットマップで示す。 ビット 0　ANSI X3.4 搭載 1 非搭載　0 ビット 1　Shift –JIS 搭載 1 非搭載　0 ビット 2　JIS 搭載 1 非搭載　0 ビット 3　日本語 EUC 搭載 1 非搭載　0 ビット 4　UCS-4 搭載 1 非搭載　0 ビット 5　UCS-2 搭載 1 非搭載　0 ビット 6　Latin -1 搭載 1 非搭載　0 ビット 7　UTF-8 搭載 1 非搭載　0 ビット 8 以上: for future reserved　0	unsigned char × 2	2 Byte	－	Get	○		
伝達文字列設定	0xB3	ユーザに伝達するべき文字列、その	unsigne	Max	－	Set	○		

		伝達文字列長及び使用文字コードを設定し、設定保持値を取得する。伝達文字バイト列データ長を最上位バイトに、次バイトに使用文字コード、次バイトは 0x00(for future reserved),次に伝達文字列先頭バイト、伝達文字列末尾バイトを最下位バイトに設定する。 1 バイト目：伝達文字バイトコード列データ長の 16 進換算値 2 バイト目：使用文字コード 3 バイト目：for future reserved 4 バイト目以降：Max244Byte 分の伝達文字列バイトコード列 文字コードは以下のコードを使用する。 ANSI X3.4=0x01 Shift –JIS=0x02 JIS =0x03 日本語 EUC=0x04 UCS-4 =0x05 UCS-2 =0x06 Latin -1=0x07 UTF-8=0x08 0x09 以上=for future reserved	d char × Max 247	247 Byte		Get			
受付け伝達文字列長	0xB4	設定された伝達文字列で最新の保持文字列の総バイト数を示す。 1 バイト目：0x00-0xF4 2 バイト目： 0x00(for future reserved)	unsigned char	2Byte	－	Get	○		

注1)　状態変化時（状変時）アナウンスの○は、プロパティ実装時には、処理必須を示す。

(1)動作状態（スーパークラスのプロパティを継承）

ディスプレーの ON／OFF を設定し、動作状態を取得する。ON／OFF にそれぞれ、0x30/0x31 のプロパティ値が対応するものとする。エネルギーサービスに対応する場合、本プロパティの「Set」の搭載は必須とする。

(2) 表示制御設定

本プロパティの Set により、本クラスの文字表示・非表示を設定し、Get により設定状態を取得する。0x30/0x31 のプロパティ値がそれぞれ文字表示／非表示に対応するものとする。なお、本プロパティは表示／非表示のみを切り替える機能であり、本プロパティが非表示(0x31)の場合においても、文字列設定受付可能状態プロパティが 0x30 の場合は、伝達文字列設定プロパティの Set を有効としなければならない。

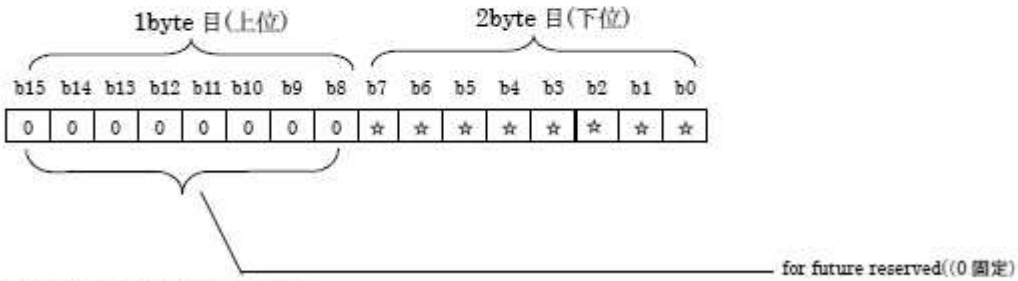
(3) 文字列設定受付可能状態

本プロパティの Get により、本クラスが伝達文字列設定プロパティ設定を受付けられるか否かの状態

を示す。0x30/0x31 のプロパティ値がそれぞれ受け付け可能状態(レディ)／受け付け不可状態(ビジー)に対応するものとする。

(4) 表示可能文字コード

本プロパティの Set により、本クラスとして表示可能な文字コード(符号化文字集合)の搭載有無一覧をビットマップで示す。ビットが 0 の場合は当該実現方法が搭載されていないことを、1 の場合は搭載されていることを示す。



各ビットの意味は以下の通りである。

ビット 0 ANSI X3.4 搭載 1 非搭載 0
ビット 1 Shift - JIS 搭載 1 非搭載 0
ビット 2 JIS 搭載 1 非搭載 0
ビット 3 日本語 EUC 搭載 1 非搭載 0
ビット 4 UCS-4 搭載 1 非搭載 0
ビット 5 UCS-2 搭載 1 非搭載 0
ビット 6 Latin -1 搭載 1 非搭載 0
ビット 7 UTF-8 搭載 1 非搭載 0
ビット 8-15:for future reserved として 0 で固定する。

各文字コード詳細規定は以下の規格を参照のこと。

・ANSI X3.4 : American National Standards Institute, "Coded character set -- 7-bit American national standard code for information interchange", ANSI X3.4-1986.(ASCII)

・Shift-JIS : JIS X 0208:1997 「7 ビット及び 8 ビットの 2 バイト情報交換用符号化漢字集合」

・JIS : ISO/IEC 2022 :1994 Information technology -- Character code structure and extension techniques, ISO-2022-JP (JIS X 0208:1997)

・日本語 EUC : ISO/IEC 2022 :1994 Information technology -- Character code structure and extension techniques, ISO-2022-JP (JIS X 0208:1997)

・UCS-4 ,UCS-2 : ISO/IEC 10646-1:2000 Information technology -- Universal Multiple-Octet Coded Character Set (UCS) -- Part 1: Architecture and Basic Multilingual Plane

・Latin-1 : ISO/IEC 8859-1:1998 Information technology -- 8-bit single-byte coded graphic character sets -- Part 1: Latin alphabet No. 1

・UTF-8 : RFC 3629 「UTF-8,a transformation format of ISO 10646」

(5)伝達文字列設定

本プロパティの Set により、本クラスのユーザに伝達するべき文字列とその文字列長およびその文字コードを設定する。文字列設定受付可能状態プロパティが 0x30 の時は、本プロパティの Set を有効としなければならない。プロパティ内容は伝達文字列のバイトコード列長の 16 進換算値を最上位バイトに、2 バイト目に伝達文字列の文字コード，3 バイト目 0x00(for future reserved),4 バイト目に伝達文字列先頭バイトコードを、伝達文字列末尾バイトコードを最下位バイトとする。伝達文字列には制御コードも含んでよい。本プロパティの Get により、1 バイト目に本プロパティに Set された伝達文字列バイトコード列長 16 進換算値，2 バイト目に Set された伝達文字列文字コード，3 バイト目 0x00(for future reserved)および 4 バイト目以降に設定に成功した伝達文字列バイトコード列が取得できる。文字コードプロパティ内容は、ANSI X3.4=0x01,Shift - JIS=0x02, JIS =0x03,日本語 EUC=0x04,UCS-4 =0x05, UCS-2 =0x06, Latin-1=0x07,UTF-8=0x08 を表す。本クラスは受信した伝達文字列を、本プロパティの文字コード設定値を文字コードとして処理する。動作状態プロパティ（0x80）が OFF(0x31)の場合であっても、本プロパティの有効性は保証されるものとする。本プロパティのバイトオーダは下図に従う。

1byte 目(最上位)	2byte 目	3byte 目	4byte 目		N byte 目(最下位)
第 4 から第 N バイト目迄のバイト数(SET 時)	伝達文字列設定文字コード	0x00(for future reserved)	伝達文字列コード先頭バイト	・・・・・・・・・	伝達文字列コード末尾バイト

本プロパティの具体例を以下に示す。

伝達文字列「ECHONET」を ANSI X3.4 で Set する場合

- 0x0701004543484F4E4554 をプロパティ内容とする。

その後 Get した値は，

- 通常成功時:0x0701004543484F4E4554
- 失敗時:0x0701004543484F4E(メモリ不足などにより末尾 2 バイト分書き込み失敗)
- 成功時:0x04020031323334(別ノードが 0x04020031323334 を追加 Set)

等。

(6)受付け伝達文字列長

本プロパティの Get により、第 1 バイトとして本クラスの伝達文字列設定により保持されている最新の伝達文字列のデータ長が取得できる。伝達文字列設定プロパティを Get して得られる第 1 バイト(バイトコード列長のバイト値)とは必ずしも一致しない。また、本プロパティ内容は伝達文字列設定プロパティに連動して更新されるものとする。本プロパティの第 2 バイト目は 0x00(for future reserved)とする。なお、書き込む動作状態プロパティ（0x80）が OFF(0x31)の場合であっても、本プロパティの有効性は保証されるものとする。

３．７．２ テレビクラス規定

クラスグループコード　　：0x06
クラスコード　　：0x02

本クラスはテレビ受像機全般に適用する。本クラス固有の機能は必要に応じて追加する。

プロパティ名称	EPC	プロパティ内容 値域(10 進表記)	データ型	データサイズ	単位	アクセスルール	必須	状変時アナウンス	備考
動作状態	0x80	クラスグループコード 0x 06, クラスコード 0x 01,ディスプレークラス参照	unsigned char	1 Byte	－	Set Get	Ⓔ ○	○	
表示制御設定	0xB0	クラスグループコード 0x 06, クラスコード 0x 01,ディスプレークラス参照	unsigned char	1 Byte	－	Set/ Get			
文字列設定受付可能状態	0xB1	クラスグループコード 0x 06, クラスコード 0x 01,ディスプレークラス参照	unsigned char	1 Byte	－	Get	Ⓜ Ⓢ	○	
表示可能文字コード	0xB2	クラスグループコード 0x 06, クラスコード 0x 01,ディスプレークラス参照	unsigned char× 2	2 Byte	－	Get	Ⓜ Ⓢ		
伝達文字列設定	0xB3	クラスグループコード 0x 06, クラスコード 0x 01,ディスプレークラス参照	unsigned char× Max 247	Max 247 Byte	－	Set Get	○		
受付け伝達文字列長	0xB4	クラスグループコード 0x 06, クラスコード 0x 01,ディスプレークラス参照	unsigned char	2Byte	－	Get	Ⓜ Ⓢ		

注1)　状態変化時（状変時）アナウンスの○は、プロパティ実装時には、処理必須を示す。

(1) 動作状態（スーパークラスを継承）

TV 受像機の ON／OFF を設定し、動作状態を取得する。ON／OFF にそれぞれ、0x30/0x31 のプロパティ値が対応するものとする。エネルギーサービスに対応する場合、本プロパティの「Set」の搭載は必須とする。

(2) 表示制御設定

説明はクラスグループコード 0x 06, クラスコード 0x 01,ディスプレークラス参照。

(3) 文字列設定受付可能状態

説明はクラスグループコード 0x 06, クラスコード 0x 01,ディスプレークラス参照。
快適支援サービスとセキュリティサービスに対応する場合、本プロパティの搭載は必須とする。

(4) 表示可能文字コード

説明はクラスグループコード 0x 06, クラスコード 0x 01,ディスプレークラス参照。

快適支援サービスとセキュリティサービスに対応する場合、本プロパティの搭載は必須とする。

(5) 伝達文字列設定

説明はクラスグループコード 0x 06, クラスコード 0x 01,ディスプレークラス参照。
本プロパティの「Set」の搭載は必須とする。

(6) 受付け伝達文字列長

説明はクラスグループコード 0x 06, クラスコード 0x 01,ディスプレークラス参照。
快適支援サービスとセキュリティサービスに対応する場合、本プロパティの搭載は必須とする。

３．７．４ ネットワークカメラ規定

クラスグループコード　：　0x06

クラスコード　　：0x04

インスタンスコード　：0x01～0x7F (0x00：全インスタンス指定コード)

プロパティ名称	EPC	プロパティ内容 値域(10 進表記)	データ型	データサイズ	単位	アクセスルール	必須	状変時アナウンス	備考
動作状態	0x80	ON/OFF 状態を表す ON=0x30, OFF=0x31	unsigned char	1Byte	-	Set Get	 ○	○	
静止画撮影設定受付可能状態	0xC0	静止画撮影を受付けられる状態にあるか否かを表す。 レディ = 0x30, ビジー = 0x31	unsigned char	1Byte	-	Get	○		
静止画撮影設定	0xC1	静止画撮影を設定する 静止画撮影 = 0x30	unsigned char	1Byte	-	Set	○		
転送設定	0xD0	撮影データの転送方法を設定し、設定状態を取得する。 ビットマップで示す。 設定有効の場合を１、 設定無効の場合は０とする。 ビット０：ローカルストレージ転送 ビット１：リモートストレージ転送 ビット２：メール転送 ビット３～７：for future reserved	unsigned char	1Byte	-	Set/ Get		○	

注1)　状態変化時（状変時）アナウンスの○は、プロパティ実装時には、処理必須を示す。

(1) 動作状態(スーパークラスのプロパティを継承)

ネットワークカメラの電源の ON/OFF を設定し、動作状態を取得する。ON/OFF それぞれに 0x30/0x31 のプロパティ値が対応するものとする。

(2) 静止画撮影設定受付可能状態

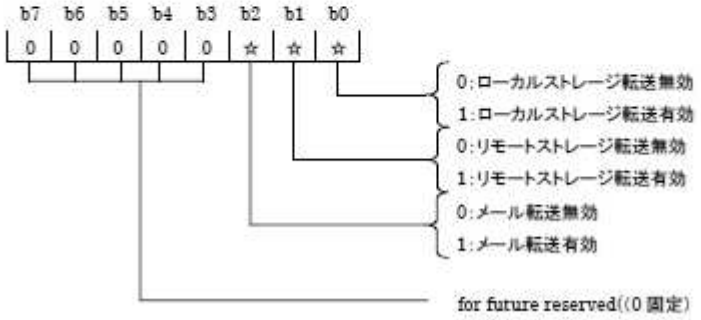
本プロパティの Get により、本クラスが静止画撮影設定プロパティ設定を受付けられるか否かの状態を取得できる。0x30/0x31 のプロパティ値がそれぞれ受け付け可能状態(レディ)／受け付け不可状態(ビジー)に対応するものとする。

(3) 静止画撮影設定

本プロパティの Set により、静止画撮影が実行される。撮影は本クラスを搭載した機器の撮影設定に従い行われる。具体的な撮影設定方法は機器依存とし、特に規定しない。

プロパティ値書き込み要求(SetI/SetC)により、静止画撮影(0x30)が設定された後、撮影可能な場合は、プロパティ値書き込み応答(Set_Res)を返す。撮影不可能な場合、プロパティ値書き込み不可応答(SetI_SNA/SetC_SNA)を返す。

EPC 0xC1「静止画撮影設定」で撮影されたデータの転送方法の一覧をビットマップで示す。実現方法として、ローカルストレージ転送/リモートストレージ転送/メール転送を規定する。以下に詳細を示す。
ローカルストレージ転送とは、機器に内蔵されている記憶装置または機器に直接接続されたUSBハードディスクなどの外部記憶装置にデータを転送することを示す。
リモートストレージ転送とは、ネットワークを介してアクセスできる記憶装置にデータを転送することを示す。
メール転送とは、メールサービスを介してデータを転送することを示す。
ビットが0の場合は当該転送方法が無効になっていることを、1の場合は有効になっていることを示す。
当該転送方法が実施できない場合は、0(無効)に設定されるものとする。



付録１　プロパティマップ記述形式

プロパティの数が、16より少ない場合には下記（1）の記述形式に従い、16以上の場合には下記（2）の記述形式に従うものとする。

記述形式（1）
1バイト目　　：プロパティの数。バイナリ表示。
2バイト目以降　　：プロパティのコード（1バイトコード）をそのまま列挙する。

スイッチクラス(0x05FD)を搭載しているECHONETノードを例にしてプロパティマップ記述形式（1）の例を示す

NO	プロパティ名称	EPC
1	動作状態	0x80
2	設置場所	0x81
3	規格Version情報	0x82
4	識別番号	0x83
5	異常発生状態	0x88
6	メーカコード	0x8A
7	状変ｱﾅｳﾝｽﾌﾟﾛﾊﾟﾃｨﾏｯﾌﾟ	0x9D
8	Setプロパティマップ	0x9E
9	Getプロパティマップ	0x9F
10	接続機器	0xE0

1バイト目はプロパティの数が10個であるので0x0A、2バイト目以降は上記EPCそのまま列挙するため、
「0x0A，0x80，0x81，0x82，0x83，0x88，0x8A，0x9D，0x9E，0x9F，0xE0」のようになる。

記述形式（2）

1バイト目　　：プロパティの数。バイナリ表示。

2～17バイト目　　：下図の16バイトのテーブルにおいて、存在するプロパティコードを示すビット位置に1をセットして2バイト目から順に列挙する。

	ビット 7（上位）	ビット 6	ビット 5	ビット 4	ビット 3	ビット 2	ビット 1	ビット 0（下位）
2バイト目	0xF0	0xE0	0xD0	0xC0	0xB0	0xA0	0x90	0x80
3バイト目	0xF1	0xE1	0xD1	0xC1	0xB1	0xA1	0x91	0x81
4バイト目	0xF2	0xE2	0xD2	0xC2	0xB2	0xA2	0x92	0x82
5バイト目	0xF3	0xE3	0xD3	0xC3	0xB3	0xA3	0x93	0x83
6バイト目	0xF4	0xE4	0xD4	0xC4	0xB4	0xA4	0x94	0x84
7バイト目	0xF5	0xE5	0xD5	0xC5	0xB5	0xA5	0x95	0x85
8バイト目	0xF6	0xE6	0xD6	0xC6	0xB6	0xA6	0x96	0x86
9バイト目	0xF7	0xE7	0xD7	0xC7	0xB7	0xA7	0x97	0x87
10バイト目	0xF8	0xE8	0xD8	0xC8	0xB8	0xA8	0x98	0x88
11バイト目	0xF9	0xE9	0xD9	0xC9	0xB9	0xA9	0x99	0x89
12バイト目	0xFA	0xEA	0xDA	0xCA	0xBA	0xAA	0x9A	0x8A
13バイト目	0xFB	0xEB	0xDB	0xCB	0xBB	0xAB	0x9B	0x8B
14バイト目	0xFC	0xEC	0xDC	0xCC	0xBC	0xAC	0x9C	0x8C
15バイト目	0xFD	0xED	0xDD	0xCD	0xBD	0xAD	0x9D	0x8D
16バイト目	0xFE	0xEE	0xDE	0xCE	0xBE	0xAE	0x9E	0x8E
17バイト目	0xFF	0xEF	0xDF	0xCF	0xBF	0xAF	0x9F	0x8F

注）各ビット値＝0：プロパティ無し、＝1：プロパティ有りを示す。

家庭用エアコンクラス（0x0130)を搭載している ECHONET ノードを例にしてプロパティマップ記述形式（2）の例を示す。

NO	プロパティ名称	EPC	対応するプロパティマップのビット
1	動作状態	0x80	2バイト目のビット0
2	設置場所	0x81	3バイト目のビット0
3	規格Version情報	0x82	4バイト目のビット0
4	識別番号	0x83	5バイト目のビット0
5	電流制限設定	0x87	9バイト目のビット0
6	異常発生状態	0x88	10バイト目のビット0
7	異常内容	0x89	11バイト目のビット0
8	メーカコード	0x8A	12バイト目のビット0
9	事業場コード	0x8B	13バイト目のビット0
10	商品コード	0x8C	14バイト目のビット0
11	製造番号	0x8D	15バイト目のビット0
12	製造年月日	0x8E	16バイト目のビット0
13	節電動作設定	0x8F	17バイト目のビット0

14	ON タイマ予約設定	0x90	2 バイト目のビット 1
15	積算運転時間	0x9A	12 バイト目のビット 1
16	SetM プロパティマップ	0x9B	13 バイト目のビット 1
17	GetM プロパティマップ	0x9C	14 バイト目のビット 1
18	状変ｱﾅｳﾝｽﾌﾟﾛﾊﾟﾃｨﾏｯﾌﾟ	0x9D	15 バイト目のビット 1
19	Set プロパティマップ	0x9E	16 バイト目のビット 1
20	Get プロパティマップ	0x9F	17 バイト目のビット 1
21	運転モード設定	0xB0	2 バイト目のビット 3
22	温度設定値	0xB3	5 バイト目のビット 3

上記のプロパティが ECHONET ノードで公開されている場合、

1 バイト目はプロパティの数が 22 個であるので 0x16、2 バイト目は 0x80, 0x90, 0xB0 のプロパティが公開されており、対応するビットは「ビット 0」「ビット 1」「ビット 3」となるので 0x0B =b'00001011'。3 バイト目、4 バイト目、9 バイト目、10 バイト目、11 バイト目は 0x81、0x82、0x87、0x88、0x89 のプロパティが公開されており、対応するビットは「ビット 0」となるので 0x01。5 バイト目は 0x83、0xB3 のプロパティが公開されており、対応するビットは「ビット 0」「ビット 3」となるので 0x09 =b'00001001'、12 バイト目～17 バイト目は 0x8A、0x9A、0x8B、0x9B、0x8C、0x9C、0x8D、0x9D、0x8E、0x9E、0x8F、0x9F のプロパティが公開されており、対応するビットは「ビット 0」「ビット 1」となるので 0x03=b'00000011'。

以上より、プロパティマップ記述形式は

「0x16, 0x0B, 0x01, 0x01, 0x09, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03」となる。

この章のまとめ

いかがでしたでしょうか。たいへん良くできている規格（仕様）だと思いませんか。この規格を設計するためには、該当する設備・機器の内容を良く理解していることが大切ですが、逆にこの規格を利用すると、該当の機器の特徴が良く分かります。この章の成果を活かして後の章では、モデルスマート家電開発に取り組みます。

３章　スマート家電の開発環境

この章では、スマート家電を開発する際に、利用できるいくつかの環境を、ECHONET Lite 対応機器に関連する SDK と、後に利用するソフトウエア開発環境（マイコン・PC）に分けて説明します。

３章１節　ECHONET Lite 対応機器開発用 SDK

HEMS（ECHONET Lite）認証センターでは、ECHONET Lite 対応機器開発に利用できる各種開発キット（SDK）のダウンロード配布を行っています。下記ページの開発キットのリンクをたどると、SDK の一覧ページが開きます。

２０１５年１１月１２日現在、ここに登録されている SDK は全部で１３あります。（下表）

NO	SDK の概要
1	SSNG for iOS V1.10 ◆ECHONET Lite 機器の動作確認ができる SDK SDK ID ： 00025 登録日/ Registered Date ： 2015. 11. 4. サイズ/ Size ： 26MB メーカー / Maker ： 神奈川工科大学 開発言語 / Programming Language ： SWIFT

	ターゲットOS / Target OS ： iOS ターゲットCPU / Target CPU ： iOS device 概要 / Summary ： SSNG for iOSはiOSデバイスで動作するECHONET Liteコマンド送受信アプリケーションです。 特徴・使用方法 ： ソースコードを公開しているので、Mac上のXcodeでビルドしてiPhoneにインストールしてご利用ください。
2	Super Speed Node Generator for ECHONET Lite Ver.3.0.1 ◆ECHONET Lite機器の動作確認ができるSDK SDK ID ： 00024 登録日/ Registered Date ： 2015.10.26. サイズ/ Size ： 139.1MB メーカー / Maker ： 神奈川工科大学 開発言語 / Programming Language ： NA ターゲットOS / Target OS ： Windows ターゲットCPU / Target CPU ： Pentium III/Celeron 866 MHz または同等のプロセッサ 概要 / Summary ： SSNGはLabVIEWで開発されたECHONET Liteパケットジェネレータです。(ECHONET Lite規格書 Ver1.10、APPENDIX Dに対応) 特徴・使用方法 ： Super Speed Node Generator for ECHONET Lite Ver.3.0.1では、ドメイン上のECHONET Lite機器をSearchボタンで探すことができ、発見された機器に任意のECHONET Liteフレームを送信することが可能です。また、テキストフィル形式で記述された任意のプロパティを用意することで、マウス操作だけでECHONET Liteフレームを生成、送受信することができます。
3	ECHONET Lite SDK Version 6.0.0 ◆ECHONET Liteアプリケーションの動作が確認できるSDK SDK ID ： 00023 登録日/ Registered Date ： 2015.10.23. サイズ/ Size ： 9.3MB メーカー / Maker ： 神奈川工科大学（日新システムズ委託） 開発言語 / Programming Language ： C ターゲットOS / Target OS ： Linux ターゲットCPU / Target CPU ： INTEL x86系 概要 / Summary ： PC上でアプリケーションの動作確認が30分間のみ可能な無償版SDKです。 なお、有償版は株式会社日新システムズ(http://www.co-nss.co.jp)より提供されています。

	特徴・使用方法　：　ECHONET Lite SPECIFICATION 1.01、1.10、ECHONET 機器オブジェクト詳細規定 Release A、B、C に対応しており、全機器オブジェクトのアプリケーション開発が可能です。開発は統合開発環境である Eclipse で行い、GUI 上での開発、コンパイル、実行、デバッグが可能です。
4	Temperature Sensor Emulator ◆温度センサーを実装した機器のエミュレータ（コントローラの動作が確認できる）SDK SDK ID ： 00022 登録日/ Registered Date ： 2015. 8. 24. サイズ/ Size ： 216.7KB メーカー / Maker ： 神奈川工科大学 開発言語 / Programming Language ： Java, OPEN ECHO for Processing ターゲット OS / Target OS ： Linux, Windows, Android, Mac OS ターゲット CPU / Target CPU ： Java, OPEN ECHO for Processing 概要 / Summary ： Temperature Sensor Emulator は ECHONET Lite を実装した「温度センサ」エミュレターです。 特徴・使用方法　：　OPENECHO for Processing のソースコードで配布され Windows/Mac/Linux 上で動作可能です。GUI から温度設定が可能です。
5	Light Emulator -1 ◆一般照明機器のエミュレータ（コントローラの動作が確認できる）SDK SDK ID ： 00019 登録日/ Registered Date ： 2015. 3. 10. サイズ/ Size ： 251.5KB メーカー / Maker ： 神奈川工科大学（スマートハウス研究センター） 開発言語 / Programming Language ： OpenECHO ターゲット OS / Target OS ： Linux, Windows, Mac OS ターゲット CPU / Target CPU ： NA 概要 / Summary ： 一般照明のエミュレーターです。照度レベルや色を変更できます。 特徴・使用方法　：　OpenECHO のソースコードで配布されます。
6	KAIT-4S ~EZ~ for iOS 20150224 ◆ECHONET Lite 機器のコントローラや連携アプリを開発する SDK SDK ID ： 00018 登録日/ Registered Date ： 2015. 2. 26. サイズ/ Size ： 4.8MB

	メーカー / Maker ： 神奈川工科大学（ユビキタス委託） 開発言語 / Programming Language ： Objective-C, Swift ターゲットOS / Target OS ： iOS ターゲットCPU / Target CPU ： iPhone, iPad 概要 / Summary ： 本SDKは、iOS機器上で動作するコンローラアプリケーションを作成するためのSDKです。 特徴･使用方法 ： デバイスを定期的にアクセスしプロパティをSQLite DBに格納するミドルウェアが提供されます。アプリケーションからはGet/Set命令の実行をDBへのread/writeで実現できます。
7	KAIT-4S ~Canvas~ / Kadeckly (20150106) ◆ECHONET Lite機器のコントローラや連携アプリを開発するSDK SDK ID ： 00017 登録日/ Registered Date ： 2015. 1. 16. サイズ/ Size ： 47.5MB メーカー / Maker ： 神奈川工科大学（SonyCSL委託） 開発言語 / Programming Language ： Google Blockly ターゲットOS / Target OS ： Linux, Windows, Android, Mac OS, iOS ターゲットCPU / Target CPU ： N/A 概要 / Summary ： 本SDKは、ECHONET Liteの知識がなくても機器制御プログラムが簡単に作成できるVisual Programingツールです。機器連携のプロトタイピングに向いています。 特徴・使用方法 ： 本SDKをWeb serverにインストールし、Browserからアクセスして利用します。また、Kadecot をインストールした Android Tabletが必要です。詳細はマニュアルを参照ください。なお、本SDKをインストールしたサーバーがSony CSL様から公開されています。ただしバージョンは本サイトからダウンロードするものと同じとは限りません。 http://app.kadecot.net/Apps/Kadeckly
8	KAIT-4S ~EZ~ for Android 20150224 ◆ECHONET Lite機器のコントローラや連携アプリを開発するSDK SDK ID ： 00014 登録日/ Registered Date ： 2014. 11. 21. サイズ/ Size ： 6.5MB メーカー / Maker ： 神奈川工科大学（ユビキタス委託） 開発言語 / Programming Language ： Java ターゲットOS / Target OS ： Android ターゲットCPU / Target CPU ： Android機器

	概要 / Summary ： 本 SDK は、Android 機器上で動作するコンローラアプリケーションを作成するための SDK です。 特徴･使用方法 ： デバイスを定期的にアクセスしプロパティを SQLite DB に格納するミドルウェアが提供されます。アプリケーションからは Get/Set 命令の実行を DB への read/write で実現できます。
9	KAIT-4S ~HA~ ◆ECHONET Lite 機器のコントローラや連携アプリを開発する SDK SDK ID ： 00013 登録日/ Registered Date ： 2014. 11. 21. サイズ/ Size ： 3MB メーカー / Maker ： 神奈川工科大学（ユカイ工学委託） 開発言語 / Programming Language ： Objective-C, Swift ターゲット OS / Target OS ： iOS ターゲット CPU / Target CPU ： iOS device 概要 / Summary ： 本 SDK は、iOS Homekit framework を利用して、ECHONET Lite 機器の制御アプリケーションを作成するためのものです。 特徴・使用方法 ： 本 SDK を使って作成したアプリケーションを実行して ECHONET 機器を制御するには、「Homekit - ECHONET ブリッジ」が必要になります。本ブリッジの入手に関してはユカイ工学(http://www.ux-xu.com)へお問い合わせください。
１０	【Sample Program】Electric Lock Emulator ver.3.00 ◆電気鍵のエミュレータ（コントローラの動作が確認できる）SDK SDK ID ： 00010 登録日/ Registered Date ： 2014. 10. 30. サイズ/ Size ： 938.6KB メーカー / Maker ： Smart House Research Center 開発言語 / Programming Language ： OpenEHCO for Processing ターゲット OS / Target OS ： Windows ターゲット CPU / Target CPU ： INTEL x86 系 概要 / Summary ： OpenEHCO for Processing のサンプルプログラムです。ECHONET Lite 対応の電気錠として動作します。 特徴・使用方法 ： Processing version 2.1 controlP5 version 2.0.4
１１	Super Speed Node Generator for ECHONET Lite Ver.2.1.4 -Japanese Edition- ◆ECHONET Lite 機器の動作確認ができる SDK SDK ID ： 00008 登録日/ Registered Date ： 2014. 10. 30. サイズ/ Size ：

	124.2MB メーカー / Maker ： 神奈川工科大学大学院　博士前期課程　電気電子工学専攻　中島義人、横山悠平 開発言語 / Programming Language ： LabVIEW2013 ターゲットOS / Target OS ： Windows ターゲットCPU / Target CPU ： Pentium III/Celeron 866 MHz または同等のプロセッサ 概要 / Summary ： SSNGはLabVIEWで開発されたECHONET Liteパケットジェネレータです。(ECHONET Lite規格書 Ver1.10に対応、APPENDIX Releaseには無依存) 特徴・使用方法 ： Super Speed Node Generator for ECHONET Lite Ver.2.1.4では、ドメイン上のECHONET Lite機器をSearchボタンで探すことができ、発見された機器に任意のECHONET Liteフレームを送信することが可能です。また、テキストフィル形式で記述された任意のプロパティを用意することで、マウス操作だけでECHONET Liteフレームを生成、送受信することができます。
１２	Super Speed Node Generator for ECHONET Lite Ver.2.1.4 -English Edition- ◆ECHONET Lite機器の動作確認ができるSDK SDK ID ： 00007 登録日/Registered Date ： 2014.10.30. サイズ/Size ： 123.8MB メーカー / Maker ： NAKAJIMA Yoshito and YOKOYAMA Yuhei, Graduate School of Engineering, Kanagawa Institute of Technolog 開発言語 / Programming Language ： LabVIEW2013 ターゲットOS / Target OS ： Windows ターゲットCPU / Target CPU ： Pentium III/Celeron 866 MHz or better 概要 / Summary ： A ready-to-go tool that can generate any kind of ECHONET Lite packets through an easy-to-use GUI. 特徴・使用方法 ： ″Super Speed Node Generator for ECHONET Lite″ (SSNG, in abbrev.) is a tool that can act as a controller or a node with a simple or complex object(s). All kinds of ECHONET Lite packets can be generated and be sent through a simple operation on a GUI control screen. All received packets are shown in a window in delimited strings as ECHONET Lite frames. The standard search sequence specified in The ECHONET Lite Specification Version 1.1 is also supported.
１３	OpenECHO for Processing 2014.07.25版

	◆ECHONET Lite 機器を開発できる SDK SDK ID ： 00006 登録日/ Registered Date ： 2014. 10. 30. サイズ/ Size ： 11.2MB メーカー / Maker ： 神奈川工科大学 (SonyCSL 委託) 開発言語 / Programming Language ： Java (Processing) ターゲット OS / Target OS ： Linux, Windows, Mac OS ターゲット CPU / Target CPU ： INTEL x86 系 概要 / Summary ： OpenECHO for Processing は、Java による ECHONET Lite のオープン ソース実装である OpenECHO を Processing 用にライブラリ化したものです。 特徴・使用方法 ： 「OpenECHO for Processing 2014.05.02 版」の微修正版です。 詳細は Tutorial.pdf をご参照ください。OpenECHO についてのご質問は info@kadecot.net へお願いいたします。

上記の SDK は、簡単な登録でダウンロードすることができます。

この節では、ここに紹介されている SDK のうち、後に開発で実際に使用するものを紹介します。

3．1．1．SSNG（Super Speed Node Generator for ECHONET Lite）

SSNG は LabVIEW で開発された ECHONET Lite パケットジェネレータです。

ECHONET Lite ネットワークを行き交う電文パケットを読み取り、フィールドごとにデータを分解して表示してくれます。ネットワーク上の ECHONET Lite 機器を検索することができて、発見した機器に簡単な操作で電文を送ることができるので、機器開発を行う際に、機器が送り出す電文の正誤を確認することや、機器が電文に対して正しく動作するかどうかを検証することができます。

マイコンの開発環境と SSNG を使って、モデルスマート家電を開発することができます。

SSNG は、ECHONET Lite 規格書と同様、バージョンアップが行われているので、上の表にあるように、複数のバージョンが存在しますが、これから開発を行う方は、最新バージョンを使ってください。表の中で （ECHONET Lite 規格書 Ver1.10、APPENDIX D に対応） と説明があるのは、Appendix も Rerease が随時進められて、収容される機器の規格に追加や変更が生じるためです。

〔1〕SSNG の Install

ダウンロードしたファイルを適当な場所に全て展開します。

展開してできたフォルダにある setup.exe を実行すると、インストール先フォルダの指定ウインドウが開きます（図 3-1-1）。そのまま、次へを選択します。

（図 3-1-1：インストールフォルダの指定）

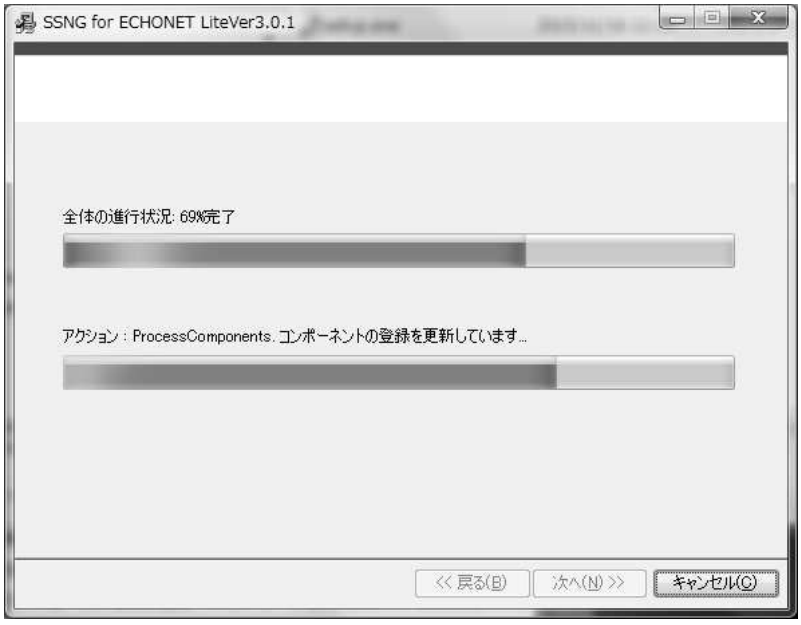
（図 3-1-2）ライセンスセンス同意のウインドウが開くので、同意を選択して、次へボタンをクリックします。

（図 3-1-2：ライセンスに同意）

インストーラ実行開始ウインドウで次へを選択します。

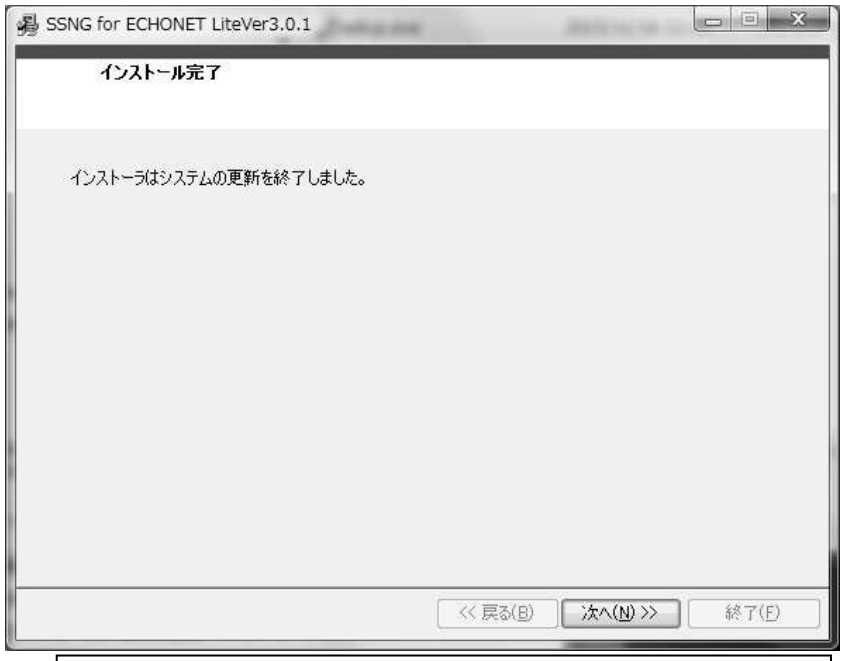
（図 3-1-3：インストーラ実行開始）

しばらくインストールが続きます。（図 3-1-4）



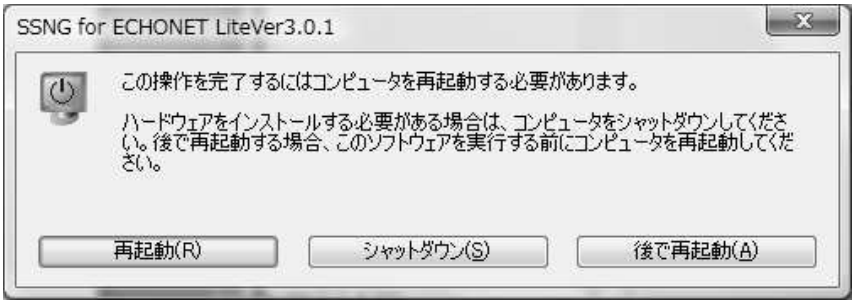
（図 3-1-4：インストール）

インストールが完了すると、次のようなウインドウが開きます（図 3-1-5）。



（図 3-1-5：インストール完了）

次へボタンをクリックして再起動ウインドウの指示に従います（図 3-1-6）。



（図 3-1-6：PC 再起動）

再起動後、全てのプログラムの中に、SSNG のグループができていて（図 3-1-7）これをクリックして起動します。

（図 3-1-7：SSNG プログラムグループ）

SSNG を起動すると、（図 3-1-8）のようなウインドウが開きます。

（図 3-1-7：SSNG のウインドウ）

これで、SSNG が使えるようになりました。

3．1．2．OPENECHO for Processing

OpenECHO for Processing は、株式会社ソニーコンピュータサイエンス研究所(Sony CSL)では、ECHONET Lite を Java で実装したクラスライブラリを開発し、オープンソース配布しているものです。これには、丁寧なチュートリアルが付属しています。このチュートリアルを読むことで、以下のことができるようになります。

①. ECHONET Lite 対応の市販の家電を操作する

②. ECHONET Lite 非対応家電や自作機器を、赤外線などを用いて ECHONET Lite に対応させ、ネットワークから操作する

③. 市販のサービスと自作のサービスを融合させ、協調制御する

なお、本チュートリアルを理解するためには、基本的な Java のプログラミング、および Processing に関する知識が必要です。ECHONET Lite についての前提知識は不要です。

OpenECHO for Processing を利用するには、ライブラリのインストールが必要ですが、手順は Processing のインストールの後に説明します。

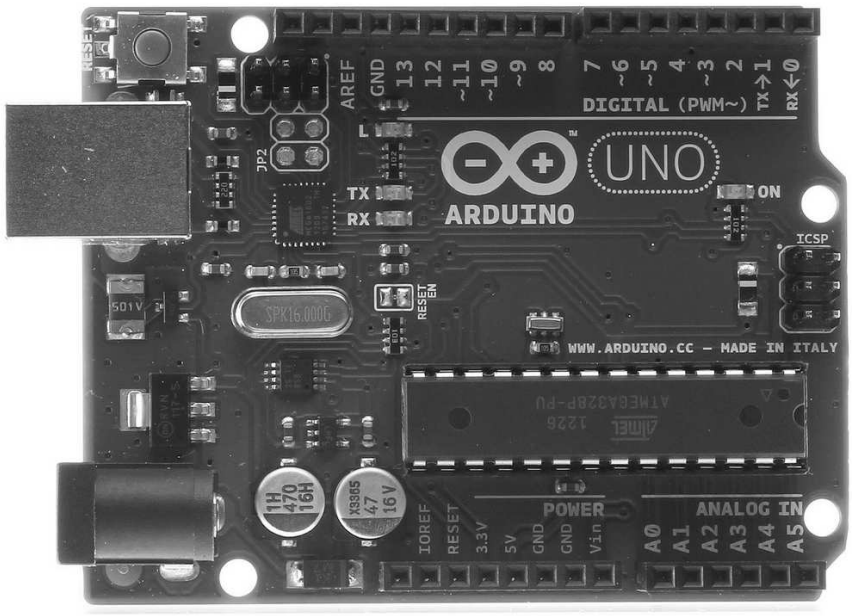
３章２節　マイコン開発環境

モデルスマート家電を開発するためには、組込系のマイコン開発環境が必要になります。現在利用できる組込系マイコン開発環境は様々なものがありますが、ECHONET Lite ネットワークに接続することを考えると、Ethernet インターフェースに対応できるものが必要です。みなさんが、後で自力開発を試みる際に必要な情報が容易に得られることと、多くの実績があるものとして、導入が容易な Arduino を利用することにします。

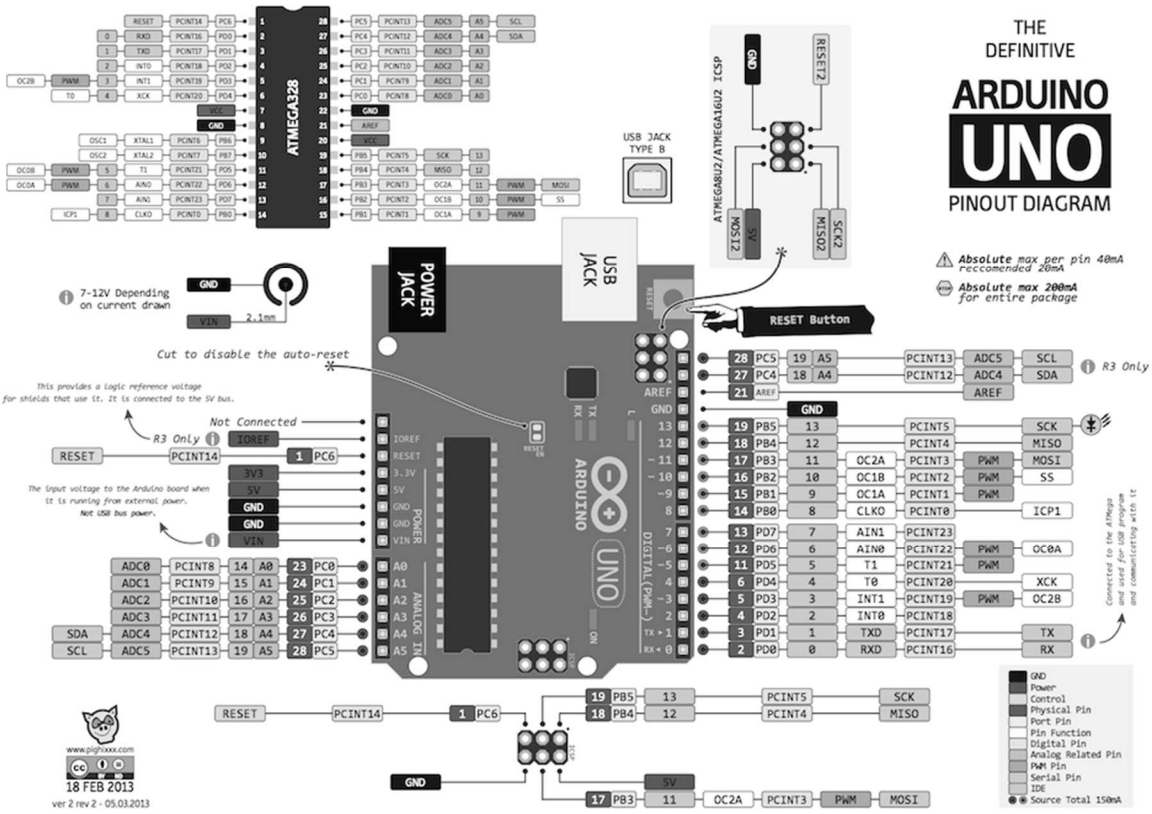
３．２．１．Arduino ボード

Arduino は、小型のマイコンボードです。Arduino はいろいろな種類がありますが、ここで利用するのは、ArduinoUNO というハードウエアです。小さな基板の上に ATmga328P という 28 本の足（PIN）を持つマイコンが載っています（図 3-2-1）。

プログラム用 Furash メモリーが 32KB、SRAM が 2KB、EEPROM が 1 KB 搭載されていてシステムクロック 16MHz で動作します。



（図 3-2-1：Arduino UNO ボード）



（図 3-2-2：PIN 配置）

Arduino ボードには、たくさんの PIN ソケットがあって、それぞれの PIN は、役割が決まっています(図 3-2-2)。

Pin0~Pin13：14 本のデジタル IO ピンです。入力または出力として使えます。プログラム中で、どちらにして使うかを指定します。

Pin0~Pin5：6 本のアナログ IN 用ピンです。センサなどの電圧計測などに利用します。読み込まれた値は、0~1023 の数値に変換されます。

Pin3,5,6,9,10,11：6 本のアナログ OUT 用ピンです。PWM 制御ができるので、サーボモータを接続することができます。

電源は、USB ポート経由で USB 充電器や PC などから供給するほか、AC アダプタ用のコネクタも付いています。

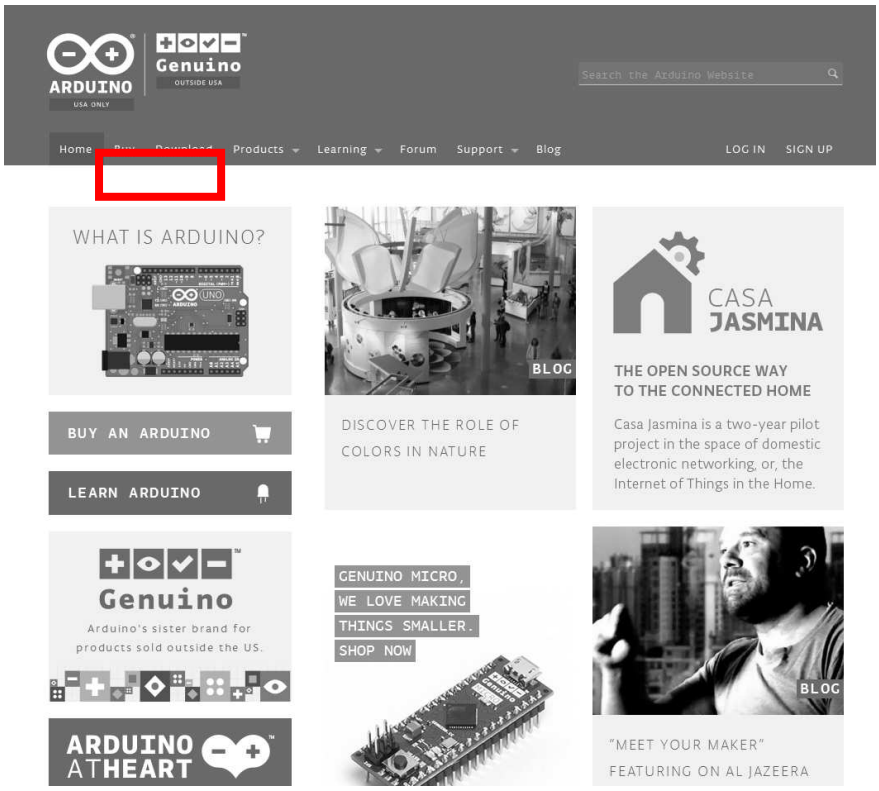
3．2．2．Arduino IDE

Arduino は、作成したプログラムをコンパイル・リンクして、CPU のフラッシュメモリに書き込んで、初めて動作します。そのプログラムの作成・コンパイル・リンク・書き込みは、全て専用 IDE を使って行います。

プログラムは C++で作成して、IDE で管理します。Arduino では、ソースファイルを【スケッチ】と呼んでいます。スケッチは、目的に応じた雛型が数多く準備されているので、初めての機能でも雛型を基にして、容易に開発ができる環境が整います。

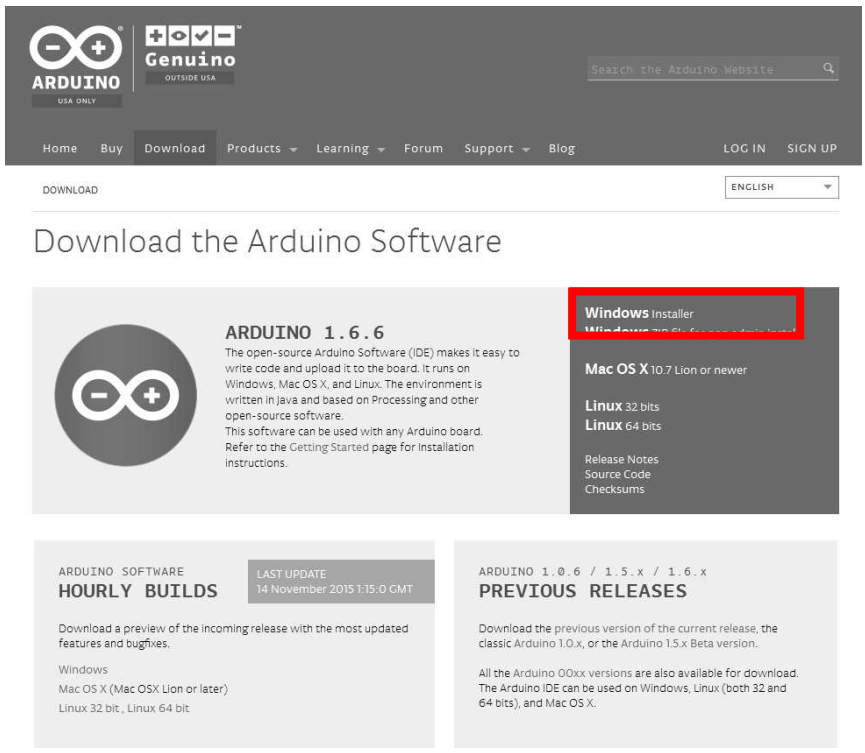
〔1〕Arduino IDE の入手

WEB で「Arduino」と検索すると、（図 3-2-3）のような Arduino のサイトが見つかります。



（図 3-2-3：Arduino WEB サイト）

ページ上部位にある Download のタブを選択すると、（図 3-2-4）の IDE ダウンロードページが開きます。



（図 3-2-4：IDE Download ページ）

ここで。右側にある Windows Installer をクリックします。次に開くページで【JustDownload】を選ぶと、IDE ファイルの Installer がダウンロードできます。

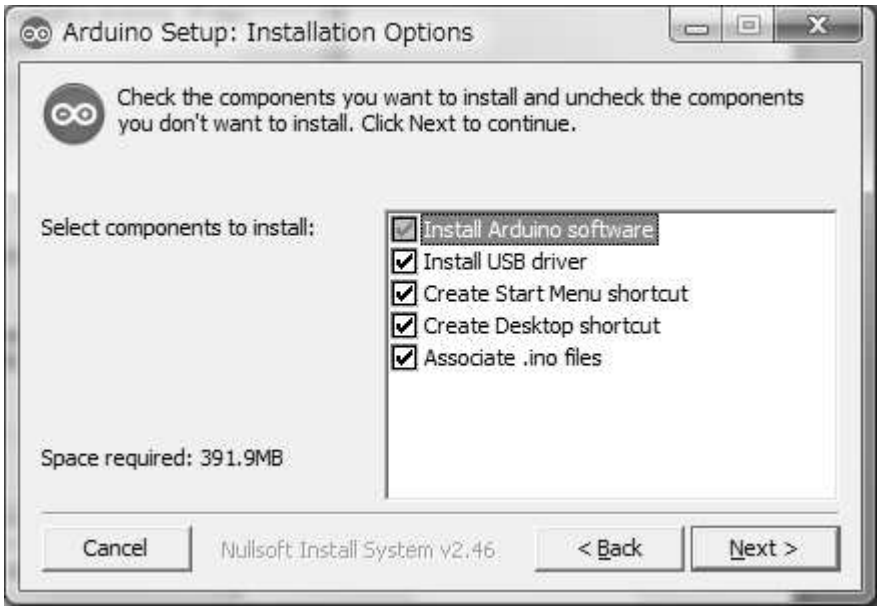
〔2〕Arduino IDE の Install

入手した Installer のファイルを適当な場所において（図 3-2-5）、管理者権限で実行すると、Install が始まります。

（図 3-2-5：IDE Installer）

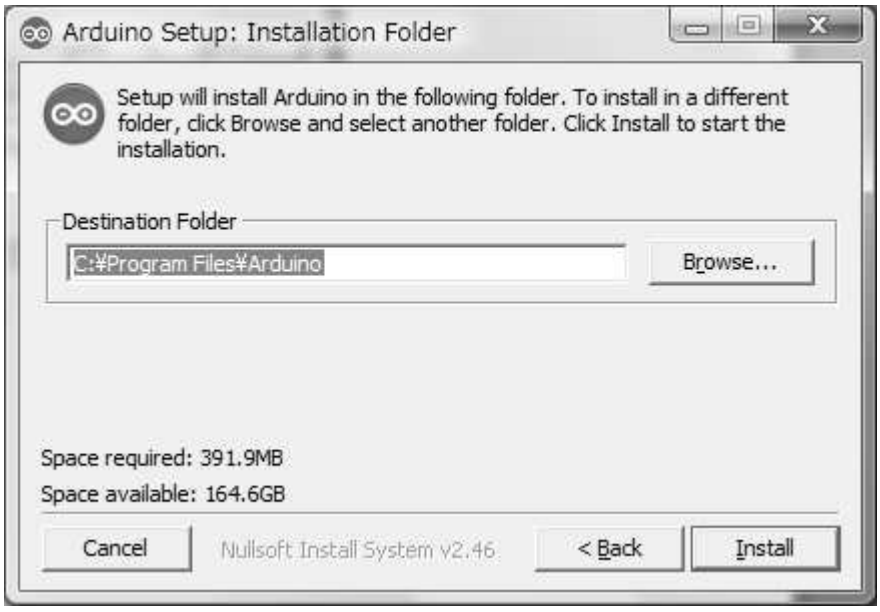
（図 3-2-6：License Agreement）

ここで、【I Agree】ボタンをクリックします。



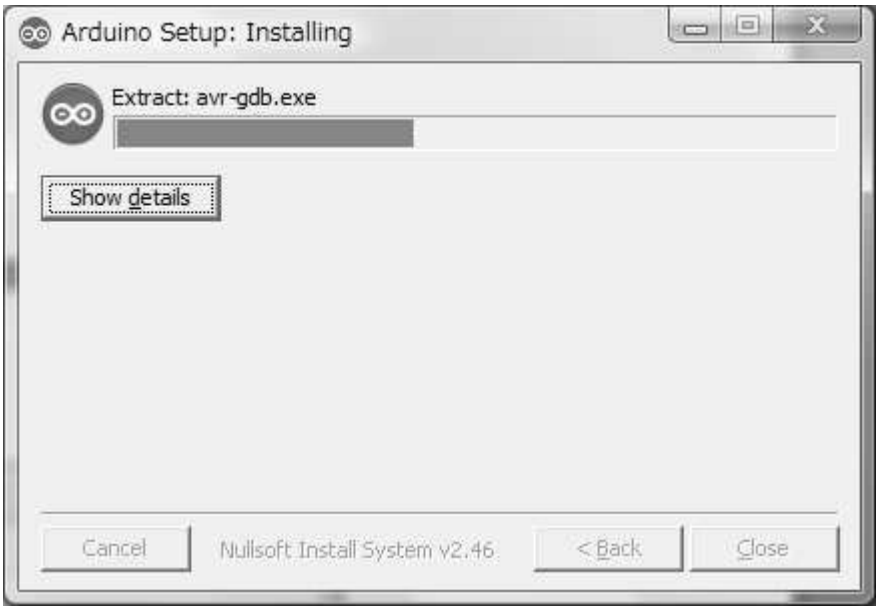
（図 3-2-7：Instartion Options）

次の Instation Options 画面（図 3-2-7）ではデフォルト状態（すべてチェックされた状態）で Next ボタンをクリックします。



（図 3-2-8：Instartion Folder）

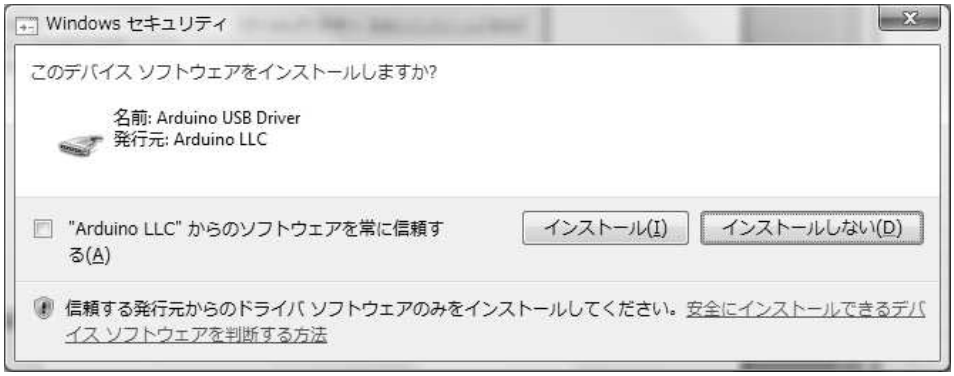
次の Instration Folder 画面（図 3-2-8）もそのまで、Install ボタンをクリックします。



（図 3-2-9：インストール中）

次から、USB ドライバのインストール画面が続きます。

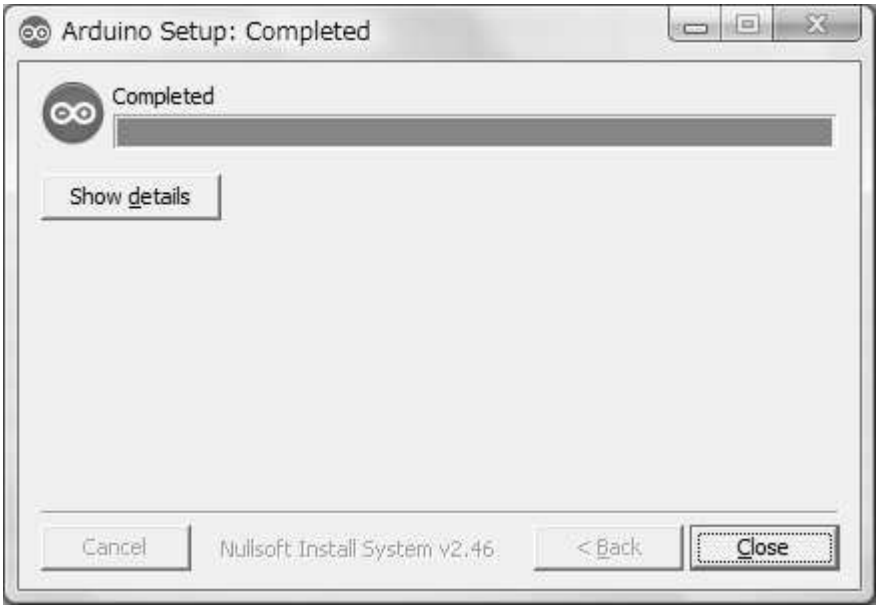
（図 3-2-10：USB ドライバの Install その 1）



（図 3-2-11：USB ドライバの Install その２）

（図 3-2-12：USB ドライバの Install その３）

（図 3-2-13：USB ドライバの Install その４）



（図 3-2-14：Install Complete）

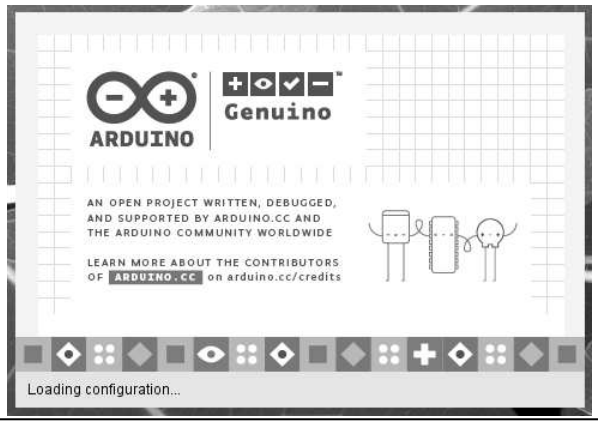
Complete が表示されれば、IDE は正しくインストールされています。
Close ボタンをクリックしてください。

〔3〕Arduino IDE の起動

デスクトップに（図 3-2-15）のようなショートカットが追加されています。

（図 3-2-15：Arduino ショートカット）

これをダブルクリックすると、Arduino IDE が起動します。



（図 3-2-16：Arduino IDE 起動画面）

（図 3-2-16）のような起動画面が表示された後、少し時間をおいて（図 3-2-17）のような IDE ウインドウが開きます。

（図 3-2-17：Arduino IDE ウインドウ）

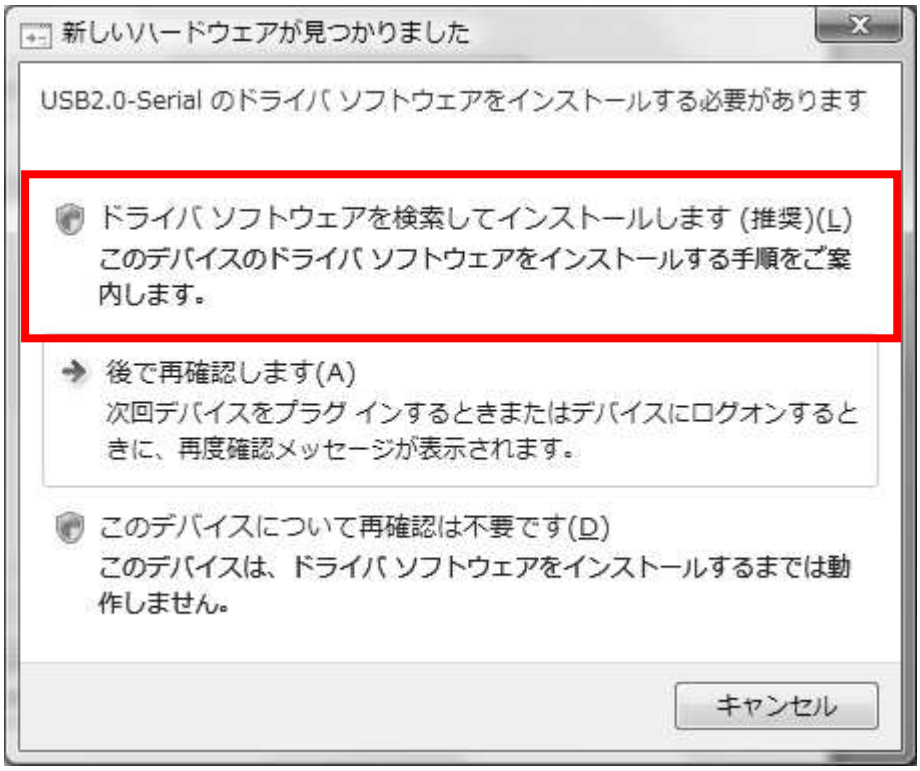
これで、Arduino IDE は利用できるようになりました。

〔4〕USB-シリアルドライバのインストール

Arduino ボードは、USB で PC と接続してプログラムの書き込みを行います。このとき使用する、USB-シリアルドライバをインストールします。

まず、USB ケーブルで Arduino ボードと PC を接続すると、「新しいハードうウェアが見つかりました」という（図 3-2-18）のウインドウが現れますので、「ドライバソフトウエアを検索してインストールします（推奨）」を選択して実行します。

※重要）Arduino は、ハードウエアの仕様も公開されていて、サードパーティ製の製品も数多く出回っています。そのような製品の場合、USB-シリアルドライバがデフォルトのシステムでは、インストールできない場合も多く見受けられます。そのような時は、Arduino ボードの USB コネクタ付近に配置されている IC の型番で Web 検索をして、ドライバを探すことができます。



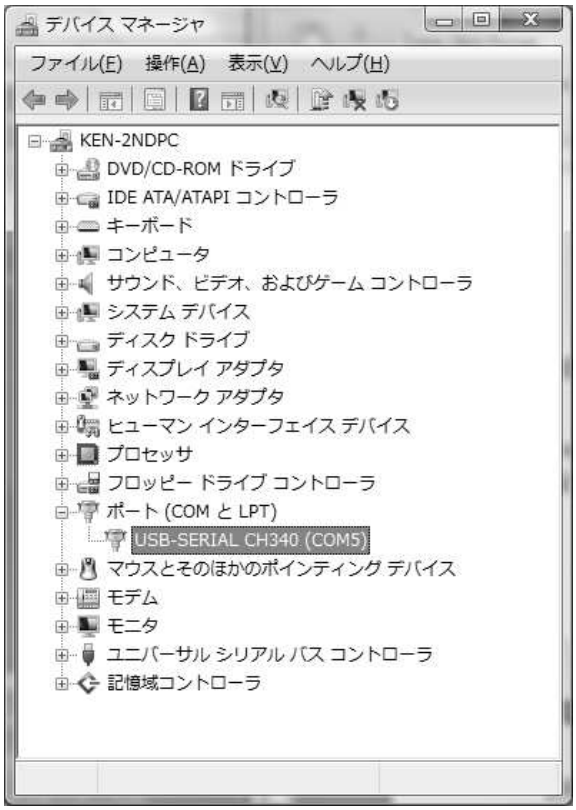
（図 3-2-18：新しいハードウエアが見つかりました）

（図 3-2-19：USB-シリアルドライバのインストール）

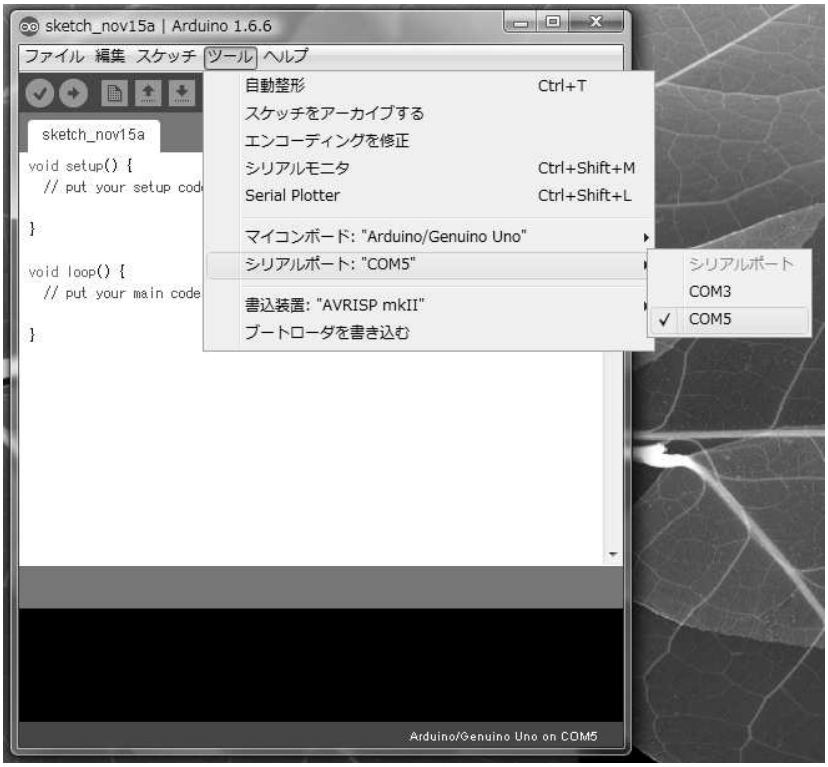
これで、USB－シリアルドライバがインストールされます。

〔5〕COM ポートの確認と設定

ドライバインストールが完了したら、利用できる COM ポート番号をデバイスマネージャの画面で確認しておきます。。



（図 3-2-20：COM ポート番号の確認）



（図 3-2-21：COM ポート番号の設定）

確認した COM ポート番号は、Arduino IDE のツール→シリアルポート→COM x で設定しておきます。この設定は、IDE を終了しても、残っていますので、次に IDE 起動するきには、すでに COM ポートが利用できる状態になっています。

※注意）複数の Arduino ボードを利用する場合は、ボードごとにインストールされる COM ポート番号が変わります。また、同じボードも違う PC に接続すると COM ポート番号が変わることがありますので、注意が必要です。

３章３節　Processing

コントローラ等を PC で開発したり、必要なツールや環境を開発したりすることも重要なことです。この節では、PC ベースで ECHONET Lite ネットワークに接続できるソフトウエアを開発できる環境を紹介します。

なぜ、Processing を選択したかと言うと【3.1.2.】OPENECHO for Processing で解説する ECHONET Lite 開発用 SDK があり、ライブラリも充実しているからです。

３．３．１．Processing の特徴

Processing は、PC で稼働します。オープンソースで開発されていて、だれでも無料で利用できます。学生、アーティスト、デザイナーなどプラグラミングが不慣れな人でも扱いやすい（やさしくプログラミングできる）環境を提供します。

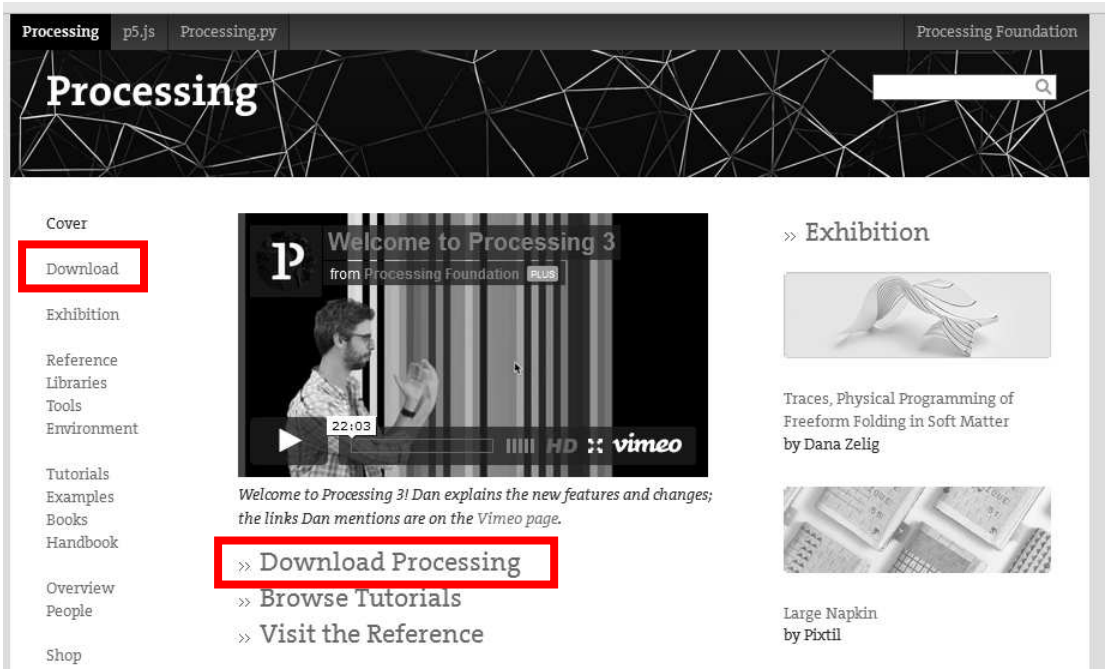
視覚的な表現（イメージの生成、アニメーション、インタラクション）が他のプログラミング言語に比べてやりやすいので、自由な発想のシステムが開発できます。

Arduino などの他の開発環境に大きな影響を与えています。後に分かりますが、Processing の IDE は Arduino とほぼ同じレイアウトで同じ操作感があります。JavaScript に移植された Processing.js を用いれば、デスクトップ PC・タブレット・スマートフォンといった端末の種類を問わずにブラウザで動作させることができるので、プロトタイピングに利用しやすいシステムです。

３．３．２．Processing の環境準備

〔1〕Processing の入手

Web で「Processing」と検索すると、（図 3-3-1）のような Processing のサイトが見つかります。



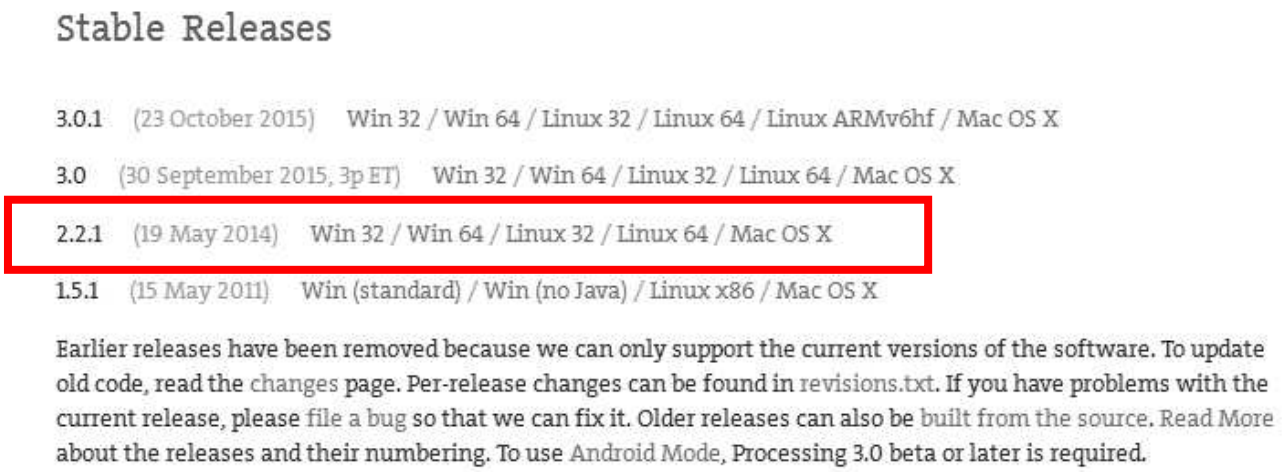
（図 3-3-1：Processing の WEB サイト）

ページ左上の Download か中央の Download Processing のリンクをたどり、続けて開くページで Donation（寄付）を適宜選択して Download をクリックす

ると、次のようなページが開きます（図 3-3-2）。このページで、使用している OS に該当するシステムをダウンロードしましょう。※NoDonation を選択してもダウンロードできます。

※重要）本書を作成している時点（2015 年 11 月 10 日）の最新バージョンでは、後に利用する OpenECHO for processing でエラーが発生しますので、ここでは、この画面の下部（図 3-3-3）の Ver.2.2.1 をダウンロードしてください。

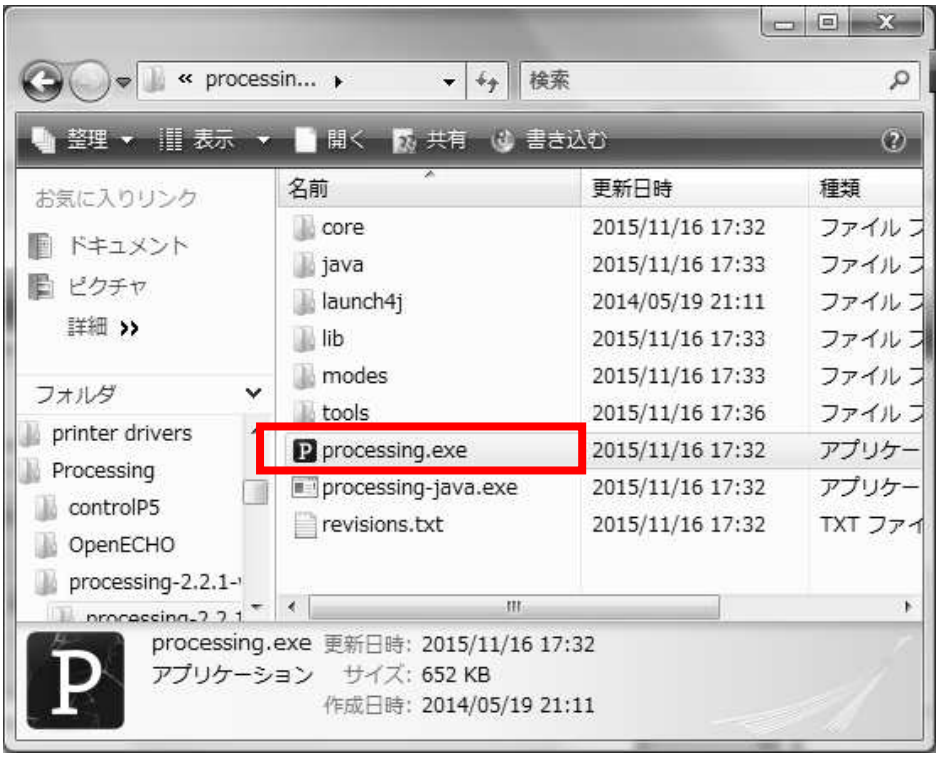
（図 3-3-2：Processing の Download ページ）



（図 3-3-3：Processing の旧バージョン）

〔2〕Processing の Install

入手した Installer のファイルを適当な場所において、展開すれば、Install は完了です。展開してできたフォルダの中の processing.exe（図 3-3-4）を実行すれば、Processing が起動します。



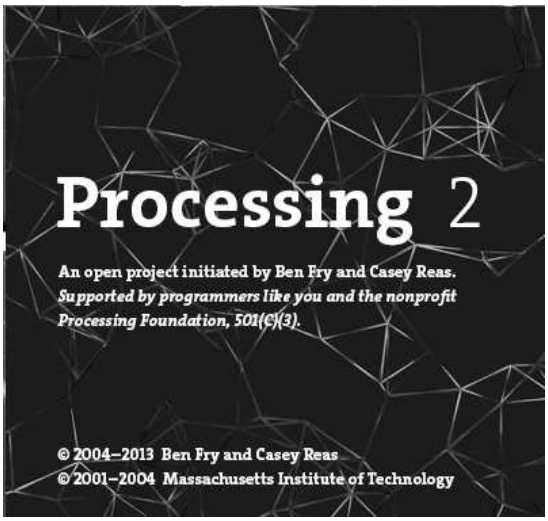
（図 3-3-4：Processing）

（図 3-3-5：Processing のショートカット）

（図 3-3-5）のようにデスクトップにショートカットを作っておくとよいでしょう。なお、展開した Processing のフォルダのパスは、後にライブラリを追加したりする場合、必要となりますので、メモをしておくとよいでしょう。

〔3〕Processing の起動

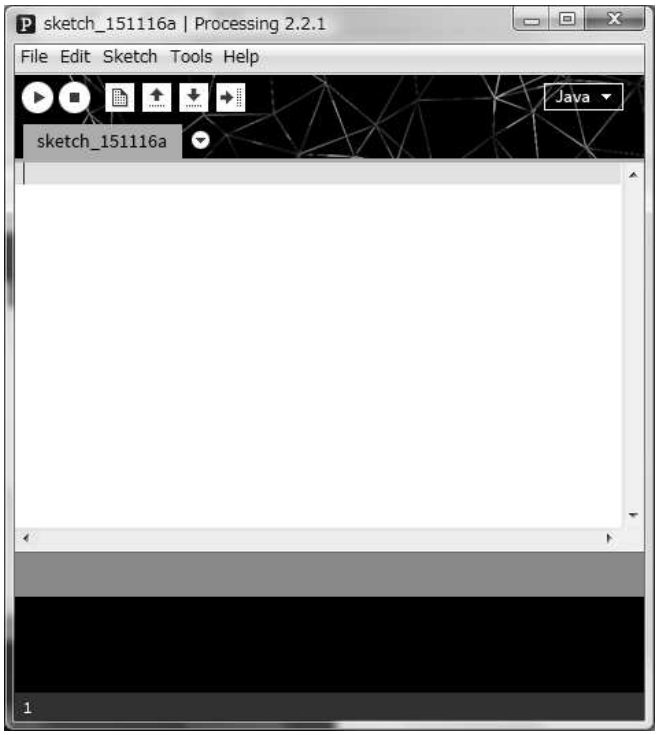
先ほど作ったショートカットをダブルクリックすると、Processing が起動します。



（図 3-3-6：Processing 起動画面）

（図 3-3-6）のような起動画面が表示された後、少し時間をおいて（図 3-3-7）のウインドウが開きます。

この（図 3-3-7）ウインドウが ProcessingIDE です。



（図 3-3-7：Processing IDE）

これで Processing も利用できるようになりました。

Processing は、PC の USB ポートを介したシリアル通信や LAN ポートで Ethernet 通信をおこなえます。これらについて特別なドライバソフトウエアは必要ありません。

〔4〕OpenECHO for Processing のインストール

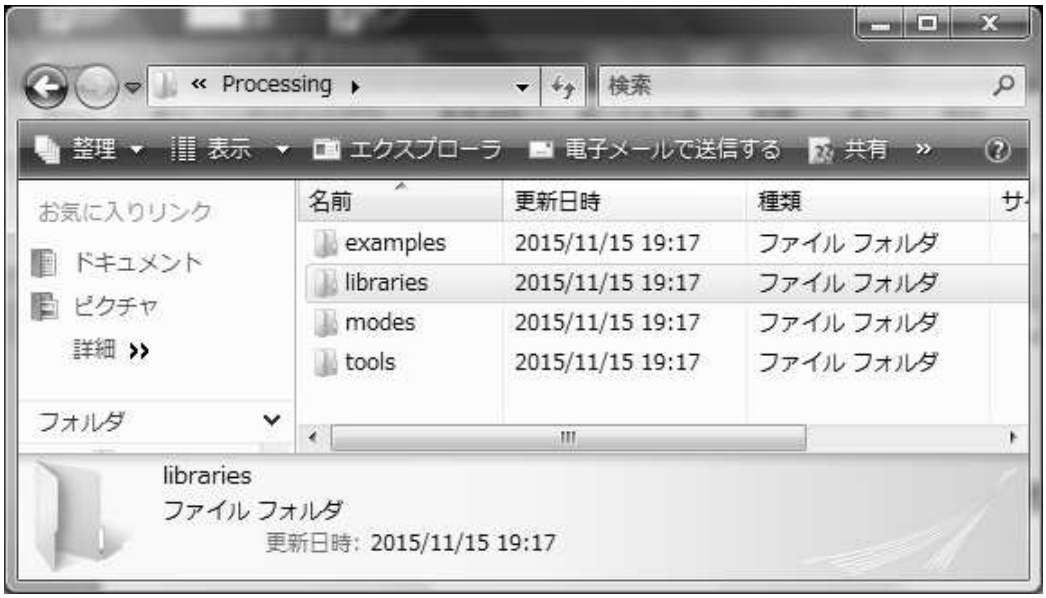
OpenECHO for Processing は Processing の後にインストールしますので、ここで手順を説明します。

3 章で紹介した SDK のダウンロードを行うと、圧縮されたファイルが入手できます。これを適当な場所に置いて解凍すると、（図 3-3-8）のようなファイルが現れます。

（図 3-3-8：OpenECHO for Processing のライブラリ）

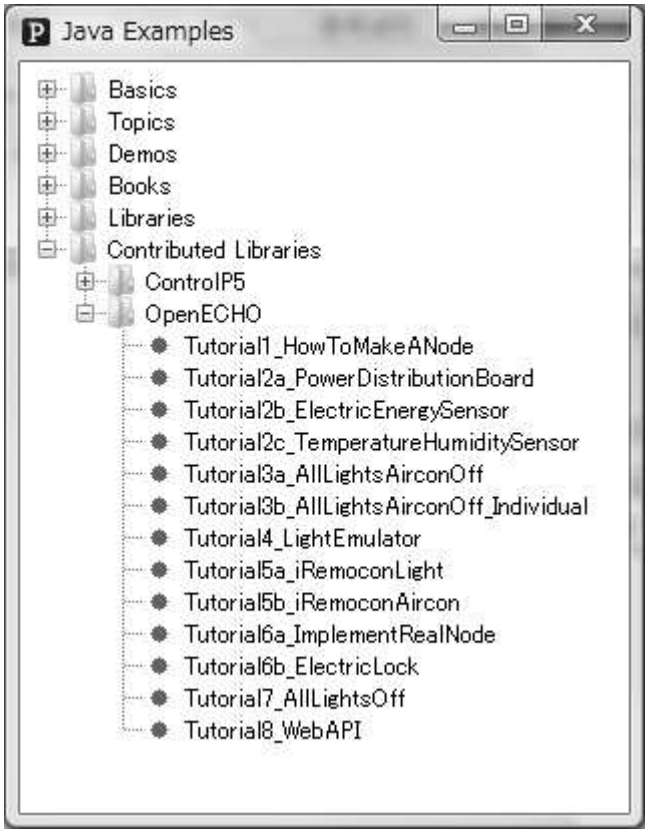
ここに含まれている librareis フォルダの中にある（図 3-3-9）のサブフォルダの内容を Processing のスケッチを保存するフォルダ（図 3-3-10）の libraries フォルダ配下にそのままコピーします。

（図 3-3-9：OpenECHO for Processing のライブラリ）



（図 3-3-10：Processing のライブラリフォルダにコピーする）

次に、Processing を起動して、メニューから、File → Example とたどり、次のようなウインドウ（図 3-3-11）が現れれば、OpenECHO forProcessing のライブラリとチュートリアルはインストールされています。



（図 3-3-11：Processing のサンプルウインドウ）

この章のまとめ

この章で解説した SDK や他の開発環境などは、ECHONET Lite だけではなく、いろいろな開発に応用できる内容です。後の章ではモデルスマート家電の開発を行いますが、是非そのときに、実際の SDK などに触れて、実感を得てください。

４章　ECHONET Lite 通信電文解析

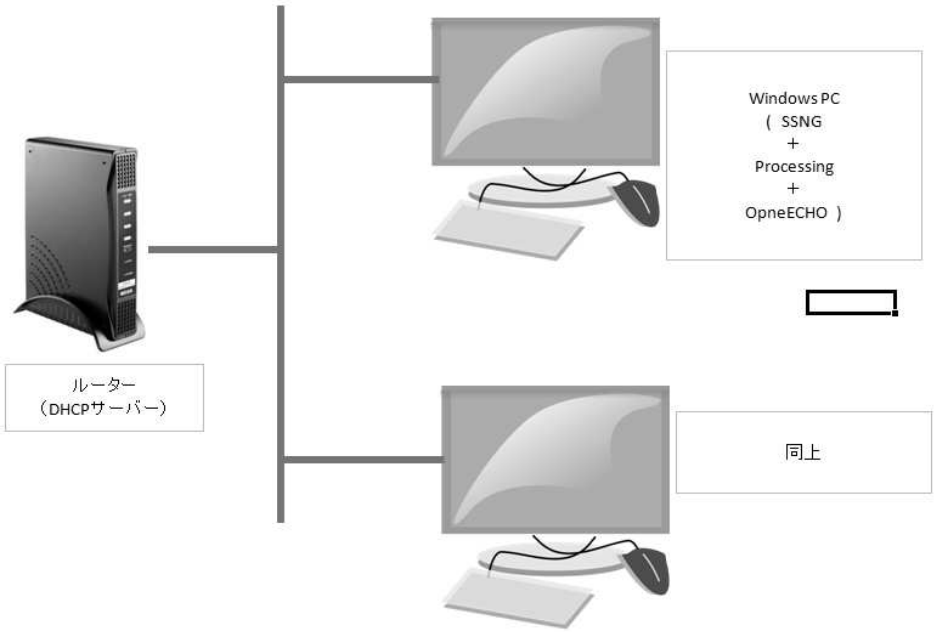
４章では、ECHONET Lite ネットワークを行き交う通信電文を実際に取得する方法を紹介して、その中に含まれるデータを解析してみましょう。電文解析には ECHONET Lite 機器が必要ですが、実施の機器が準備できるとは限らないので、ここでは先に紹介した ECHONET Lite SDK を利用して、機器をエミュレーションする方法を採用します。

４章１節　PC の準備

この章で行う実験には、次のような準備が必要です。（図 4-4-1）

①. 2 台以上の LAN 接続された PC。　※PC に設定されている IP アドレスをあらかじめ調べておいてください。

②. 2 台の PC には、3 章で説明した SSNG（Super Speed Node Generator for ECHONET Lite）と Processing をインストールした後、OpenECHO for Processing のライブラリとチュートリアルをインストールしておいてください。※ 2 台とも同じ環境の PC を利用します。こうすることで、学校で実験を行う際には、2 人 1 組であるいは、複数人のグループで、相互接続した環境での実験が可能です。



（図 4-1-1：想定環境）

４章２節　電文の発行と取得

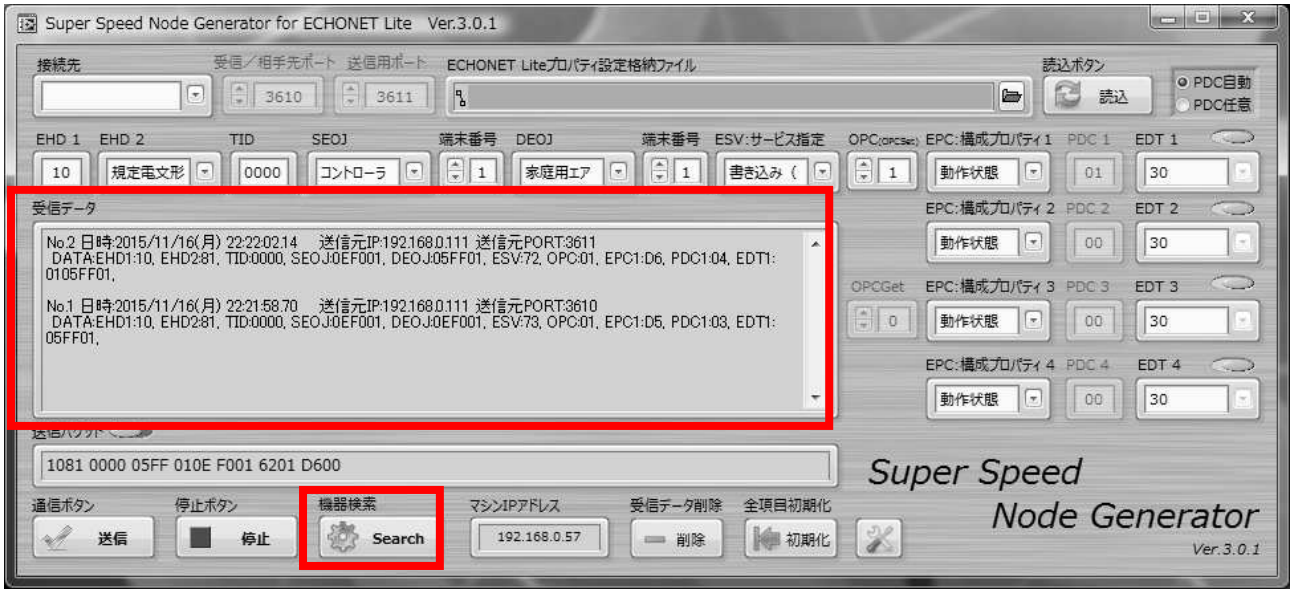
環境が整いましたら、実際に電文の発行と取得を行ってみましょう。

４．２．１．SSNG の相互通信テスト

まず両方の PC で、Super Speed Node Generator を起動します。（図 4-2-1）

（図 4-2-1：SSNG 起動）

次に、**両方の PC** で、ウインドウ左下にある「**通信開始**」ボタンを押します。この操作で SSNG は ECHONET Lite ノードとして動作し始めます。次に、**片方の PC** で「**Search**」ボタンを押すと、他方の SSNG から返信を受け取り、受信データに解析されたデータが表示されます。（図 4-2-2）



（図 4-2-2：受信データの解析）

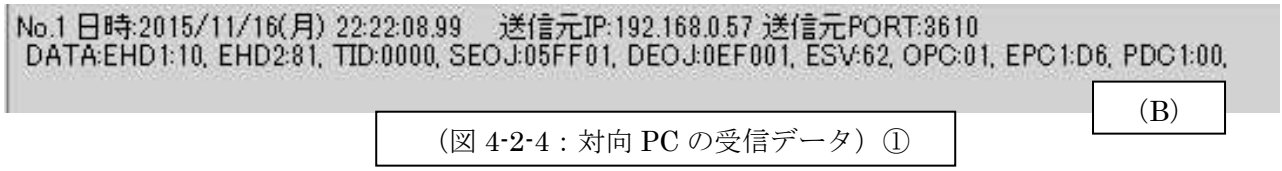
「停止」ボタンを押せば、SSNG は停止します。再開する際は、もう一度「通信開始」ボタンを押してください。

４．２．２．解析　その１

このとき受信したデータを拡大表示したものが（図 4-2-3）です。

（図 4-2-3：受信データ）②

受信データは、受信した順に NO がついていて、新しいデータが上部に表示されています。このとき、相手方 PC の SSNG で受信されたデータを示します。(図 4-2-4)



（図 4-2-4：対向 PC の受信データ）①

この例では、後から SSNG を開始した PC から②（図 4-2-3）の No.1（B）が発行されました。その後①（図 4-2-4）の No.1 の電文（A）が発行されて、その応答として ②（図 4-2-3）の No.2（C）の電文が送信されたことになります。

（A）の電文は、送信元 IP：192.168.111 から、Port：3610 で発信され
その内容は、次の通りです。

①．EHD1：10（ほぼ固定）
　ECHONET Lite ヘッダ１
②．EHD2：81（ほぼ固定）
　ECHONET Lite ヘッダ２
③．TID：0000（送信元の連続番号と思えばよい）
　Transaction ID
④．SEOJ：0EF001（プロファイルクラス＋ノードプロファイル＋01 番）
　送信元 ECHONET Lite オブジェクト指定

⑤．DEOJ：0EF001（プロファイルクラス＋ノードプロファイル＋01 番）
相手先 ECHONET Lite オブジェクト指定
⑥．ESV：73（プロパティ値通知＝INF）
ECHONET Lite サービス
⑦．OPC：01（以後のプロパティ処理数は１つ）
処理プロパティ数
⑧．EPC：D5（インスタンスリスト通知）
ECHONET Lite プロパティ
⑨．PDC：00（以後のデータは無し）
EDT バイト数

となっています。

この電文は、コントローラが立ち上がった際に、自分もノードの一つなので、ネットワーク内のコントローラにノードが立ち上がったことを通知している電文です。

これらの定義は、【2.3.5.】で説明した、ECHONET Lite 規格書 第２部 第３章 電文構成に規定があります。参照して、各フィールドの値を確認してください。

（B）の電文は、送信元 IP：192.168.057 から、Port：3610 で発信され
その内容は、次の通りです。

①．EHD1：10（ほぼ固定）
ECHONET Lite ヘッダ１
②．EHD2：81（ほぼ固定）
ECHONET Lite ヘッダ２
③．TID：0000（送信元の連続番号と思えばよい）
Transaction ID
④．SEOJ：05FF01（管理・操作機器クラスグループ＋コントローラ＋01 番）
送信元 ECHONET Lite オブジェクト指定
⑤．DEOJ：0EF001
（プロファイルクラスグループ＋ノードプロファイル＋01 番）
相手先 ECHONET Lite オブジェクト指定
⑥．ESV：62（プロパティ値読み出し要求＝Get）
ECHONET Lite サービス
⑦．OPC：01（以後のプロパティ処理数は１つ）
処理プロパティ数
⑧．EPC：D6（自ノードインスタンスリスト S）
ECHONET Lite プロパティ

⑨．PDC：00（以後のデータは無し）
EDT バイト数

となっています。

（C）の電文は、送信元 IP：192.168.111 から、Port：3611 で発信され
その内容は、次の通りです。

①．EHD1：10（ほぼ固定）
ECHONET Lite ヘッダ１
②．EHD2：81（ほぼ固定）
ECHONET Lite ヘッダ２
③．TID：0000（受信した電文と同じフィールドの値をセット）
Transaction ID
④．SEOJ：0EF001
（プロファイルクラスグループ＋ノードプロファイル＋01 番）
送信元 ECHONET Lite オブジェクト指定
⑤．DEOJ：05FF01（管理・操作機器クラスグループ＋コントローラ＋01 番）
相手先 ECHONET Lite オブジェクト指定
⑥．ESV：72（プロパティ値読み出し応答＝Get_Res）
ECHONET Lite サービス
⑦．OPC：01（以後の処理プロパティ数）
処理プロパティ数
⑧．EPC：D6（自ノードインスタンスリスト S）
ECHONET Lite プロパティ
⑨．PDC：04（以後のバイト数）
EDT バイト数
⑩．EDT1：0105FF01
（インスタンス総数＝１＋管理・操作関連機器クラス＋コントローラ＋01 番）
プロパティ値データ

となっています。

このやりとりは、電文（A）は、ノードが立ち上がったことを報告した後に、機器検索（B）からの問い合わせがあり、その電文に対して、対向 PC が（C）の電文で応答を返したことになります。

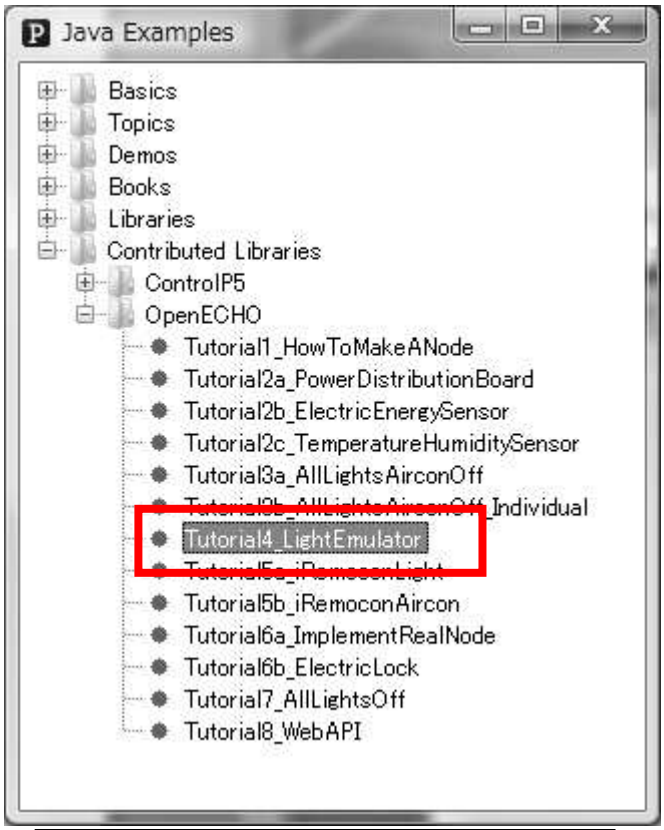
４章３節　一般照明機器コントロールの通信テスト

本物の ECHONET Lite 対応の一般照明器具があれば話は早いのですが、ここでは、SDK を使って一般照明エミュレータを作ります。そして SSNG を一方の PC で起動しておき、他方の PC で一般照明エミュレータを起動します。一般照明機器の代わりです。その後、エミュレータ上で一般照明の点灯・消灯を行ったときの電文を取得して、解析します。

４．３．１．一般照明器具エミュレータの作成

これには、OpenECHO for Processing を使います。作成と言っても、すでに提供されている Example をそのまま使うので、簡単にできます。

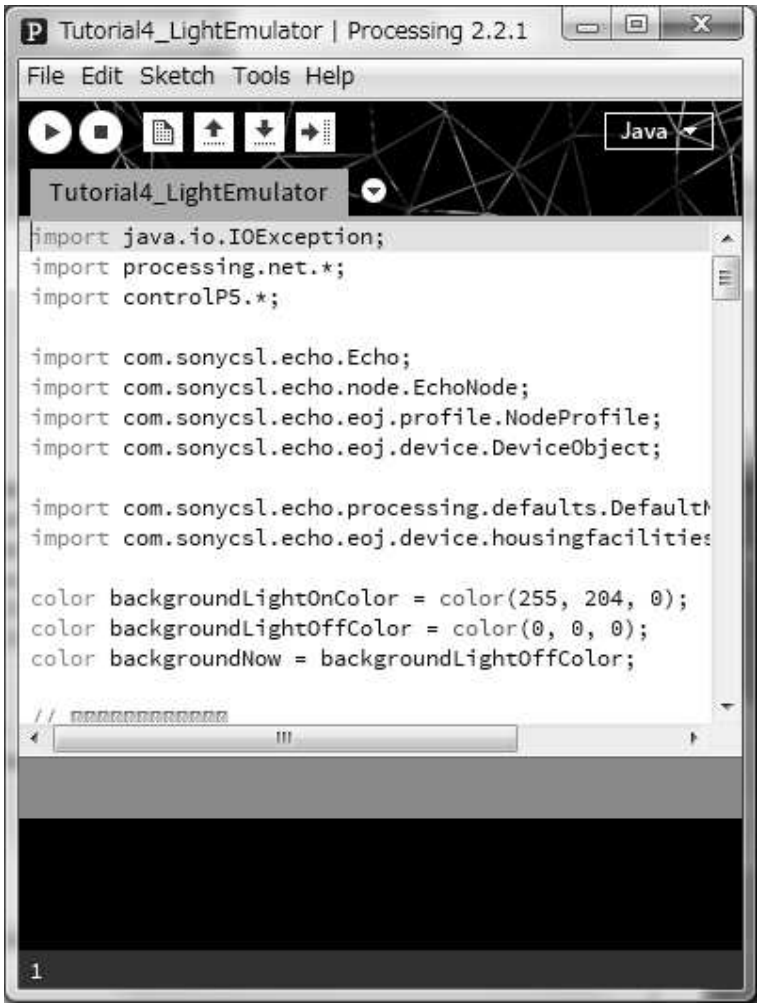
まず、Processing を起動します。メニューで File → Examples とたどると次のウインドウが表示されます。（図 4-3-1）



（図 4-3-1：Exsample）

このウインドウで Tutorial4_LightEmulator をダブルクリックします。

Processing のウインドウに、一般機器エミュレータのスケッチ（Processing も Arduino と同様にプログラムのことをスケッチと呼びます。）が開きます。（図 4-3-2）



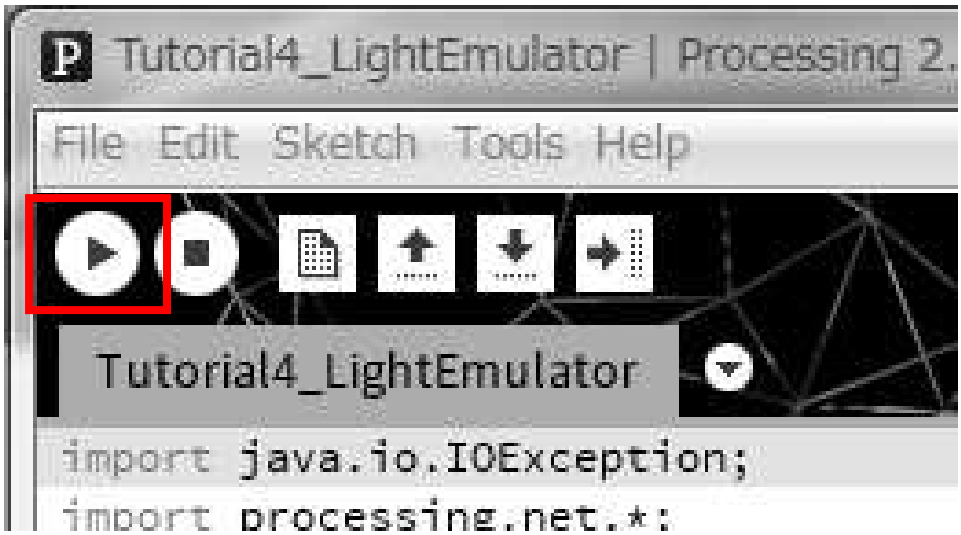
（図 4-3-2：一般照明機器のスケッチサンプル）

これで終わりです。もう準備ができました。簡単でしょう！！

４．３．２．一般照明器具エミュレータの電文取得

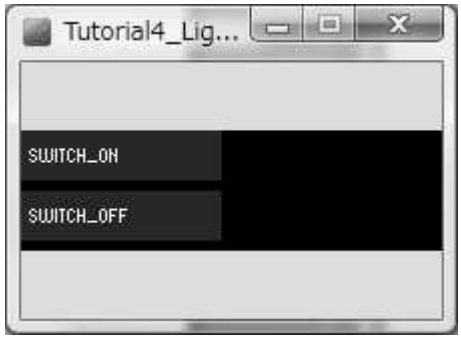
一般照明機器の立ち上がりからの電文を取得したいので、先に、他方の PC で SSNG を起動して、通信を開始しておきます。

次に、Processing のウインドウ上部にある右向きの三角印のボタン（図 4-3-3）をクリックします。これがスケッチの実行ボタンです。



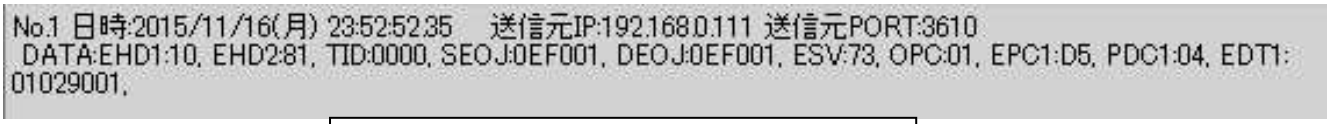
（図 4-3-3：スケッチの実行）

スケッチを実行すると、次のウインドウが現れます。（図 4-3-4）



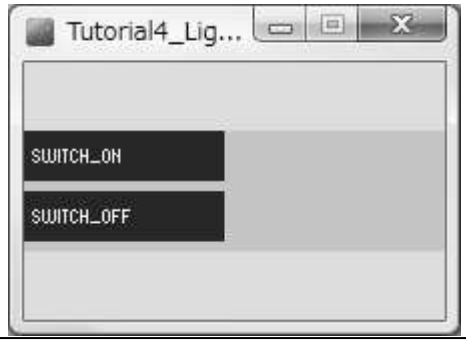
（図 4-3-4：一般照明エミュレータ）

このとき、すでに SSNG は電文を取得しています。これは、一般照明が起動したときの電文です。（図 4-3-5）



（図 4-3-5：一般照明起動時の電文）

次に、エミュレータウインドウの SWITCH_ON ボタンをクリックします。すると、ウインドウ内部が黄色に変化して、明りが点いたような雰囲気になります。（図 4-3-6）



（図 4-3-6：一般照明点灯状態）

これで、SSNG は次の電文が取得できています（図 4-3-7）。Processing ウインドウ上部、先ほどのボタンの右隣の四角いマークが入ったボタンをクリックして、スケッチの実行を停止してください。

```
No.3 日時:2015/11/17(火) 00:04:34.75　送信元IP:192.168.0.111 送信元PORT:3610
 DATA:EHD1:10, EHD2:81, TID:0004, SEOJ:029001, DEOJ:0EF001, ESV:73, OPC:01, EPC1:80, PDC1:01, EDT1:31,

No.2 日時:2015/11/17(火) 00:04:32.45　送信元IP:192.168.0.111 送信元PORT:3610
 DATA:EHD1:10, EHD2:81, TID:0002, SEOJ:029001, DEOJ:0EF001, ESV:73, OPC:01, EPC1:80, PDC1:01, EDT1:30,

No.1 日時:2015/11/16(月) 23:52:52.35　送信元IP:192.168.0.111 送信元PORT:3610
 DATA:EHD1:10, EHD2:81, TID:0000, SEOJ:0EF001, DEOJ:0EF001, ESV:73, OPC:01, EPC1:D5, PDC1:04, EDT1:
01029001,
```

（図 4-3-7：一般照明の電文）

４．３．３．電文解析　その２

さて、取得した電文を解析してみましょう。

（図 4-3-7）の電文 No.1 は、送信元 IP：192.168.111 から、Port：3610 で発信されその内容は、次の通りです。

①．EHD1：10（ほぼ固定）
　ECHONET Lite ヘッダ１
②．EHD2：81（ほぼ固定）
　ECHONET Lite ヘッダ２
③．TID：0000（送信元の連続番号と思えばよい）
　Transaction ID
④．SEOJ：0EF001（プロファイルクラス＋ノードプロファイル＋01 番）
　送信元 ECHONET Lite オブジェクト指定
⑤．DEOJ：0EF001（プロファイルクラス＋ノードプロファイル＋01 番）

相手先 ECHONET Lite オブジェクト指定

⑥．ESV：73（プロパティ値通知＝INF）

ECHONET Lite サービス

⑦．OPC：01（以後のプロパティ処理数は１つ）

処理プロパティ数

⑧．EPC：D5（インスタンスリスト通知）

ECHONET Lite プロパティ

⑨．PDC：04（以後のバイト数）

EDT バイト数

⑩．EDT1：01029001

（インスタンス総数＝１＋住宅・設備関連機器クラス＋一般照明機器＋01 番）

プロパティ値データ

となっています。

この電文は、一般照明機器が立ち上がった際に、ネットワーク内のコントローラに一般照明機器ノードが立ち上がったことを通知している電文です。

これらの定義は、【2.3.5.】で説明した、ECHONET Lite 規格書 第 2 部 第 3 章 電文構成に規定があります。参照して、各フィールドの値を確認してください。

（図 4-3-7）の電文 No.2 は、送信元 IP：192.168.111 から、Port：3610 で発信されその内容は、次の通りです。

①．EHD1：10（ほぼ固定）

ECHONET Lite ヘッダ１

②．EHD2：81（ほぼ固定）

ECHONET Lite ヘッダ２

③．TID：0002（送信元の連続番号と思えばよい）

Transaction ID

④．SEOJ：029001（住宅・設備関連機器クラス＋一般照明機器＋01 番）

送信元 ECHONET Lite オブジェクト指定

⑤．DEOJ：0EF001（プロファイルクラス＋ノードプロファイル＋01 番）

相手先 ECHONET Lite オブジェクト指定

⑥．ESV：73（プロパティ値通知＝INF）

ECHONET Lite サービス

⑦．OPC：01（以後のプロパティ処理数は１つ）

処理プロパティ数

⑧．EPC：80（動作状態）

ECHONET Lite プロパティ

⑨．PDC：01（以後のバイト数）

EDT バイト数

⑩．EDT1：30（ON）

（インスタンス総数＝１＋住宅・設備関連機器クラス＋一般照明機器＋01 番）

プロパティ値データ

となっています。

この電文は、一般照明機器のスイッチが入り照明が点灯したことをコントローラ機器に通知する電文です。

（図 4-3-7）の電文 No.3 は、送信元 IP：192.168.111 から、Port：3610 で発信され、その内容は、次の通りです。

①．EHD1：10（ほぼ固定）

ECHONET Lite ヘッダ１

②．EHD2：81（ほぼ固定）

ECHONET Lite ヘッダ２

③．TID：0002（送信元の連続番号と思えばよい）

Transaction ID

④．SEOJ：029001（住宅・設備関連機器クラス＋一般照明機器＋01 番）

送信元 ECHONET Lite オブジェクト指定

⑤．DEOJ：0EF001（プロファイルクラス＋ノードプロファイル＋01 番）

相手先 ECHONET Lite オブジェクト指定

⑥．ESV：73（プロパティ値通知＝INF）

ECHONET Lite サービス

⑦．OPC：01（以後のプロパティ処理数は１つ）

処理プロパティ数

⑧．EPC：80（動作状態）

ECHONET Lite プロパティ

⑨．PDC：01（以後のバイト数）

EDT バイト数

⑩．EDT1：31（OFF）

（インスタンス総数＝１＋住宅・設備関連機器クラス＋一般照明機器＋01 番）

プロパティ値データ

となっています。

この電文は、一般照明機器のスイッチが切れて照明が消灯したことをコントローラ機器に通知する電文です。

５章　モデルスマート家電の開発

いよいよマイコンを使って、モデルスマート家電を開発します。マイコン関連の必要パーツは別途、実習キットが用意されています。必要に応じて、他の章を参照しながら手順どおりに進めて、ECHONET Lite 対応の照明を開発しましょう。

この章では、Arduino の開発に親しみを持ってもらえるように、段階的にプログラムを作ります。

５章１節　必要な機器と環境

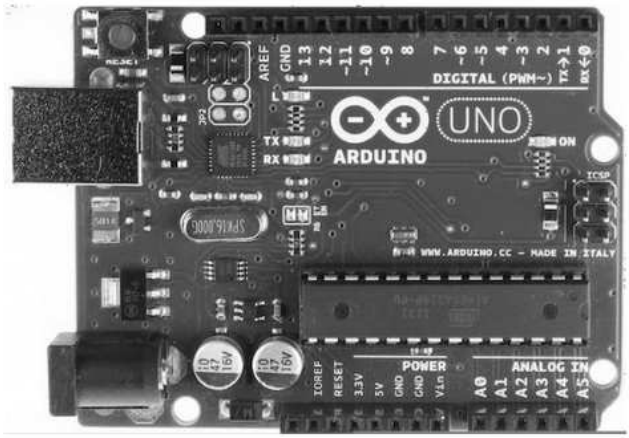
必要な機材と環境を説明します。

５．１．１．PC

PC の OS は、Windows・Linux・MAC が利用できますが、ここでは Windows を利用した開発について解説します。

５．１．２．マイコン（Arduino UNO）

マイコンには、3 章で解説した、Arduino UNO を使います。（図 5-1-1）

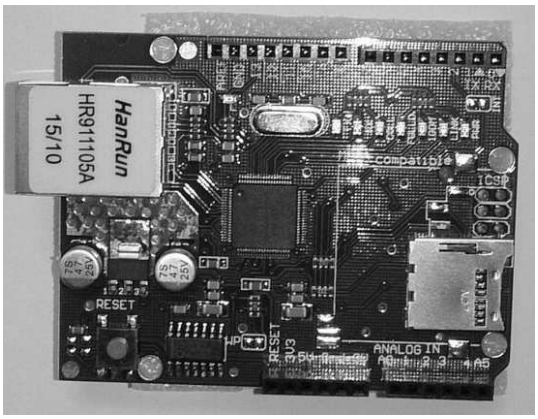


（図 5-1-1：Arduino UNO）

※注意）Arduino ボードは、サードパーティ製のものが沢山作られています。どれを利用しても良いのですが、Web などに公開されている稼働実績のあるものが、問題に直面した際に、課題解決のヒントを得やすくて、安心して利用できます。

５．１．３．Ethernet Shield

Arduino ボードを拡張する際に追加する基盤で、Ethernet Shield は、マイコンをネットワークに接続するために利用します。（図 5-1-2）



（図 5-1-2：Ethernet Shield）

※重要）Ethernet Shield は、MAC アドレスが添付されているものと、そうでないものが存在します。MAC アドレスが添付されていないものは、利用しているネットワーク環境で重複しない MAC アドレスを利用して下さい。実際には、すでに利用していないルーターや古い PC のネットワークカードに割り当てられている MAC アドレスを利用するのがよいでしょう。

５．１．４．開発環境（IDE）

利用するマイコの開発環境は、PC にインストールした IED（統合開発環境）を利用できます（図 5-1-3）。IDE の入手やインストールの手順については、３章２節で説明していますので、適宜参照してください。

（図 5-1-3：Arduino IDE）

５．１．５．LED

Arduino ボードには、LED がついていますが、基板の外で LED を点灯する際に使用します。手持ちのパーツを使用する方は、数 mA の電流で点灯できるような LED を利用してください。

（図 5-1-4：LED）

５．１．６．抵抗

LED を点灯させる際、LED にあまり大きな電流が流れないようにするための抵抗です。抵抗値を LED の規格に合わせて適宜選択します。ここでは 470Ω のものを準備しました。

（図 5-1-5：抵抗 470Ω）

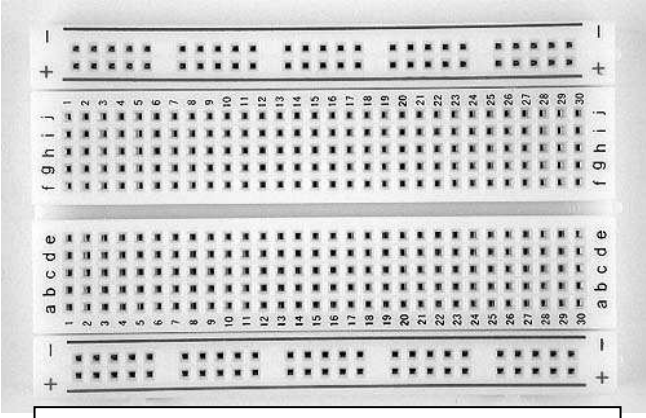
５．１．７．ジャンパワイヤ

マイコン基板のソケットにジャンパワイヤを挿しこみ、周辺デバイスと（例えばセンサなど）接続するので半田付けの必要はありません。いろいろな色があるので、信号の種類によって使い分けをします。

（図 5-1-6：ジャンパワイヤ）

５．１．８．ブレッドボード

Arduino ボード外部に回路を作り、はんだ付けをしないでマイコンと接続することができる便利な基板です。



（図 5-1-7：ブレッドボード）

５章２節　Arduino ボード　初めの一歩

まず、最初に LED を点滅させてみましょう。これが、初めの一歩です。

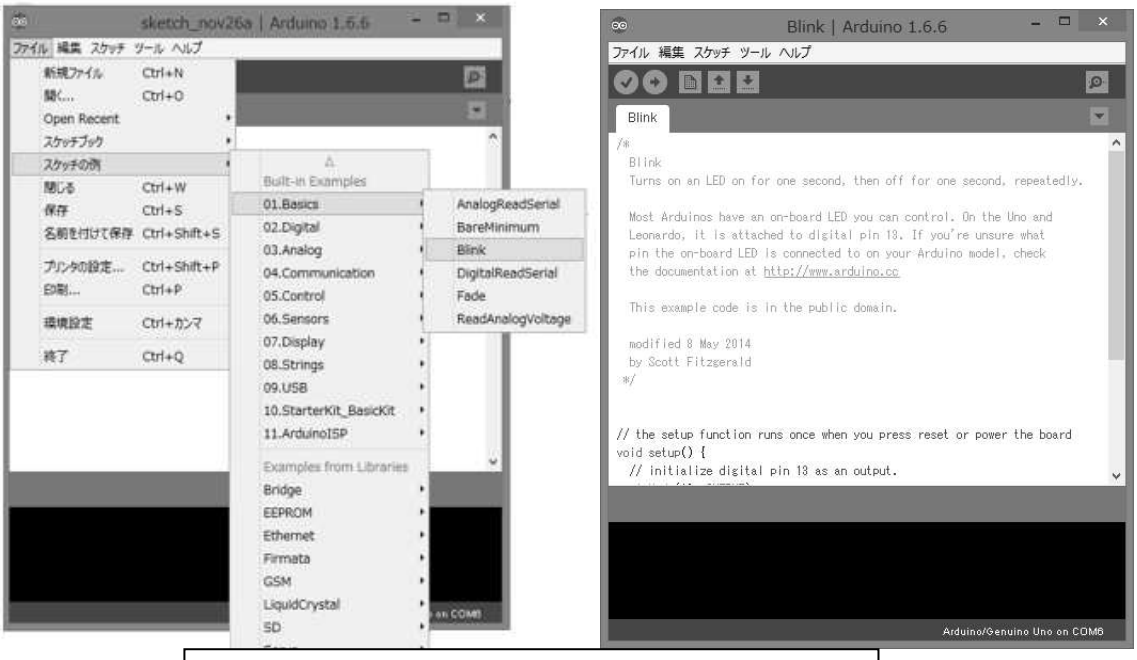
５．２．１．プログラム開発

PC のデスクトップにあるメガネマーク（図 5-2-1）のショートカットをダブルクリックして Arduino IDE を起動します（図 5-2-2）。

（図 5-2-1：Arduino ショートカット）

（図 5-2-2：Arduino IDE）

次に、IDE のメニューから ファイル→スケッチの例→01.Basics とたどり、Blink をクリックします。（図 5-2-3）



（図 5-2-3: スケッチの例を選択と Blink のスケッチ）

すると、Arduino プログラムのひな形が現れます。LED の点滅はこのひな形をそのまま使い、Arduino の動作確認とプログラムの書込み操作の練習をします。初めて利用する機能も、この「スケッチの例」を参考にして開発することができるようになっていて大変便利です。

※Arduino では、プログラムのことを「スケッチ」と呼んでいますが、本書では、

「プログラム」という呼称も使用しています。
プログラムの内容は、後に説明することにして、先を急ぎます。

５．２．２．Arduino にプログラム書込み

次は、Arduino を付属の USB ケーブルで PC に接続してください。PC によっては通知音が鳴り、USB ポートにデバイスが接続されたことを知らせてくれるでしょう。

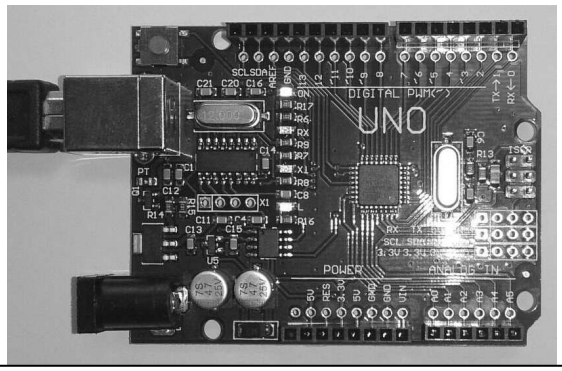
ここで、3 章で確認した COM ポート番号が必要になります。メニューからツール→シリアルポート とたどり、3 章で確認した COM ポート番号にチェックが入っていることを確認してください。3 章で確認・設定したので、そのままで OK のはずです。もし変わっているようでしたら、ここで設定してください。

（図 5-2-4：COM ポート番号の確認）

いよいよ Arduino にプログラムを書き込みます。IDE の左上にある右向きの矢印ボタンをクリックして下さい。すると、プログラムのコンパイルが行われて、エラーが無いことが確認されると、Arduino にプログラムの書込みが行われます。（図 5-2-5）

（図 5-2-5：コンパイル（左）と書込み（右））

書込みが終了すると、自動的に Reset がかかり、プログラムが動き始めます。写真（図 5-2-6）では、下側の赤い LED が点滅を始めています。



（図 5-2-6：LED の点滅）

５．２．３．Blink プログラムの説明

（図 5-2-7）に Blink プログラムを示します。

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

（図 5-2-7：Blink プログラム）

Arduino のプログラムは C 言語で記述します。プログラムには setup()と loop()という 2 つの関数が必要です。setup()関数は、Arduino ボードの電源を入れた時や Reset をしたときに 1 度だけ実行されます。変数や PIN の入出力の初期化・設定などをこの関数内で行います。loop()関数には、実際に行いたい処理を記述します。loop()関数は繰り返し実行されます。この 2 つの関数は省略することができません。

Blink プログラムでは、setup()関数内に pinMode(13,OUTPUT)という記述があります。これは、13 番 PIN を出力で使うという初期設定です。Arduino では 13 番 PIN に LED がつながっているので、これを点滅させるために、出力に設定しています。loop()関数では、degitalWrite(13,HIGH)と degitalWrite(13,LOW)という記述があります。これは、13 番 PIN の出力を HIGH にしたり、LOW にしたりしています。HIGH にすれば、13 番 PIN は 5V になり、そこにつながっている LED が点灯します。反対に LOW にすれば、13 番 PIN は 0V になり LED は消灯します。そして、その間に delay(1000)という記述がありますが、ここで 1000 ミリ秒（1 秒）プログラムの実行を停止します。このプログラムが繰り返し実行されるので、1 秒間隔で LED の点滅を繰り返すプログラム＝Blink になるということです。

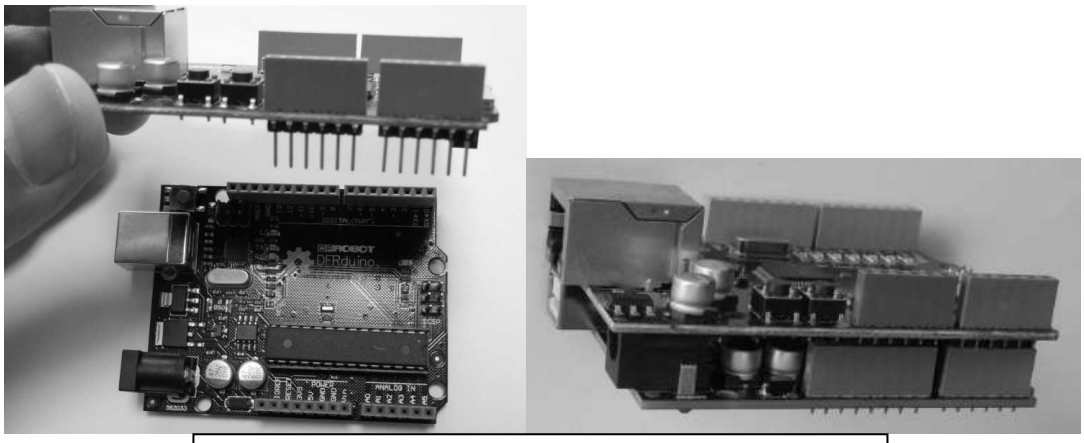
５章３節　ECHONET Lite 対応照明の開発

次は、ECHONET Lite に対応した照明を開発しましょう。

５．３．１．Ethernet Shield の準備と配線

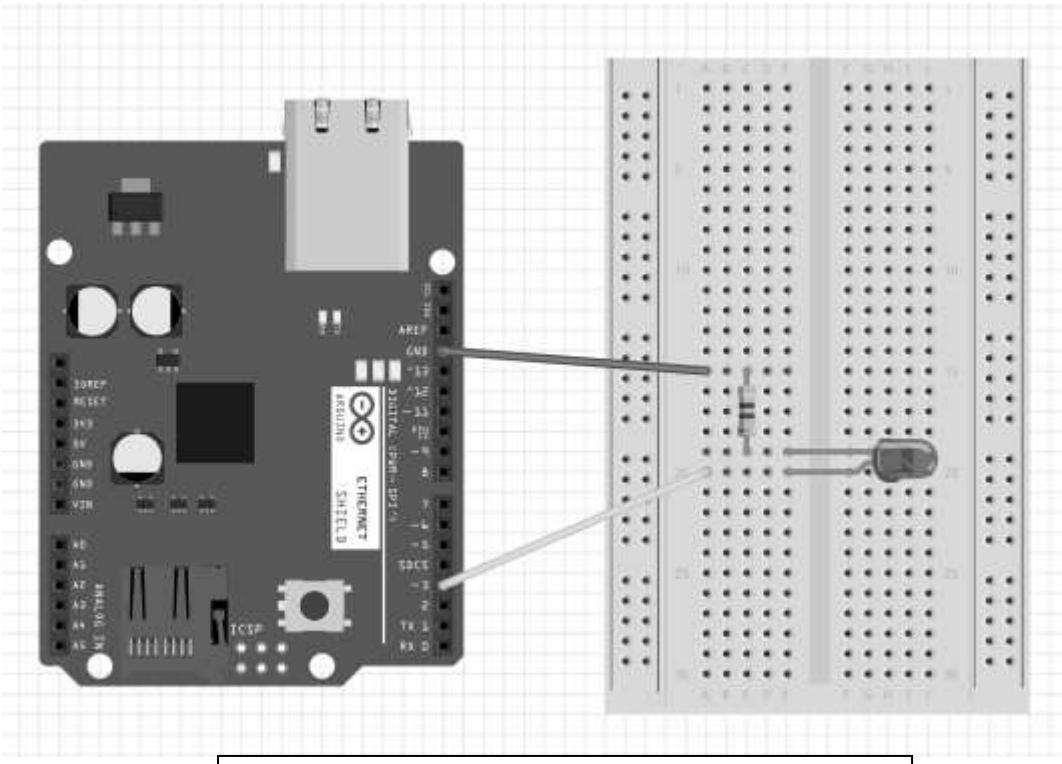
ネットワークに接続するために Ethernet Shield を使います。この拡張基板は、Arduino ボードのピンソケットに差し込める PIN がついていますので、（図 5-3-1）のように Arduino ボードの上に重ねて差し込みます。

※重要）後に Arduino にプログラムを書き込む際に、Ethernet Shield が正しく初期化されていないために、書き込みエラーが発生する場合があります。その時は、Ethernet Shield を一度はずして書込を行った後に、電源を切って（USB ケーブルを外す）Ethernet Shield を取り付けるようにしてください。プログラムで Ethernet Shield が正しく初期化されるようになれば、書き込みエラーが発生することはありません。



（図 5-3-1：Ethernet Shield の準備）

Ethernet Shield を装着したら、次は配線を行います。（図 5-3-2）のように、Ethernet Shield のピンソケットとブレッドボード、LED、抵抗をジャンパー線で配線して回路を作ります。この時、注意が必要な点は、LED の向きです。LED は、脚が 2 本出ていて、長さが違います。（図 5-3-2）で、曲げて表示している側が長い方の脚です。Ethernet Shield のピンソケットは、そのまま Arduino ボードのピンソケットにつながっていますので、回路は No.3 の PIN→黄色のジャンパー線→LED の長い方の脚→LED の短い方の脚→抵抗→青色のジャンパー線→GND PIN　となります。



（図 5-3-2：配線図）

５．３．２．プログラムの作成

Arduino IDE に次のソースコードを記述します。コードの内容は、コメントに記入しましたので、お読みください。ソースコードを入力したら、Arduino を PC に接続して、コンパイルと書込みを行ってください。

※重要）Arduino にプログラムを書き込む際、Ethernet Shield が正しく初期化されていないために、書き込みエラーが発生する場合があります。その時は、Ethernet Shield を一度はずして書込を行った後に、電源を切って（USB ケーブルを外す）Ethernet Shield を取り付けるようにしてください。プログラムで Ethernet Shield が正しく初期化されるようになれば、書き込みエラーが発生することはありません。

（重要事項）

〔1〕の MAC Address は、ネットワーク内でユニークにする事。

〔2〕の IP Address は、DHCP の場合は、自動で設定されます。

```
#include <SPI.h>
#include <Ethernet.h>
#include <Udp.h>

//#define CS 4   //Required Chip select

byte mac[] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };      //〔1〕MAC Address
byte ip[] = { 192, 168, 0, 39 };            //〔2〕IP Address

IPAddress remIP( 192, 168, 0, 101);  //〔3〕コントローラの IP のデフォルト値
                                     //後で返信先の IP が入るので特に意味はない。
unsigned int localPort = 3610;       //〔4〕local port to listen on

long EHD1= 0x10;            //〔5〕ここから、ECHONET Lite の通信電文の定義をしている
long EHD2= 0x81;
long TID1= 0x00;
long TID2= 0x00;
long SEOJ1= 0x02;           //〔6〕一般照明を表す
long SEOJ2= 0x90;
long SEOJ3= 0x01;
long DEOJ1= 0x0E;
```

```
long DEOJ2= 0xF0;
long DEOJ3= 0x01;
long ESV= 0x73;              //〔7〕INF
long OPC= 0x01;
long EPC= 0x80;
long PDC= 0x01;
long EDT= 0x31;              //〔5〕ここまで、ECHONET Lite の通信電文の定義をしている

EthernetClient client;
EthernetUDP Udp;

int RECEIVE_PACKET_MAX_SIZE = 256;
byte rBuffer[256];           //〔8〕receive buffer

int devStatus = 0;           //〔9〕デバイスステータス，OFF=0，ON=1
int RED_PIN   = 3;           //〔10〕照明 LED は、No.3 の PIN に接続している

////////////////////////////////////////////////////////////////////////////
// 初期化 //
void setup() {               //〔11〕
  pinMode( RED_PIN, OUTPUT);    //〔12〕ON OFF 制御は 3 番ポートになる

  //〔14〕シリアルポートをオープンして、しばし待つ
  Serial.begin(9600);        //〔15〕通信速度は 9600bps
  delay(1000);               //〔16〕1 秒間待つ
  Serial.println("Start ECHONET Lite!!");        //〔17〕シリアル通信で、メッセージ出力

    //〔18〕Ethernet 接続開始
  while(1) {
    if (Ethernet.begin(mac) == 0) {              //〔19〕DHCP で IP アドレスをもらう
      Serial.println("Failed to configure Ethernet using DHCP");   //〔21〕NG の場合メッセージ
    }
    else {
      Serial.println("DHCP ok!");                //〔22〕巧く行ってもメッセージ
      break;
    }
```

```
  }

//〔23〕print your local IP address:
  Serial.print("My IP address: ");
  for (byte thisByte = 0; thisByte < 4; thisByte++) {
        // print the value of each byte of the IP address:
    Serial.print(Ethernet.localIP()[thisByte], DEC);
    Serial.print(".");
  }
  Serial.println();  //〔24〕改行

  Ethernet.begin(mac, Ethernet.localIP());      //〔25〕Ethernet の開始
  Udp.begin(localPort);                         //〔26〕ポート指定して UDP 通信開始

  //〔27〕ブロードキャストを使い
  //  機器が立ち上がったことを ECHONET Lite ネットワークに通知する
  //  本当は 0ef0 からのマルチキャストでデバイス通知をするべきだが
  //  簡単実装のためこれで良しとする
  IPAddress remBroad(255, 255, 255, 255 );
  Udp.beginPacket(remBroad, localPort);


  Udp.write(EHD1);          //〔28〕ここから、電文パケット作成。2 章 3 節、4 章参照
  Udp.write(EHD2);
  Udp.write(TID1);
  Udp.write(TID2);
  Udp.write(SEOJ1);
  Udp.write(SEOJ2);
  Udp.write(SEOJ3);
  Udp.write(DEOJ1);
  Udp.write(DEOJ2);
  Udp.write(DEOJ3);
  Udp.write(ESV);           // INF
  Udp.write(OPC);
  Udp.write(EPC);
  Udp.write(PDC);
```

```
  Udp.write(EDT);
  Udp.endPacket();
}

////////////////////////////////////////////////////////////////////////
// main loop
void loop() {                    //〔29〕繰り返し実行。UDP パケットを受信したとき処理をする。
  int remPort = 3610;            //〔30〕UDP ポート番号

  int packetSize = Udp.parsePacket();          //〔31〕UDP パケット解析して、サイズを調べる。
  if(packetSize) {               //〔32〕UDP パケットの有無確認
    Serial.print("packetSize = ");       //〔33〕パケットがあった
    Serial.println( packetSize );

    remIP = Udp.remoteIP();              //〔34〕送ってきた相手の IP Address
    Udp.read( rBuffer, RECEIVE_PACKET_MAX_SIZE);          //〔35〕パケットデータ読込

    Serial.print("packet: ");            //〔36〕ここから、パケット内容のシリアル出力
    for (int i =0; i < packetSize; i++)
    {
      Serial.print(rBuffer[i], HEX);
      Serial.print(" ");
    }
    Serial.println();;                   //〔36〕ここまで、パケット内容のシリアル出力

    ESV = 0x72;                          //〔37〕Get_Res　2 章 3 節参照
    DEOJ1=0x05;                          //〔38〕コントローラ
    DEOJ2=0xFF;
    DEOJ3=0x01;

    if( rBuffer[10] == 0x60 || rBuffer[10] == 0x61 ) {       //〔39〕 SetI or SetC
      if( rBuffer[12] == 0x80 && rBuffer[14] == 0x30 ) {  //〔40〕動作状態＝ON
          // Serial.print("A");           //〔41〕テスト用
        devStatus = 1;                   //〔42〕内部ステータス
      }
      else if( rBuffer[12] == 0x80 && rBuffer[14] == 0x31 ) {        //〔43〕動作状態＝OFF
```

```
      // Serial.print("B");          //〔44〕テスト用
   devStatus = 0;                   //〔45〕内部ステータス
}
else{
   Serial.print("??? 1: ");
   Serial.print(rBuffer[14]);
   Serial.println();
}

   // ON = 1, OFF = 0
if( devStatus == 1 ) {              //〔46〕内部ステータスによって、LED の ON・OFF 制御
   digitalWrite( RED_PIN, HIGH) ;   //〔47〕LED を点灯
   EDT = 0x30;                      //〔48〕後の電文のために・・・
}
else {                              //〔49〕内部ステータスによって、LED の ON・OFF 制御
   digitalWrite( RED_PIN, LOW) ;    //〔50〕LED を消灯
   EDT = 0x31;                      //〔51〕後の電文のために・・・
}

   //〔52〕処理した後のデバイス状態を返信する
Udp.beginPacket(remIP, remPort);    //〔53〕ここから、UDP パケット作成
Udp.write(EHD1);
Udp.write(EHD2);
Udp.write(TID1);
Udp.write(TID2);
Udp.write(SEOJ1);
Udp.write(SEOJ2);
Udp.write(SEOJ3);
Udp.write(DEOJ1);
Udp.write(DEOJ2);
Udp.write(DEOJ3);
Udp.write(ESV);
Udp.write(OPC);
Udp.write(EPC);
Udp.write(PDC);
Udp.write(EDT);                     //ON・OFF により変化
```

```
        Udp.endPacket();                //〔53〕ここまで、UDP パケット作成
      }
    }
    delay(10);                          //〔54〕10ms 待つ
}
//-------------------------
//-------------------------
```

５．３．３．モデルスマート家電の稼働

まずネットワークに接続している PC（開発用 PC でも構いません）で、SSNG（Super Speed Node Generator：3 章参照）を立ち上げて、通信開始ボタンを押しておきます。

（図 5-3-3：SSNG）

次に、作成したモデルスマート家電を Ethernet ケーブルで LAN に接続します。

ここで Arduino ボードに電源を入れます。すでに電源が入った状態でしたら、Ethernet Shield の Reset ボタンを押して再起動してください。すると、（図 5-3-4）のように SSNG の受信データの窓に、ECHONET Lite のパケットデータが現れるはずです。

（図 5-3-4：受信データパケット）

これは、ソースコード〔28〕で作成した、【機器が立ち上がった際の ECHONET Lite ネットワークへの通知電文】を受信したので、その内容が表示されたものです。

Arduino から送られた INF パケットは EPC（動作状態）の EDT が 0x80（電源 OFF）という内容の電文です。

次は、SSNG から開発したモデルスマート家電に電文を送ってみましょう。

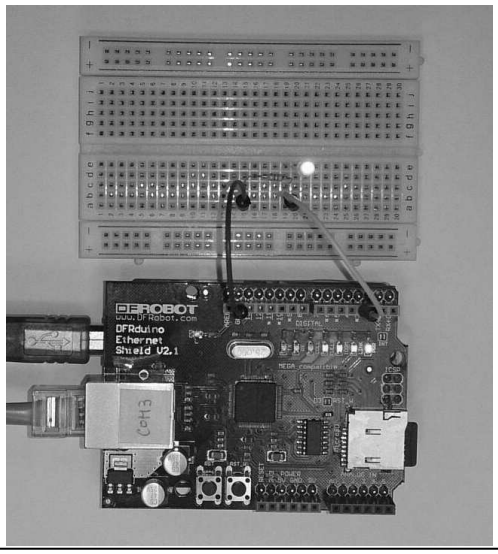
まず、（図 5-3-5）のように、SSNG を設定します。

（図 5-3-5：送信データパケット）

〔1〕接続先リストボックスから、モデルスマート家電の IP を選択します。SSNG は一度パケットを受信すると、相手先の IP アドレスをこのリストボックスに追加してくれるので、次からは選択できるようになっています。

〔2〕EHD=10、EHD2=規定電文形、TID=0000、SEOJ=コントローラ、端末番号=1、EV=書込み、OPC=1、EPC=動作状態、EDT=30（電源 ON）とします。EDT は、直にキー入力してください。

〔3〕これで、左下にある【送信】のボタンを押します。この時、LED が点灯し（図 5-3-6）SSNG には、No.2 のパケットが受信されました（図 5-3-7）。

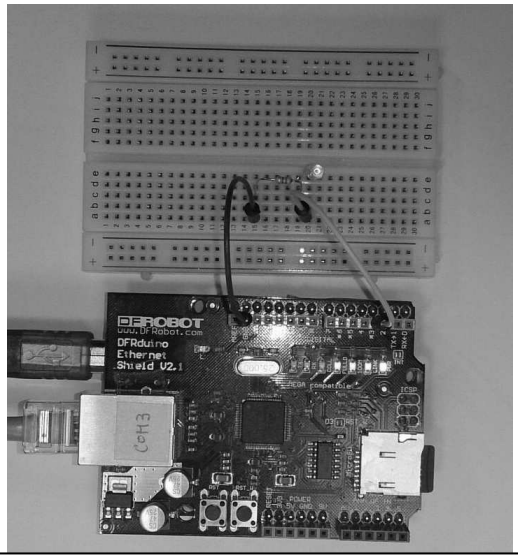


（図 5-3-6：LED 点灯）

（図 5-3-7：LED 点灯時の受信パケット）

〔4〕続けて、EDT=31（電源 OFF）とします。

〔5〕左下にある【送信】のボタンを押します。この時、LED が消灯し（図 5-3-8）SSNG には、No.2 のパケットが受信されました。（図 5-3-9）

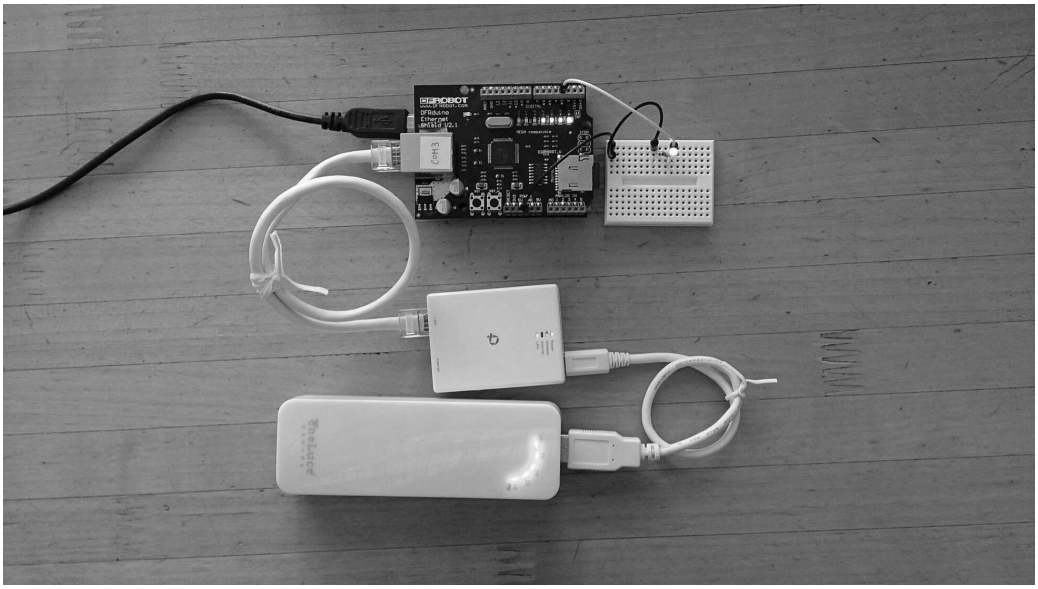


（図 5-3-8：LED 消灯）

（図 5-3-9：LED 消灯時の受信パケット）

５．３．４．モデルスマート家電の拡張（ワイヤレス化）

開発したモデルスマート家電は、Ethernet Shield で LAN 接続されていて、LAN ケーブルでネットワークに参加していますが、ポータブルルータを用いることで、ワイヤレス化が容易にできます。（図 5-3-10）

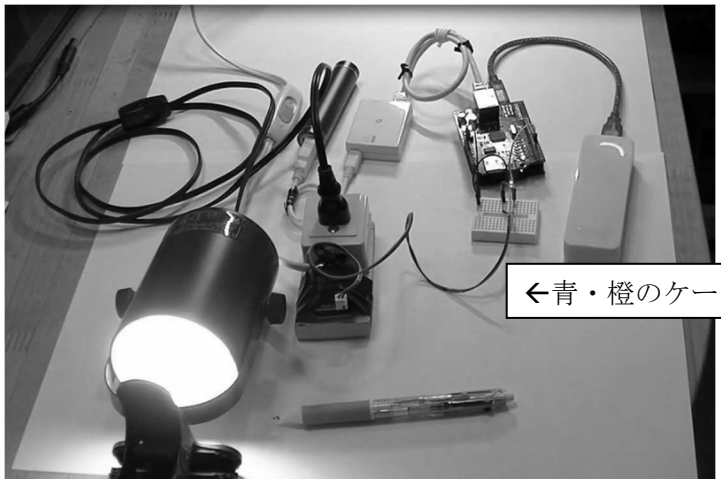


（図 5-3-10：ワイヤレス化）

市販のポータブルルータ（中央の小さな四角い箱）を利用して、ECHONET Lite ネットワークにワイヤレス接続をしています。使用するポータブルルータは「コンバータモード」が必要です。このルーターの設定を正しく行えば、モデルスマート家電に変更を行う必要はありません。

５．３．５．モデルスマート家電の拡張（AC 電源制御）

開発したモデルスマート家電の LED ON/OFF 信号を SSR（Solid-State Relay）に接続すると、AC100V の実際の家電品の ON/OFF 制御を行うことができます。(図 5-3-11)



（図 5-3-11：SSR による AC100V 家電制御）

実習キットには、AC100V 家電の ON/OFF 制御を行う為に、SSR ユニットが含まれています。橙色（または赤）を LED の長い脚と、青（または黒）を GND に接続すると、LED の点灯に同期して、SSR が ON/OFF して、家電品の電源が制御できます。

６章　モデルコントローラの開発

この章では、５章で開発したモデルスマート家電をコントロールする、モデルコントローラの開発を行います。モデルコントローラと云っても、すべての機能を網羅したコントローラではなく、５章のモデルスマート家電の照明の ON/OFF を ECHONET Lite ネットワークを通じて行うリモコンのようなものと考えてください。モデルコントローラには２つの SW があって、片方の SW は、照明を点灯させるための SW で、他方は消灯するための SW です。SW が押されたら、必要な電文を作成して、照明機器に向けて UDP パケット送信します。マイコン関連の必要パーツは別途、実習キットが用意されています。５章を思い出しながら進めましょう。

６章１節　必要な機器と環境・条件

必要な機材と環境は、５章のモデルスマート家電の必要機材に、押しボタンスイッチが２つと、抵抗（100KΩ）２つが追加になっているだけです。実習キットには、モデルスマート家電（５章）とこのモデルコントローラが開発できるパーツが含まれていますので、自分で開発したモデルスマート家電を自分で開発したコントローラ（この章）で制御する実験を行うことができます。

６．１．１．条件

６章のモデルコントローラの開発は、次の条件で行うこととします。

〔1〕モデルスマート家電は、固定 IP アドレスとします。学習目的のため、開発したシステムは、同じ学内のネットワーク上に複数のスマート家電が存在することになり、それらを特定して電文を送ることは困難なために、IP アドレスを固定とします。このために、５章のソースプログラムをよく見て、重要事項を理解したうえで、IP アドレスを固定にする変更を行って、Arduino に書込みをしてください。

〔2〕開発するコントローラは、一つのモデルスマート家電を制御するためだけのものとして開発します。ですから〔1〕で固定した IP アドレスに向けて、UDP 電文を送信するものとします。

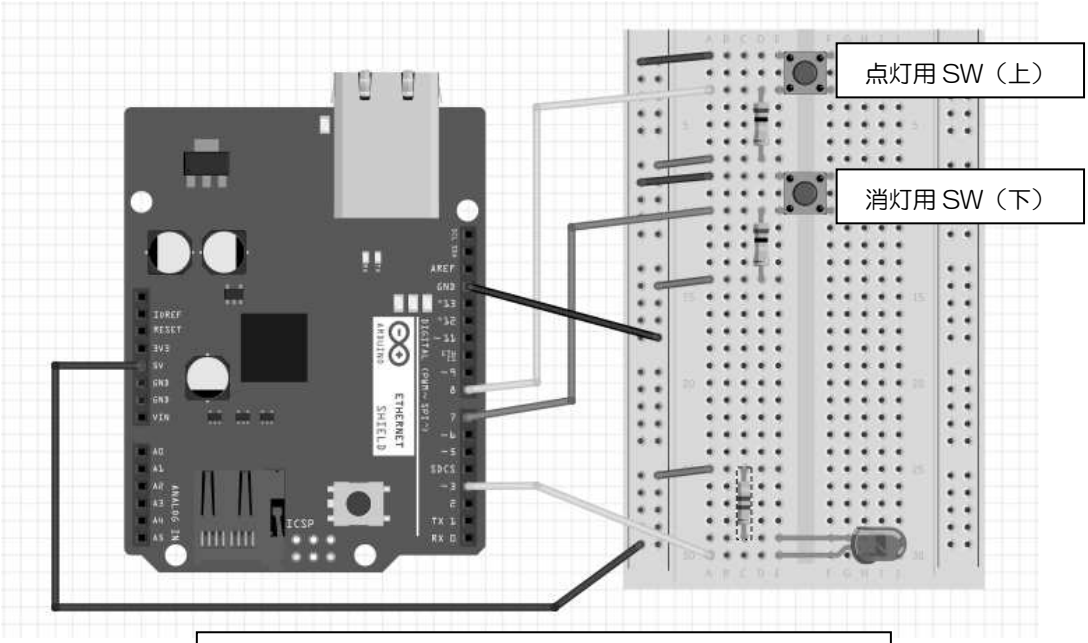
〔3〕本来のコントローラは、様々な機器の情報を取り込み、さまざまな情報を記憶することになりますが、ここで開発するコントローラは、点灯 SW と消灯 SW のみがあり、その SW が押されると、対応する ECHONET Lite 電文を編成

して UDP パケットにして送信するものとします。

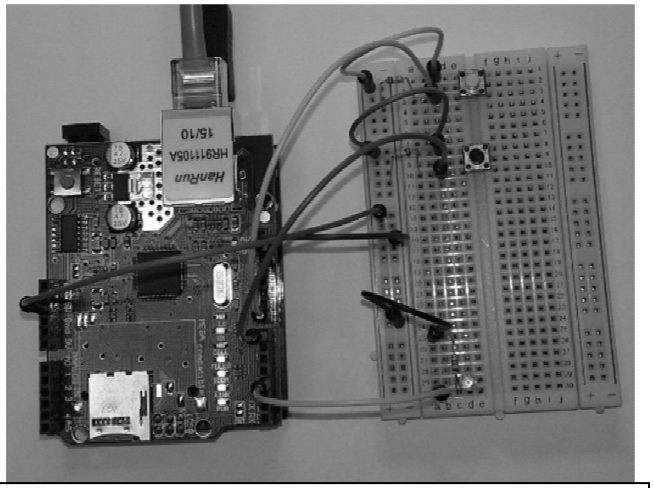
６章２節　配線

（図 6-2-1）の様に配線します。ブレッドボード上部にある SW は、Arduino の No.7,8 PIN に接続されていて、その信号は 100KΩの抵抗で GND にプルダウンされています。ですから、通常は、No.7,8 PIN をマイコンで読むと、LOW になっています。SW の左上の PIN が５V に接続されているので、SW が押されると HIGH になります。

また、No.3 PIN は、5 章と同様に LED に接続します。この LED の目的は、点灯 SW が押されたとき LED を点灯し、消灯 SW が押されたとき LED を消灯する。つまり、コントローラが制御対象とする照明機器のＬＥＤをモニタするのが、このＬＥＤの目的です。



（図 6-2-1：モデルコントローラの配線）



（図 6-2-2：実際のモデルコントローラ）

６章３節　プログラムの作成

Arduino IDE に次のソースコードを記述します。コードの内容は、コメントに記入しましたので、お読みください。ソースコードを入力したら、Arduino を PC に接続して、コンパイルと書込みを行ってください。

※重要）Arduino にプログラムを書き込む際、Ethernet Shield が正しく初期化されていないために、書き込みエラーが発生する場合があります。その時は、Ethernet Shield を一度はずして書込を行った後に、電源を切って（USB ケーブルを外す）Ethernet Shield を取り付けてみてください。プログラムで Ethernet Shield が正しく初期化されるようになれば、書き込みエラーが発生することはありません。

（重要事項）

〔1〕の MAC Address は、ネットワーク内でユニークにする事。

〔2〕の IP Address は、DHCP の場合は、自動で設定されます。

〔3〕の IP Address は、SSNG で調べて設定します。

```
#include <SPI.h>
#include <Ethernet.h>
#include <Udp.h>

//#define CS 4   //Required Chip select

byte mac[] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };      //〔1〕MAC Address
byte ip[] = { 192, 168, 0, 39 };            //〔2〕IP Address

IPAddress remIP( 192, 168, 0, 150);              //〔3〕モデルスマート家電の IP Address
unsigned int localPort = 3610;                   //〔4〕local port to listen on

long EHD1= 0x10;            //〔5〕ここから、ECHONET Lite の通信電文定義
long EHD2= 0x81;
long TID1= 0x00;
long TID2= 0x00;

long SEOJ1= 0x0E;           //〔6〕コントローラ
long SEOJ2= 0xF0;
```

```
long SEOJ3= 0x01;

long DEOJ1= 0x02;
long DEOJ2= 0x90;
long DEOJ3= 0x01;

long ESV= 0x61;                    // SetC
long OPC= 0x01;
long EPC= 0x80;
long PDC= 0x01;
long EDT= 0x31;                    // OFF = 0x31 :   (ON = 0x30)

long EDT1= 0x01;          // 起動時の通知用
long EDT2= 0x05;          // 起動時の通知用
long EDT3= 0xFF;          // 起動時の通知用
long EDT4= 0x01;          //〔5〕ここまで、ECHONET Lite の通信電文の定義

EthernetClient client;
EthernetUDP Udp;

//int RECEIVE_PACKET_MAX_SIZE = 256;
//byte rBuffer[256];          //〔8〕receive buffer

//int devStatus = 0;          //〔9〕デバイスステータス，OFF=0，ON=1（未使用）
int LED_PIN   = 3;          //〔10〕照明 LED は、No.3 の PIN に接続している
int LED_ON    = 8;
int LED_OFF   = 7;

int sw = LOW;


////////////////////////////////////////////////////////////////////////////
// 初期化
void setup() {              //〔11〕
  pinMode( LED_PIN, OUTPUT);                    //〔12〕 LED は 3 番ポートにする
  pinMode( LED_ON, INPUT);                      //〔13〕ON SW
```

```
pinMode( LED_OFF, INPUT);                                    //〔14〕OFF SW

        //〔15〕シリアルポートをオープンして、しばし待つ
Serial.begin(9600);          //〔16〕通信速度は 9600bps
delay(1000);                 //〔17〕1 秒間待つ
Serial.println("Start ECHONET Lite Controler!!");        //〔18〕シリアル通信で、メッセージ出力

        //〔19〕Ethernet 接続開始
while(1) {
  if (Ethernet.begin(mac) == 0) {              //〔20〕DHCP で IP アドレスをもらう
    Serial.println("Failed to configure Ethernet using DHCP");   //〔22〕NG の場合メッセージ
  }
  else {
        Serial.println("DHCP ok!");            //〔23〕巧く行ってもメッセージ

    break;
  }
}

        //〔24〕print your local IP address:
Serial.print("My IP address: ");
for (byte thisByte = 0; thisByte < 4; thisByte++) {
          // print the value of each byte of the IP address:
  Serial.print(Ethernet.localIP()[thisByte], DEC);
  Serial.print(".");
}
Serial.println();  //〔25〕改行

Ethernet.begin(mac, Ethernet.localIP());       //〔26〕Ethernet 開始
Udp.begin(localPort);                          //〔27〕ポート指定して UDP 通信開始

//〔28〕ブロードキャストを使い
//  機器が立ち上がったことを ECHONET Lite ネットワークに通知する
//  本当は 0ef0 からのマルチキャストでデバイス通知をするべきだが
//  簡単実装のためこれで良しとする
IPAddress remBroad(255, 255, 255, 255 );
```

```
Udp.beginPacket(remBroad, localPort);


DEOJ1= 0x0E;                                    //コントローラに
DEOJ2= 0xF0;
DEOJ3= 0x01;

ESV= 0x73;                                      // INF プロパティ値通知
OPC= 0x01;
EPC= 0xD5;                                      // インスタンスリスト通知
PDC= 0x04;


Udp.write(EHD1);          //〔29〕ここから、電文パケット作成。2 章 3 節、4 章参照
Udp.write(EHD2);
Udp.write(TID1);
Udp.write(TID2);
Udp.write(SEOJ1);
Udp.write(SEOJ2);
Udp.write(SEOJ3);
Udp.write(DEOJ1);
Udp.write(DEOJ2);
Udp.write(DEOJ3);
Udp.write(ESV);                                 // INF
Udp.write(OPC);
Udp.write(EPC);
Udp.write(PDC);
Udp.write(EDT1);          //
Udp.write(EDT2);          //
Udp.write(EDT3);          //
Udp.write(EDT4);          //
Udp.endPacket();          //〔29〕ここまで、電文パケット作成。2 章 3 節、4 章参照


DEOJ1= 0x02;                                    // 一般照明
```

```
  DEOJ2= 0x90;
  DEOJ3= 0x01;

  ESV= 0x61;                                    // SetC
  OPC= 0x01;
  EPC= 0x80;
  PDC= 0x01;

}


////////////////////////////////////////////////////////////////////////////
// main loop
void loop() {          //〔30〕繰り返し実行。SW が押されたときに処理をする。
  int remPort = 3610;          //〔31〕UDP ポート番号


  sw = digitalRead(LED_ON);          //〔32〕点灯用 SW 読込
  if(sw == HIGH) {                   //〔33〕押されているか？
    delay(50);                       //〔34〕押されているが、少し待つ（チャタリング対応）
    sw = digitalRead(LED_ON);        //〔35〕もう一回読み
    if(sw == HIGH) {                 //〔36〕押されているか？
      digitalWrite(LED_PIN, HIGH);  //〔37〕モニタ LED 点灯

        //〔38〕ここから、点灯用電文パケット送信
      EDT= 0x30;                                  // OFF = 0x31 :  (ON = 0x30)
      Udp.beginPacket(remIP, remPort);            // (8)
      Udp.write(EHD1);
      Udp.write(EHD2);
      Udp.write(TID1);
      Udp.write(TID2);
      Udp.write(SEOJ1);
      Udp.write(SEOJ2);
      Udp.write(SEOJ3);
      Udp.write(DEOJ1);
      Udp.write(DEOJ2);
```

```
      Udp.write(DEOJ3);
      Udp.write(ESV);
      Udp.write(OPC);
      Udp.write(EPC);
      Udp.write(PDC);
      Udp.write(EDT);
      Udp.endPacket();     //〔38〕ここまで、点灯用電文パケット送信
      delay(500);

      sw = digitalRead(LED_ON);    //〔39〕SW が離されるまで待つ
      while(sw==HIGH){
        sw = digitalRead(LED_ON);
      }
    }
}

sw = digitalRead(LED_OFF);
if(sw == HIGH) {
   delay(50);
   sw = digitalRead(LED_OFF);
   if(sw == HIGH) {
      digitalWrite(LED_PIN, LOW);

      EDT= 0x31;                                    // OFF = 0x31 :   ON = 0x30
      Udp.beginPacket(remIP, remPort);              // (8)
      Udp.write(EHD1);
      Udp.write(EHD2);
      Udp.write(TID1);
      Udp.write(TID2);
      Udp.write(SEOJ1);
      Udp.write(SEOJ2);
      Udp.write(SEOJ3);
      Udp.write(DEOJ1);
      Udp.write(DEOJ2);
      Udp.write(DEOJ3);
      Udp.write(ESV);
```

```
        Udp.write(OPC);
        Udp.write(EPC);
        Udp.write(PDC);
        Udp.write(EDT);
        Udp.endPacket();
        delay(500);

        sw = digitalRead(LED_OFF);
        while(sw==HIGH){
          sw = digitalRead(LED_OFF);
        }
      }
    }
  }
```

６章４節　モデルコントローラの稼働

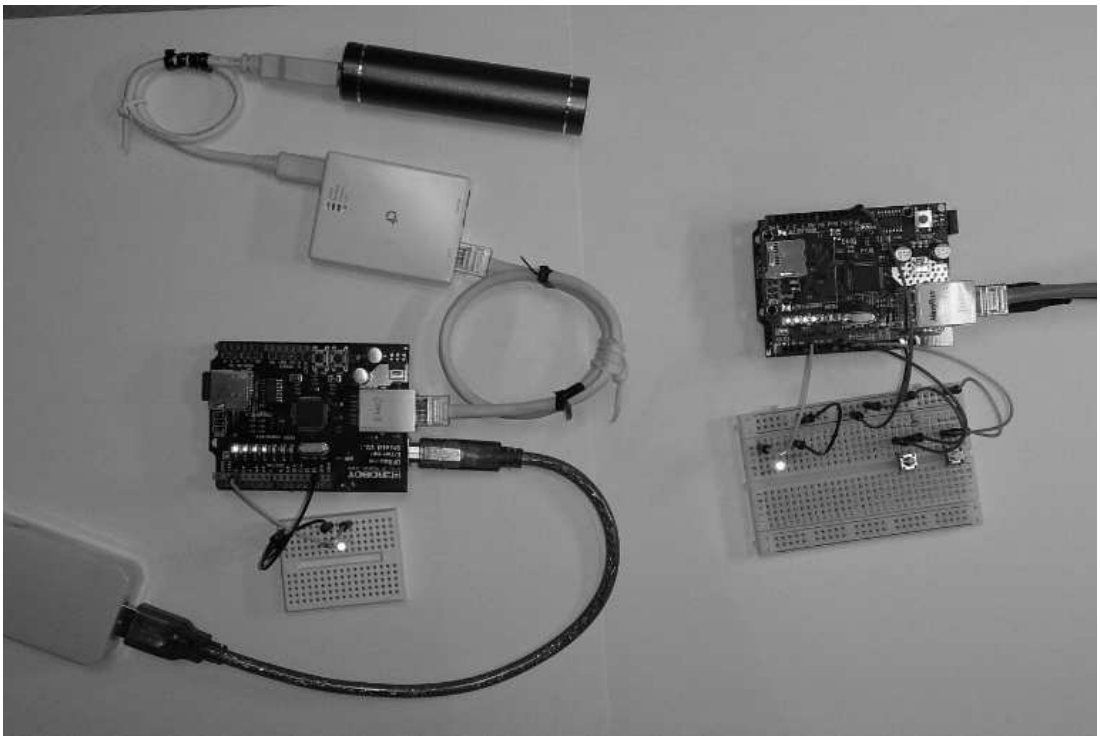
コンパイルと書込みが完了したら、コントローラが正しく動作するか確認しましょう。まず、PC で SSNG を起動して通信開始しておきます。次に、開発したモデルコントローラを LAN ケーブルで ECHONET Lite ネットワークに接続します。電源を入れます。電源が入った状態ならば、Reset ボタンを押して再起動します。

その時、SSNG には、ソースコードの〔29〕で送信したパケットが受信されているはずです。もし、受信されなかった場合は、プログラムの見直し、接続の確認を行いましょう。

SSNG で受信した電文が意図した通りであったら、５章で開発したモデルスマート家電を今回開発したモデルコントローラで制御してみましょう。

LED 点灯用 SW を押します。このときモニタ LED が点灯するはずです。そして、モデルスマート家電の LED が点灯していれば、半分成功です。

次に、LED 消灯用 SW を押します。このときモニタ LED が消灯して、モデルスマート家電の LED も消灯します。完成したモデルコントローラとモデルスマート家電を（図 6-4-1）に示します。



（図 6-4-1：モデルコントローラとモデルスマート家電）

この章のまとめ

実習キットを使いながらのモデル開発はいかがでしたか。Ethernet 上の通信は、Arduino で容易に実現できることが理解できたはずです。スマート家電開発だけではなく、広く応用できるネットワーク通信とデジタル制御のスキルが身についたと思いますので、これからは皆さん独自のアイデアでシステム開発にチャレンジしてみてください。

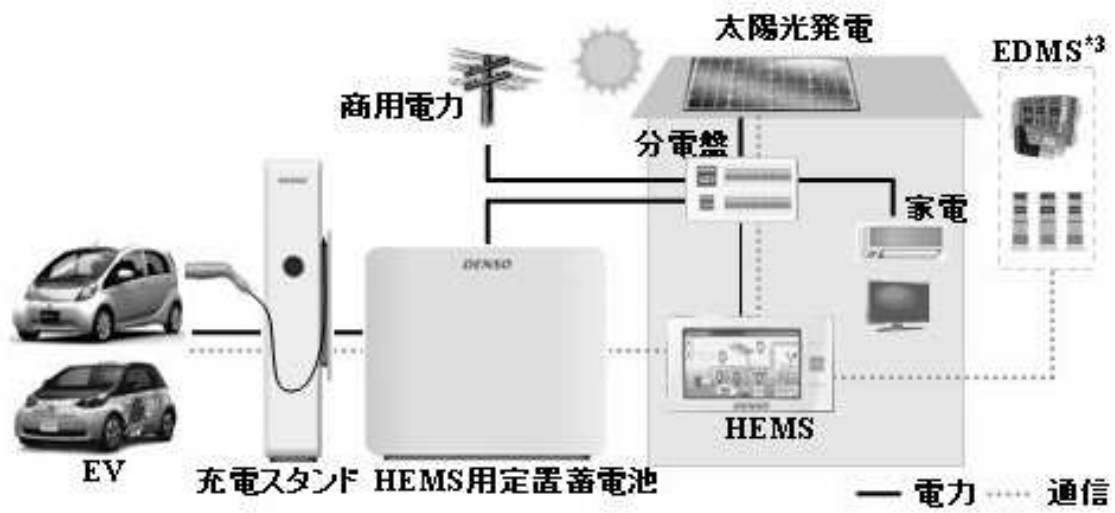
７章　自動車とスマート家電（モデル V2H の開発）

自動車は人やモノを移動するためのツールとしての利用が主な目的でしたが、最近では、大きなディスプレイを搭載した移動広報車（テレビカー）や消防署などで災害の体験をする目的で開発された地震体験車など、多彩な機能を持つ自動車がたくさん開発されています。スマート家電と自動車の関係では、自動車に充電した電力を、HEMS を通じて家庭用電力として利用する機能を搭載した次世代自動車の開発が進められています。

この章では、モデル V2H の開発方法も解説します。前の章までの演習の集大成として、実際に家庭用電源（AC100V）から充電をして動き、ECHONET Lite を経由したコントロールと電源状態のモニタができるようなシステムを目指します。モデル V2H の開発を行うことで、求められる機能と実現方法を考える材料としてください。

７章１節　V2H（Vehicle to Home）

Vehicle to Home とは、電気自動車（EV）やプラグインハイブリッド自動車（PHV）、燃料電池車（FCV）などの自動車が蓄電池に蓄えた電力を家庭用電力として利用することを指します。



（図 7-1-1：V2H 概念図　出典：DENSO）

自動車を敷地内に駐車している間は、自動車の蓄電池を家庭内の発電システムの一部として、充電・放電するという双方向な電力のやりとりを行うことができます。

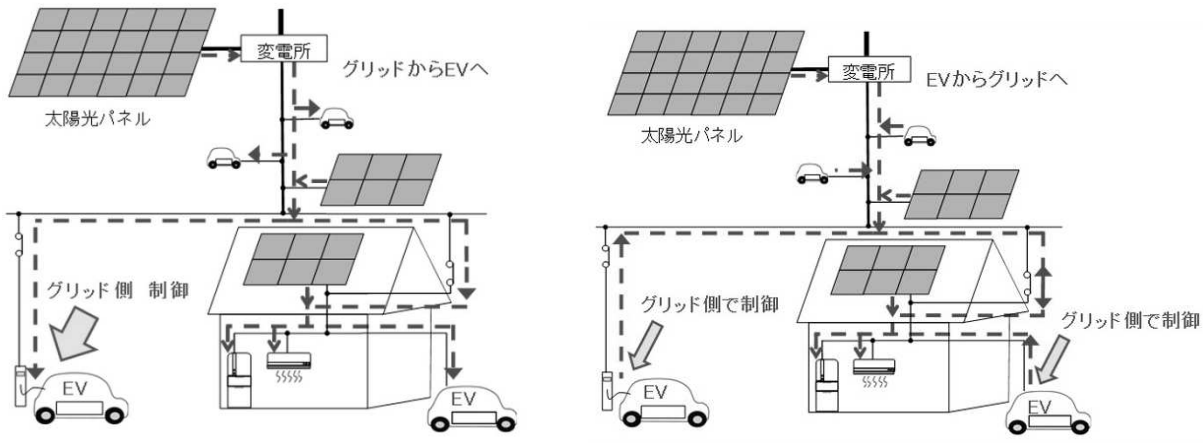
このV2Hを実現するには、自動車の蓄電池が蓄積している直流の電力を家庭で利用できる交流の電力に変換しなくてはなりません。そのため双方向の直流／交流変換回路（インバーター）を用意する必要があります。

また、次世代自動車には、V2Hを含む３つの機能も期待されています。

名称	電気の流れ	期待される機能
Vehicle to Home（V2H）	BEV・PHEVから家庭へ	車両の電力を家庭用の電力供給減として利用。電力系統への連携。 逆潮流の有無でその役割等に違いあり。
Grid to Vehicle(G2V)	グリッドからBEV・PHEVへ	車両への単純な充電だけではなく、太陽光・風力発電からの余剰電力のバッファ等として利用。
Vehicle to Grid(V2G)	BEV・PHEVからグリッドへ	車両から電力系統に電力を供給。電力系統の周波数調整・需給調整等へ利用。（アンシラリーサービス）

（図 7-1-2：期待されている機能　出典：次世代自動車振興センター）

」※上の表でV2Gに期待される「アンシラリーサービス」というのは、送配電ネットワークにおいて必要となる電気の品質維持のためのサービスのことを指しています。具体的には、系統制御、発電設備からの無効電力の供給・電圧制御、需給バランス調整・周波数制御、事故により周波数が低下した場合の発電機出力の増加などをおこなうサービスのことです。



（図 7-1-3：G2V(左)とV2G（右）　出典：次世代自動車振興センター）

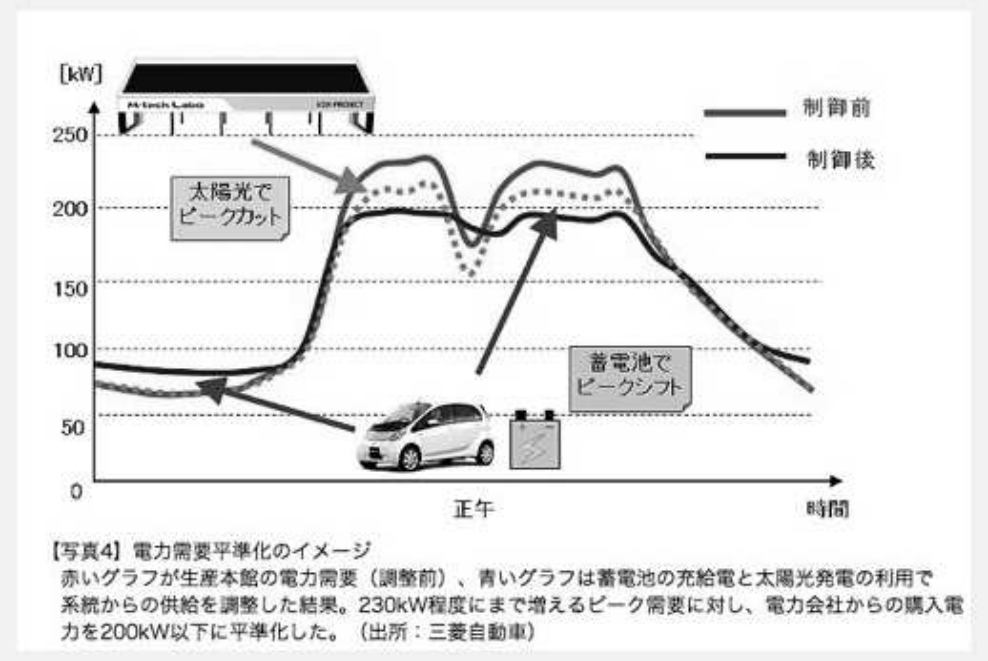
７章２節　スマートハウスでのV2H

家庭でV2Hを導入する大きなメリットは、安価な深夜電力で自動車を充電しておくと、昼間のピーク時に電力会社からの電力を使わずに済ませるピークシフトが可能に

なることが挙げられます。

さらに、自動車と太陽光発電などの自家発電システムを接続し、電気供給力を上げることもできます。昼間は太陽光発電システムからの電力をメインに利用しながら自動車を充電し、夜間は自動車からの電力を家庭で利用するとよいでしょう。

また、停電時に蓄電池の電力がどうしても足りなくなった場合、充電ステーションまで自動車を走らせて充電してくることも可能です。非常時に電力を供給しやすい点は自動車ならではの強みです。



【写真4】電力需要平準化のイメージ
赤いグラフが生産本館の電力需要（調整前）、青いグラフは蓄電池の充給電と太陽光発電の利用で系統からの供給を調整した結果。230kW程度にまで増えるピーク需要に対し、電力会社からの購入電力を200kW以下に平準化した。（出所：三菱自動車）

（図 7-2-1：消費電力の平滑化　出典：三菱自動車）

7章3節　出力 10kw・・・法律について少し・・・

ここまでの説明では、V2H を説明する図に EV が使われていましたが、2016 年 1 月現在では、FCV も販売が開始されており、V2H を実現するための有力な候補になっています。単純に考えると、EV も FCV も電気で走る自動車であることは同じなのですが、電気事業法という法律では、V2H において、電源となる自動車や供給設備などは「電気工作物」という区分となっています。現在この法律に従い EV は実用化されてきたのですが、FCV は新たな技術開発により開発されたものなので、EV と同様に扱えるかどうか、慎重に検討が進められました。

ここでは、その経緯についての記録を引用し、V2H を行う際の出力制限が 10kW となっている理由について説明します。

７．３．１．FCV で V2H を行うための法整備

経済産業省では、平成 25 年に閣議決定された「燃料電池自動車からの一般住宅等への給電（V2H）の実施に向けた電気事業法整備」という規制改革実施計画に基づいて、同法律の整備を行うべく検討が開始されました。その際の貴重な記録が入手できたので、以下にこれを抜粋したものを掲載します。

国や関係省庁などが行っている仕事については、このテキストをお読みいただいている学生の方々は、私も含めて平時あまり関心や理解が進まないところだと思いますが、以下に掲載する記録は、大変読みやすくまとめられていますので、お読み下さい。

◇まず、平成 26 年 10 月に公開された省令の一部改正についての 2 ページの文書です。ここには、検討の結果がまとめられていて、FCV が V2H を行う際 10kW 未満の出力の場合に限り、電気事業法で定める一般工作物として区分し、このことでそれまで必要であった、保安規定の届出や主任技術者などの選任がなくとも保安が確保されると判断されたということです。要するに「家庭でも FCV による給電が法律上 EV と変わらず行うことができるようになる。」ということが記述されています。

電気事業法施行規則及び発電用火力設備に関する技術基準を定める省令の一部改正について

平成２６年１０月
経済産業省
商務流通保安グループ
電力安全課

１．Ｖ２Ｈ（Vehicle to Home，自動車から家等への給電）を行う際の燃料電池自動車の取扱いについて

（１）背景・検討結果

電気事業法施行令（昭和４０年政令第２０６号）第１条第１号の規定により、燃料電池自動車に設置される燃料電池発電設備を当該自動車の動力を得るために利用する限りにおいては、当該燃料電池発電設備は、電気事業法（昭和３９年法律第１７０号）第２条第１６号に規定する電気工作物に該当せず、電気事業法の保安規制の適用対象外となっている。

しかし、燃料電池自動車で発電し、家等の燃料電池自動車以外の場所に給電（以下「Ｖ２Ｈ」という。）を行う場合には、上記のような場合に該当せず、電気工作物として電気事業法の保安規制の適用を受けることとなる。また、このようにＶ２Ｈを行う場合、燃料電池自動車の燃料電池発電設備は、電気事業法施行規則（平成７年通商産業省令第７７号）第４８条第４項で規定している小出力発電設備に該当しないため、電気事業法第３８条第１項に規定する一般用電気工作物にはならず、事業用電気工作物（一般用電気工作物以外の電気工作物）として区分されるため、保安規程の届出（同法第４２条第１項）や主任技術者の選任（同法第４３条第１項）等が必要となる。

今般、Ｖ２Ｈを行いたいというニーズや燃料電池自動車が実用化されつつある状況を踏まえ、燃料電池自動車の保安規制上の取扱いを検討したところ、道路運送車両法（昭和２６年法律第１８５号）及び高圧ガス保安法（昭和２６年法律第２０４号）に基づく規制が燃料電池自動車に課せられていることを考慮すれば、１０ｋＷ未満の出力で給電を行う場合に限り、電気事業法上一般用電気工作物として区分し、保安規程の届出や主任技術者の選任等がなくとも保安が確保できると判断された（詳細は参考資料参照）。

（２）改正内容

今回の改正では、下記の改正を行う。

①電気事業法施行規則第４８条の改正

燃料電池自動車（※１）に設置される燃料電池発電設備であって、道路運送車両法の規制を満たし、１０ｋＷ未満の出力でＶ２Ｈを行うためのものを、小出力発電設備に位置付けることで、電気事業法上一般用電気工作物として扱うこととする。

※１　二輪自動車、側車付き二輪自動車、三輪自動車、カタピラ及びそりを有する軽自動車、大型特殊自動車、小型特殊自動車並びに被牽引自動車以外のものに限る。

②発電用火力設備に関する技術基準を定める省令（平成９年通商産業省令第５１号）の改正

燃料電池自動車に設置される燃料電池発電設備であって、道路運送車両法の規制を満たし、１０ｋＷ未満の出力でＶ２Ｈを行うためのものについては、

（ⅰ）一般用電気工作物にのみ適用されている、非常停止装置に関する技術基準（同省令第３４条第２項）

（ⅱ）燃料ガスの置換に関する技術基準（同省令第３５条本文（※２））

を適用しないこととする。

※２　燃料電池設備の燃料ガスを通ずる部分は、不活性ガス等で燃料ガスを完全に置換できる構造のものでなければならない。

２．スターリングエンジン発電設備の取扱いについて

（１）背景・検討結果

シリンダー内の作動ガスを外部から加熱・冷却し、その体積変化によりエネルギーを得るスターリングエンジンを活用した発電設備（以下「スターリングエンジン発電設備」という。）については、電気事業法上火力発電設備に該当するが、電気事業法施行規則第４８条第４項で規定している小出力発電設備に該当しないため、電気事業法第３８条第１項に規定する一般用電気工作物にはならず、事業用電気工作物（一般用電気工作物以外の電気工作物）として区分されるため、保安規程の届出や主任技術者の選任等が必要となる。また、実用化事例が乏しかったことから、スターリングエンジン発電設備固有の技術基準が整備されていなかった。

今般、実際にスターリングエンジン発電設備を実用化する事例が出てきたことを踏まえ、電気事業法上の安全性について検討した結果、作動ガスとして不活性ガスを利用し、加熱用熱源が小出力（暖炉の排気熱程度）となる１０ｋW未満の出力のスターリングエンジン発電設備については、安全性が十分に高いため、一般用電気工作物として区分することが適当との結論を得た。

また、スターリングエンジン発電設備固有の技術基準の内容を整備した。

（２）改正内容

今回の改正では、下記の改正を行う。

①電気事業法施行規則第４８条の改正

作動ガスとして不活性ガスを利用する出力１０ｋW未満のスターリングエンジン発電設備を一般用電気工作物に位置付ける。

②発電用火力設備に関する技術基準を定める省令の改正

内燃力発電設備等の他の発電用火力設備に関する技術基準を参考に、下記事項に関するスターリングエンジン発電設備の技術基準を新設する。

（ⅰ）スターリングエンジン及びその附属設備の材料

（ⅱ）スターリングエンジン及びその附属設備の構造

（ⅲ）調速装置

（ⅳ）非常停止装置

（ⅴ）計測装置

３．今後のスケジュール（予定）

平成２６年１１月５日　公布・施行

◇次に、この結論に至る検討過程を審議した10ページの資料です。この資料では、EV と FCV の違いを分かりやすく説明し、両者を同一の取扱いとするためにどのように解釈することにしたのかが解説されています。

（審議）燃料電池自動車からの一般住宅等への給電（V2H）の実施に向けた法的環境整備について

平成26年3月10日
商務流通保安グループ
電力安全課

1．検討の経緯

平成25年6月14日に閣議決定された「規制改革実施計画」において、以下が決定された。

＜規制改革実施計画（平成25年6月14日閣議決定）（抄）＞

事項名	規制改革の内容	実施時期
燃料電池自動車からの一般住宅等への給電（V2H※）の実施に向けた電気事業法の整備	燃料電池自動車を活用して一般住宅等への給電を行う場合において、安全性に関する技術的検証を踏まえ、一定の出力未満の場合は燃料電池自動車を小出力発電設備（一般用電気工作物）として位置付ける検討を行い、必要に応じ法的環境整備を行う。	平成25年度検討・結論、結論を得次第措置

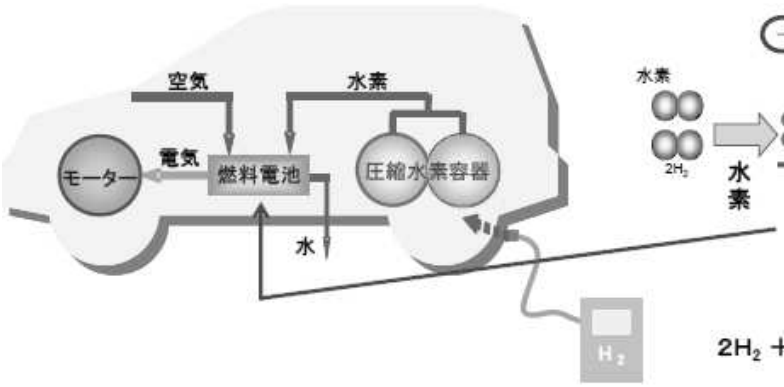
※ V2H（Vehicle to Home）：自動車を電源として住宅等に給電すること。

2. 燃料電池自動車、燃料電池の仕組み

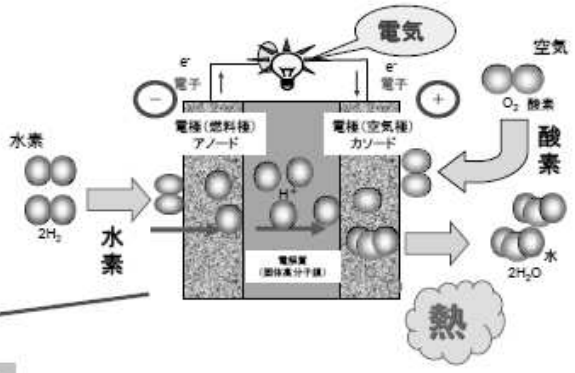
① 「燃料電池自動車」は、「燃料電池」で発電した電気エネルギーを使って、モーターを回して走る自動車。ガソリン内燃機関自動車が、ガソリンスタンドで燃料を補給するように、燃料電池自動車は水素ステーションで燃料となる水素を補給する。

② 「燃料電池」は、水素と空気中の酸素の反応により、水の電気分解と逆の化学反応を利用して電気を発生する仕組み。

＜燃料電池自動車の仕組み＞



＜燃料電池の仕組み＞



$2H_2 + O_2 \rightarrow 2H_2O$ ＋ 電気 ＋ 熱

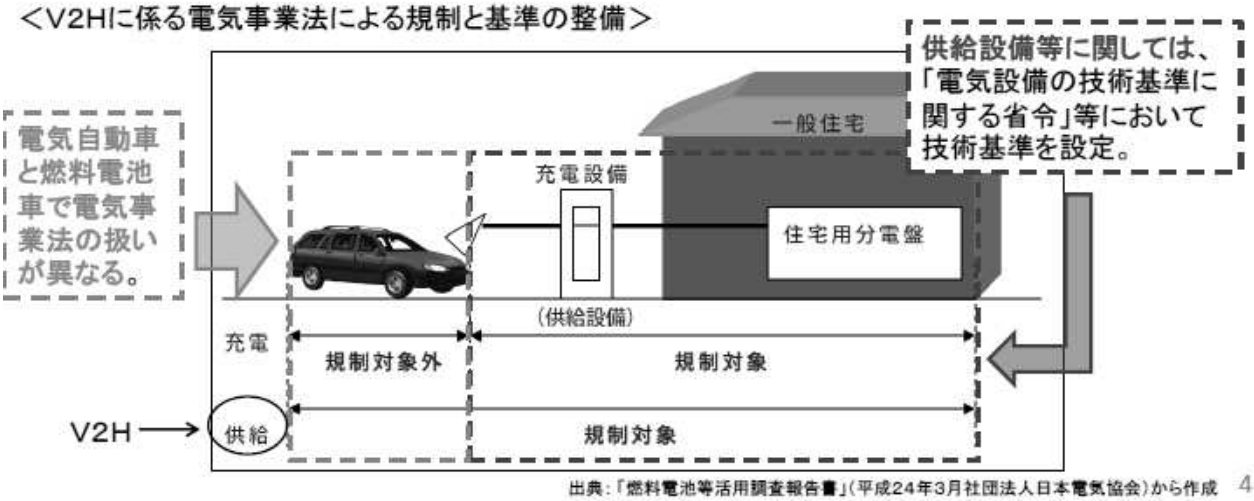
3. V2Hに係る電気事業法の整備の状況

① V2Hにおいて、電源となる自動車や、供給設備・電線・接続器等（以下「供給設備等」という。）は、電気事業法上の電気工作物に該当することから、電気事業法における安全を確保する必要がある。

② 供給設備等に関しては、その技術基準である「電気設備の技術基準に関する省令」に関して、V2Hに必要な安全対策の検討を行い、具体的な給電方法や電気自動車等の出力を10kW未満とすること等を「電気設備の技術基準の解釈」において措置済み（平成23年7月に第199 条の2の追加）。

③ 電源となる電気自動車等に関しては、その技術基準は、電気自動車と燃料電池自動車で異なる。（次頁において電気事業法上の扱いの違いを示す。）

＜V2Hに係る電気事業法による規制と基準の整備＞



出典：「燃料電池等活用調査報告書」（平成24年3月社団法人日本電気協会）から作成

3. V2Hに係る電気事業法の整備の状況（続き）

① 電気自動車を電源とするV2Hは、電気自動車が電気事業法上「電力貯蔵装置」とみなすことから、「一般用電気工作物」に区分され、保安規程、主任技術者、工事計画の届出等が不要。
② 燃料電池自動車を電源とするV2Hは、燃料電池自動車を電気事業法上「燃料電池発電所」とみなすことから、「自家用電気工作物」に区分され、保安規程、電気主任技術者が必要。
③ 一定の出力未満の燃料電池自動車が「一般用電気工作物」として位置づけられるかどうか検討する。
④ 燃料電池自動車は、燃料電池設備として「電気設備の技術基準を定める省令」（以下「電技」という。）及び「発電用火力設備に関する技術基準を定める省令」（以下「火技」という。）が適用される。
⑤ 「火技」の燃料電池設備に対する規定は、発電所扱いとなる燃料電池設備と家庭用燃料電池を前提とした規定であり、燃料電池自動車を電源とするV2Hについては考慮していない。

＜V2Hにおける電気自動車等の電気事業法の保安規制

電源となる電気自動車等	電気工作物の区分	保安規制
電気自動車（電力貯蔵装置）	一般用電気工作物（600V以下で受電※、又は一定出力未満の小出力発電設備で受電線路以外の線路で接続されていない等安全性の高い電気工作物） （例）バッテリー、出力10kW未満の内燃力発電設備、出力10kW未満のエネファーム等	保安規程、主任技術者、工事計画の届出は、不要。 電技維持義務
燃料電池自動車（燃料電池発電所） 電気自動車と燃料電池自動車は扱いが異なる。	自家用電気工作物（事業用電気工作物のうち、電気事業の用に供する電気工作物以外のもの） （例）発電所、工場・ビル等の600Vを超えて受電する需要設備	保安規程、電気主任技術者が必要。工事計画は500kWを超えない限り不要。 電技、火技の維持義務

※構内の電気工作物から構外の電線路へ電気を送り出すことも「受電」に含まれる。

5

4. 検討結果

（1）検討の進め方

① 燃料電池自動車は、改質器を伴う家庭用燃料電池（エネファーム等）とは構造的に差異があることから、V2Hを行う場合に電源となる燃料電池自動車の電気事業法上の安全性について評価するため、自動車業界等からの情報提供や協議を踏まえ、検討を進めた。
② 「一般用電気工作物」として位置づけられるかについては、既に「一般用電気工作物」として位置づけられている電気自動車との比較を行った。

（2）検討の前提

① 燃料電池自動車が停止状態でV2Hを行うときの安全性についての検討する。
② 一般用電気工作物として扱うことができるかの検討であるため、出力は、一般用電気工作物に電気を供給する場合は10kW未満とされていることを踏まえ、これと整合する水準とすることを前提とする。

＜一般用電気工作物に電気を供給する場合の出力制限＞

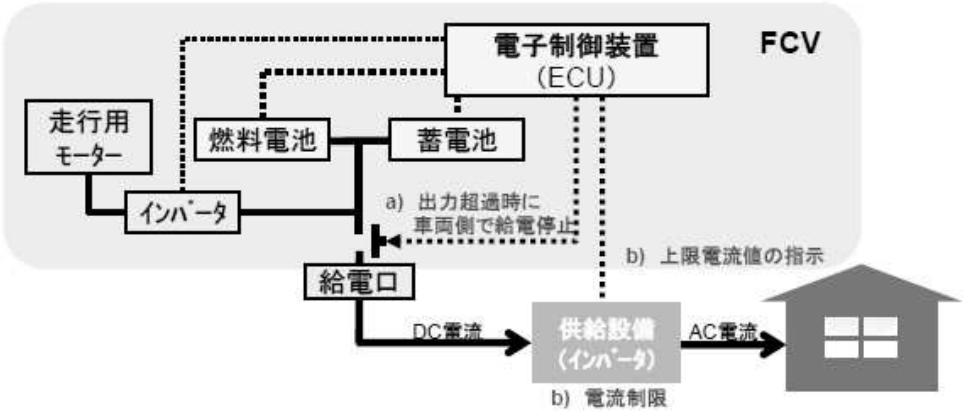
電気設備の技術基準の解釈（抄）

【電気自動車等から電気を供給するための設備等の施設】（省令第4条、第7条、第44条第1項、第56条第1項、第57条第1項、第59条第1項、第63条第1項）

第199条の2 電気自動車等（道路運送車両の保安基準（昭和26年運輸省令第67号）第17条の2第3項に規定される電力により作動する原動機を有する自動車をいう。以下この条において同じ。）から供給設備（電力変換装置、保護装置又は開閉器等の電気自動車等から電気を供給する際に必要な設備を収めた筐体等をいう。以下この項において同じ。）を介して、一般用電気工作物に電気を供給する場合は、次の各号により施設すること。
一 電気自動車等の出力※は、10kW未満であるとともに、低圧幹線の許容電流以下であること。

4. 検討結果（続き）

（3）10kW未満の出力制限について



○ a）燃料電池が出力超過になると、ECU指示により車両からの給電が停止する。

○ b）供給設備（インバーター）でも電流制限をしていることから、上限電流値以上の電流が流れると、給電が停止する。

○ECUのプログラムは暗号化されていることから出力制限は可能と言える。

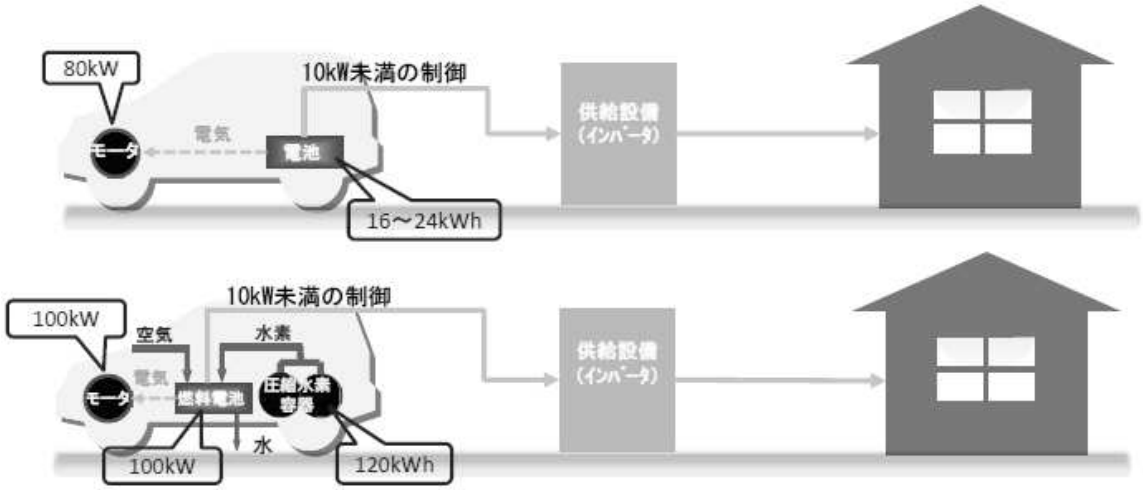
7

4. 検討結果（続き）

（4）電気自動車と燃料電池自動車の違い

① EVも燃料電池自動車も、80から100kWのモータを駆動する出力を有しているが、外部給電する場合、供給設備の能力に関わらず出力は10kW未満に制御される。

② 電気自動車と燃料電池自動車の違いは搭載する電池の種類（電気自動車は、リチウムイオン電池、ニッケル水素電池、鉛蓄電池等であり、燃料電池自動車は燃料電池）の違いであり、燃料電池部分（圧縮水素容器を含む）の安全性以外は、電気自動車と燃料電池自動車に違いがない。



8

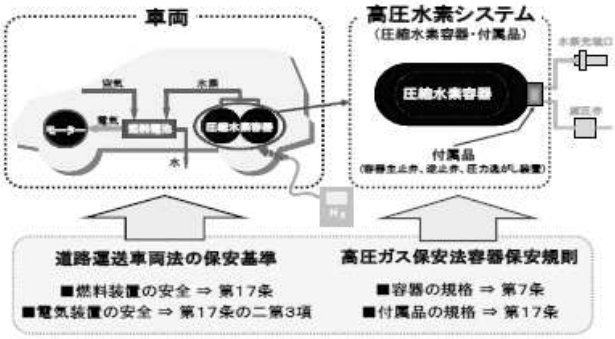
4. 検討結果（続き）

（5）燃料電池部分の安全性

① 燃料電池部分（圧縮水素容器を含む）の安全性については、道路運送車両法及び高圧ガス保安法で電気事業法と同様の要求がなされており、道路運送車両法で定めている技術基準等を精査したところ、電気事業法で求める安全性を満たしていると評価できる。

② しかし、電気事業法で想定している燃料電池発電所や家庭用燃料電池と燃料電池自動車では、設備の構造が異なるため、現行規定では、条文に適合していることが明確ではないところがあるため、必要な法的環境整備を行う必要がある。

＜燃料電池自動車に対する関係法規による規制＞



電気事業法の要求			道路運送車両法保安基準	
電技	4条、15条、58条	感電・火災等の防止、地絡に対する保護対策、電気使用場所に施設する電気機械器具の感電、火災等の防止	17条の2第3項	電気装置（火花、高電圧からの乗車人員の保護）
火技	30条	燃料電池設備の材料	17条	高圧ガスを燃料とする自動車の燃料装置（圧縮容器・付属品については、高圧ガス保安法容器保安規則を引用） ・材料、構造、安全弁、ガス漏洩対策、非常停止装置等について規定
	31条	燃料電池設備の構造等		
	32条	安全弁等		
	33条	ガス漏洩対策等		
	34条	非常停止装置		
	35条	燃料ガスの置換	／	改質器がないため、規制の必要がない。
	36条	空気系統設備の施設		
／	／	該当無し	17条の2第1項、第2項、第4項	・電波による無線や制御装置への影響の防止。 ・衝突、追突等による衝撃を受けた場合でも高電圧による乗車人員への傷害等を生ずる恐れが少ない保護性能、構造

4. 検討結果（続き）

（5）結論

①V2Hを行う燃料電池自動車は、10kW未満の制限をつけることにより、V2Hを行う電気自動車と同様の安全性であると評価できる。

②燃料電池自動車の燃料電池設備としての安全性は、道路運送車両法、高圧ガス保安法により、電気事業法で要求する安全水準を担保できていると評価できる。

③以上のことから、V2Hを行う燃料電池自動車を「一般用電気工作物」として位置付けることが妥当であり、V2Hを行う燃料電池自動車を「一般用電気工作物」として位置づけるために必要な法的整備を行うこととする。

5. 今後のスケジュール

平成26年3月10日　　電力安全小委員会で審議
平成26年春　　パブリックコメント等を実施し、必要な措置をとる。

◇いかがでしたか。このような検討が行われて、EV と FCV が V2H を行う状態においては同じ取扱いができ、EV も FCV も安全に V2H を提供できる環境が法律上整い、メーカーが法律に沿って製品開発を行うことで社会環境が整備されてゆくという流れです。今回は V2H についての検討経過を紹介しましたが、実は一般の家電品や食品、医薬品、衣料品など、私たちの身の回りのものは全てこのような検討が行われ

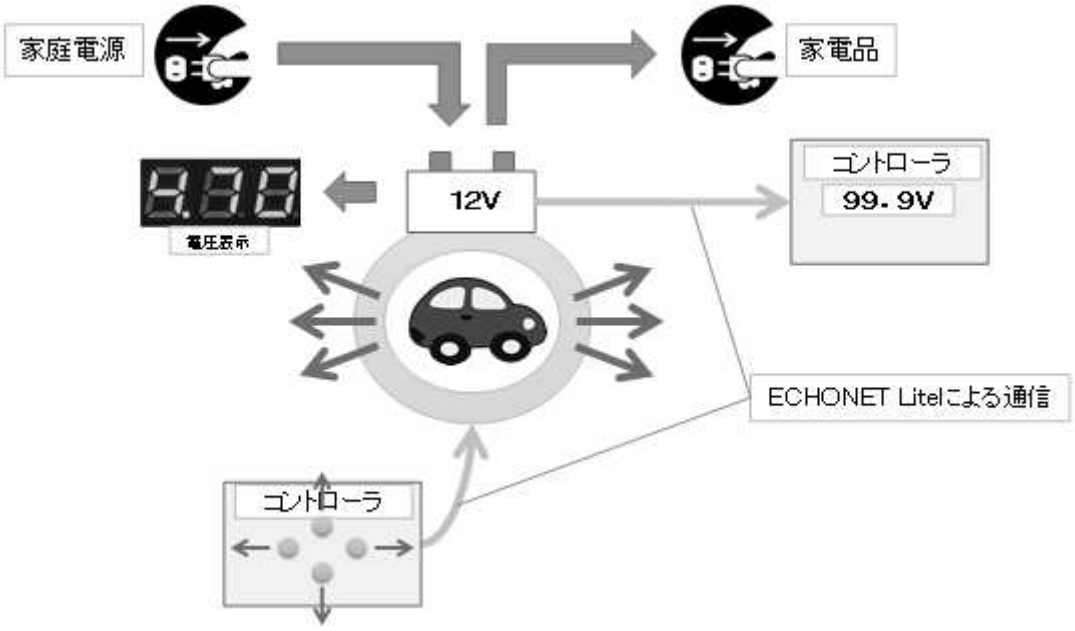
ているので、視点を変え言葉も変えると「法律で私たちの生活がを守られている」ということなのです。

７章４節　モデル V2H の開発

次はいよいよモデル V2H の開発手順を解説します。

このシステムでは、家庭電源（AC100V）から自動車の 12V バッテリーに充電を行います。充電した電力を利用して自動車の駆動系を動かします。自動車には ECHONET Lite に対応したマイコンシステムを搭載して、コントローラと通信します。駆動系はコントローラの指示に従い自動車を動かします。バッテリーは、インバータ経由で家電品を動かします。これが V2H のモデルです。バッテリーからインバータへの電源供給は、コントローラからの指示に従います。バッテリーの電圧は、自動車の電圧計で表示をしますが、ECHONET Lite を経由して別のコントローラでもモニタできるようにしてみましょう。

これらを、動き回る自動車に対して作り込むためには、無線の環境が必要となります。自動車に搭載するシステムの Ethernet 系は無線化してルーターに接続することにしましょう。開発するシステムの概要全体像を図に示します（図 7-4）。



（図 7-4：モデル V2H システム）

７．４．１．自動車

ここで使用する自動車は、蓄電用バッテリーや充電器、開発するシステム、イン

バータを載せられるようなものが必要です。外部から制御可能（操縦可能）なものとして、子供用の乗用可能なリモコン自動車で最大荷重 20Kg のものがあります。駆動系の電源は 12V のバッテリーを搭載していればインバータで容易に AC100V を作り出せるのですが、国内の玩具では 6V バッテリー搭載のものとなります。

そこで、バッテリーを 12V のものに交換して、DC-DC コンバータで変圧して 6V を作り出し、駆動系、制御系の電源を作ります。バッテリーとその他のシステム全体を搭載しても、数 Kg ですので、子供用自動車の最大荷重 20Kg には、十分な余裕があります。

（図 7-4-1：V2H システムモデルカー）

７．４．２．充電器

バッテリーは AC100V で充電しますので、充電器を自動車に搭載することになります。基本的な充電機能だけを持つ小型のもので、重さは 120g 程度です。

（図 7-4-2：12V 用小型充電器）

７．４．３．バッテリー

12V-7.2Ah のバッテリーを使用します。V2H の実験が目的なので、もっと小さい容量のバッテリーでもかまいません。

（図 7-4-3：12V バッテリー）

７．４．４．V2H を実現するインバータ

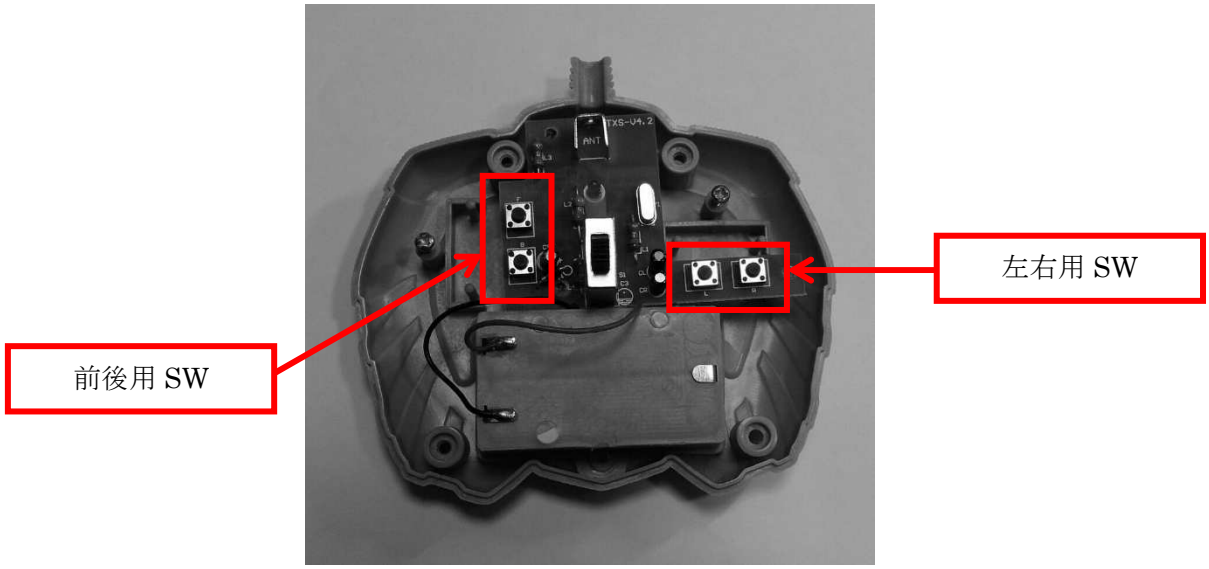
12V バッテリーがあれば、自動車用インバータを利用して AC100V を作ることができます。これが V2H の機能となります。このことで、災害時には充電されたバッテリーとインバータがあれば、家電を利用できることが分かります。

これは簡単なことですが、大変重要なことです。インバータの ON/OFF は、ECHONET Lite の電文で制御します。

（図 7-4-4：インバータ）

7．4．5．リモートコントローラ

リモートコントローラは自動車に付属しているものをそのまま利用することにしますが、ECHONET Lite 経由でも操縦できるようにしたいと考えているので、自動車に搭載するマイコンとリモートコントローラを接続して、コントローラのレバー操作をマイコンでも行えるようにします。マイコンは、もちろん ECHONET Lite の電文で処理を行います。



（図 7-4-5：コントローラ内部）

リモートコントローラの裏蓋を開けると、前後・左右の操作をおこなうレバーが ON/OFF する SW が見えます。（図 7-4-5）の左側に上下にあるのが前後、右側に水平にあるのが左右の操作SW です。この SW の接点をマイコンと接続します。接続は、マイコンのデジタル出力で ON/OFF する FET の回路を作り、そのドレインーソース間に SW の接点を接続すれば、マイコンのプログラムで操作レバーの動きをエミュレーションできます。中央のスライド SW は、電源用です。このリモートコントローラを自動車に搭載したまま、ECHONET Lite 電文を送ることで自動車が遠隔制御できるというわけです。

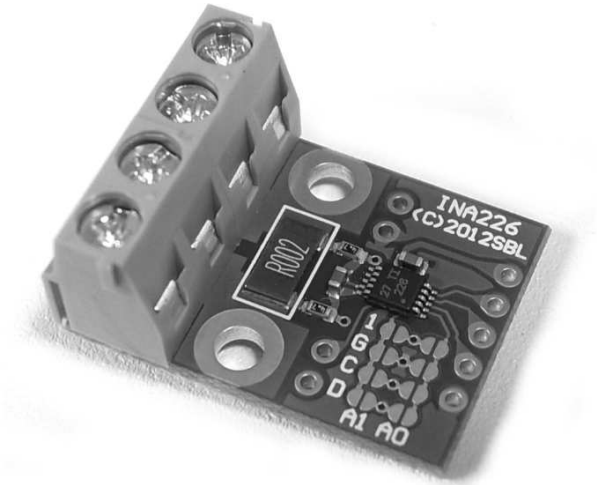
7．4．6．電圧計

バッテリーの電圧が分かる様に電圧計を取り付けます。単純に端子間の電圧を測定して表示します。

（図 7-4-6：電圧計）

７．４．７．電圧・電流センサ

バッテリーの電圧は、別途センサでも計測して、専用コントローラに電文送信することで、外部でも電圧をモニタできるようにします。このセンサは電流も計測できます。マイコンとの接続は I2C インターフェースで利用できます。このセンサはテキサス・シンスツルメント社製の INA226 というチップが使われていて、チップ名で WEB 検索するとデータシートがヒットします。詳細は、その内容を参照してください。



（図 7-4-7：電圧・電流センサ）

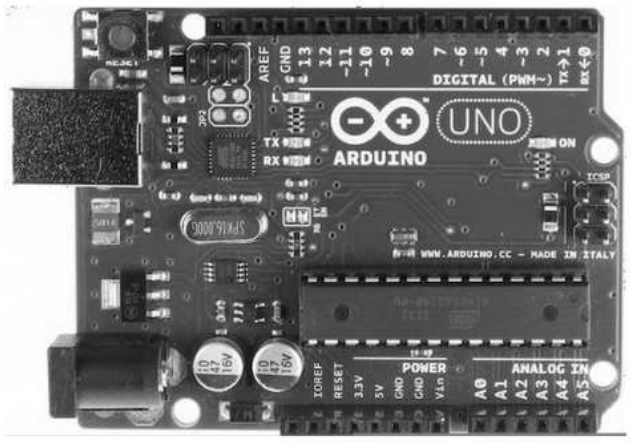
７．４．８．液晶表示器（LCD）

電圧・電流センサの計測値をコントローラ側で表示するのに使用します。マイコンとの接続は I2C インターフェースを利用した接続で、2 本の信号（SCL,SDA）で表示を行います。具体的な使い方は、実際に出来上がったモデル V2H カーのコントローラのプログラムを参照してください。いとも簡単に表示ができます。

（図 7-4-8：液晶表示器）

７．４．９．マイコン（Arduino UNO）

マイコンには、3 章で解説した、Arduino UNO を使います。（図 7-4-9）



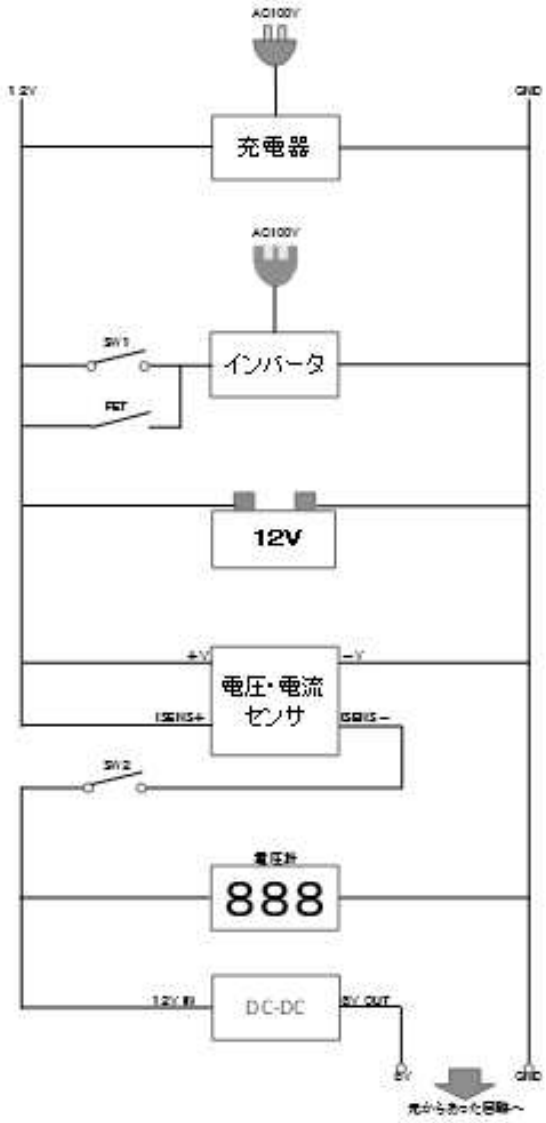
（図 7-4-9：Arduino UNO）

７．４．１０．その他

他には、充電の際に電源回路を切り離すスイッチや、インバータ回路を強制的に ON/OFF するスイッチ。マイコンからインバータの ON/OFF 制御をするための FET、逆接続したときのガード用ダイオード、それらをまとめる基板や配線材料などを使います。

７．４．１１．基本的な考え方と追加する回路

今回利用するリモコン自動車は、そもそもが 6V 駆動のものなので、電気系統はそのまま使います。12V のバッテリーから 6V を作り出して、元の電源ラインに接続するという単純な方法で開発を進めます。こうしておけば、すぐに元に戻せますし、リモコンの機能はそのまま残せます。本当に開発しなければいけないのは、自動車制御用マイコンのプログラムとコントローラマイコンのプログラムです。これらは、それぞれ 5 章のモデルスマート家電と 6 章のモデルコントローラのプログラムをベースに開発することができるでしょう。（図 7-4-11）に電源ブロックを示します。



（図 7-4-11：電源ブロック）

（図 7-4-11）で、SW1 は、強制的にインバータを ON/OFF するスイッチで、ON すると AC100V が得られます。SW2 がメインの電源スイッチとなります。SW1 と並列にある FET は、マイコンからの制御信号で ON/OFF する FET のドレインとソースに接続されます。最終的に DC-DC コンバータで作られる 6V の電源は、もとの駆動回路に入力されて、自動車全体を動かします。

このほか、利用するマイコンの電源にも 12V を接続し、マイコンで作られる 5V を電圧・電流センサの電源とします。

７．４．１２．自動車側マイコンの信号割り付け

リモコン自動車側のマイコンは、Ethernet に接続し ECHONET Lite 電文を送受

信しながら、自動車のリモコンに接続されているデジタル出力（4bit）とインバータ制御用デジタル出力（1bit）を ON/OFF して全体の制御を行います。また、空き時間に電圧と電流を計測して、コントローラに送信します。

これらすべてを Arduino の信号に割り当てると次のようになります。

ピン番号	割当
A0	未使用
A1	未使用
A2	未使用
A3	未使用
A4	I2C：SDA－電圧・電流センサ
A5	I2C：SCL－電圧・電流センサ
D0	Rx：PC とのシリアル通信
D1	Tx：PC とのシリアル通信
D2	未使用
D3	未使用
D4	SD カード用スレーブセレクト（Ethernet カードで接続されている）
D5	リモートコントローラ制御用 FET 出力（前進）
D6	リモートコントローラ制御用 FET 出力（後退）
D7	リモートコントローラ制御用 FET 出力（右）
D8	リモートコントローラ制御用 FET 出力（左）
D9	インバータ制御用 FET 出力
D10	SPI：Ethernet 用
D11	SPI：Ethernet 用
D12	SPI：Ethernet 用
D13	SPI：Ethernet 用

（図 7-4-12：自動車マイコン信号割り当て）

７．４．１３．コントローラ側マイコンの信号割り付け

コントローラ側のマイコンは、Ethernet に接続し ECHONET Lite 電文を送受信しながら、SW 入力を検出し電文送信すること。送られてきた電圧・電流を液晶表示器に表示すること。この２つの仕事を行います。液晶表示器は、I2C インターフェースで接続ができますので、自動車側のマイコンと同様に信号割り当てを行うと次のようになります。

ピン番号	割当
A0	未使用
A1	未使用
A2	未使用
A3	未使用
A4	I2C：SDA－LCD（液晶表示器）
A5	I2C：SCL－LCD（液晶表示器）
D0	Rx：PC とのシリアル通信
D1	Tx：PC とのシリアル通信
D2	未使用
D3	未使用
D4	SD カード用スレーブセレクト（Ethernet カードで接続されている）
D5	SW 入力（前進）
D6	SW 入力（後退）
D7	SW 入力（右）
D8	SW 入力（左）
D9	SW 入力（インバータ制御用）
D10	SPI：Ethernet 用
D11	SPI：Ethernet 用
D12	SPI：Ethernet 用
D13	SPI：Ethernet 用

（図 7-4-13：コントローラ信号割り当て）

７．４．１４．実際の製作について

実際にこのシステムを製作する場合、コントローラ側はブレッドボードを利用して作ることができますので、実習するのに最適です。その場合、SW は 10～20kΩ程度の抵抗でプルダウンしておきます。(プルダウンとは、信号がふらつかないように、GND レベルに固定することです。)

しかし自動車に搭載するマイコンに接続する回路の場合は、振動・衝撃等の影響を考えて、基板に半田付けした製作が必要です。特に（図 7-4-11）で示した電源側回路と周辺の回路は、ショートなどが起こらないように、しっかりとした配線を行う必要があります。

７．４．１５．電文設計

このシステムを ECHONET Lite に接続するために、規格に従い設計する電文は次のようなものになるでしょう。【課題】です。「空欄に不足と思われる、あるいはあった方が良い機能を実現するための電文を自由に記入してみてください。」そもそも、スマート家電として自動車を自動操縦することは考えられないのですが、基本的なことは意外に少ないものです。未来の社会では、このテキストで学ぶことが役立つことになります。

発信元	送信先	内容
コントローラ	自動車	接続ノード問い合わせ
自動車	コントローラ	起動しました
自動車	コントローラ	電圧・電流
コントローラ	自動車	停止・前進・後退・右・左

（図 7-4-15：ECHONET Lite 通信電文）

さて、必要となる電文が洗い出せるでしょうか。ヒントを少し与えるとしたら、例えば、右・左などをどのように電文にするか？悩みますよね。それは簡単です。右・左はそれを示すライト（ウインカー）のようなものだと考えれば、簡単でしょう。現実にはあり得ませんが、前進する照明をつけると、照明が灯ると同時に自動車が前に進むと考えてはどうでしょうか。同じく、後退を示すライトが灯るとバックする。と考えるのです。すると、前進・後退・右・左、全て普通の照明を制御する電文と同じになります。そして、一般照明の例では 1 つしか照明がありませんが、複数の電球を持つ照明だと思えば、1 番の電球は前進で、2 番は後退で・・・というように、ECHONET Lite で規格化された電文を自由に使うことができるようになります。

実際のスマート家電では、そうもいかない部分もあるのですが、そのような考え方をすれば、設計が進むということです。ここでは、いかに設計するかが命題ですから、それでよいのです。

また、2 章で説明しているように、ECHONET Lite 電文の EHD2（ヘッダー2）を 0x82 とすると、電文中の EDATA 部を任意フォーマットとして使うことができる

ので、ECHONET 機器オブジェクト詳細規定では制御しにくい機器を開発するばあいは、これを利用すればよいでしょう。任意フォーマットの電文を設計することで、どのようにも利用できる、柔軟性のある規格となっていることが、改めて理解できると思います。

5 章、6 章を読み込んだ皆さんには、既定の電文設計が容易に行える状態になっています。また、任意フォーマット電文も利用できます。

ぜひ、（図 7-4-15）の空欄を埋めて、実際の電文設計を行い、5、6 章のプログラムを改造して、モデル V2H とコントローラを開発してください。

今回、コントローラでは液晶表示器、自動車側では電圧・電流センサが具体的な解説もなしに出てきていますが、実際に開発されているモデル V2H を見ながら、使い方やソフトウエアの記述をまねて、新たに独自のシステムを開発してみてください。

－終わり－

※文書の引用について

本章の内容は、次の各 HP からの引用があります。

記載された内容について下記の引用元が保証するものではありません。また、書かれている内容に不具合が出た場合でも JMAAB では一切の責任を負いかねます。

【引用元】

経済産業省

エコーネット・コンソーシアム

HEMS 認証支援センター

日経コンピュータ

Arduino-Home

Processing.org

オーム社

次世代自動車振興センター

DENSO

三菱自動車

平成 27 年度文部科学省委託 「東日本大震災からの復興を担う専門人材育成支援事業」
東北の復興・再生を担う自動車組込みエンジニア育成支援プロジェクト

次世代組込み技術（応用）教材

平成 28 年 2 月

東北の復興･再生を担う自動車組込みエンジニア育成支援プロジェクト推進協議会

学校法人日本コンピュータ学園（東北電子専門学校）
〒 980-0013 宮城県仙台市青葉区花京院一丁目3番1号

●本書の内容を無断で転記、掲載することは禁じます。