

AJAN

ネットワークコマンド  
リファレンス

## 目 次

---

<b>第 1 章</b>	<b>はじめに</b>	<b>3</b>
<b>第 2 章</b>	<b>機能説明</b>	<b>4</b>
<hr/>		
2.1	ネットワークコマンドについて.....	4
2.2	TCP によるデータ送受信 .....	6
2.3	UDP によるデータ送受信.....	8
2.4	UDP によるブロードキャスト送信.....	9
2.5	UDP によるマルチキャスト送信.....	11
2.6	MQTT によるデータ送受信 .....	12
2.7	Tips .....	13
<b>第 3 章</b>	<b>リファレンス</b>	<b>17</b>
<hr/>		
3.1	コマンド一覧.....	17
3.2	コマンド個別説明(TCP/UDP 通信) .....	18
3.3	コマンド個別説明(MQTT 通信).....	29
<b>第 4 章</b>	<b>サンプルプログラム</b>	<b>33</b>
<hr/>		
4.1	サンプルプログラム一覧.....	33
<b>第 5 章</b>	<b>索引</b>	<b>35</b>
<b>第 6 章</b>	<b>重要な情報</b>	<b>36</b>

---

## 第1章 はじめに

本ドキュメントは、AJANのネットワークコマンドの説明を記載しております。

本ドキュメントでは、説明で表現している表記として下記のように定義します。

- ・ コマンドの書式の説明において、[]内の引数は省略できます。
- ・ 文字の大小について  
コマンドは大文字/小文字のどちらでも動作します。  
変数名は大文字/小文字も同じものとして扱われます。  
ファイルパス/ファイル名は大文字/小文字で区別されます。



本ドキュメント記載の、AJANはIoT用プログラミング言語です。  
Interface Linux System上でのみ動作可能です。

## 第2章 機能説明

### 2.1 ネットワークコマンドについて

ネットワークコマンドは、TCP/IPや各種ネットワークプロトコルを扱うコマンド群です。

ここでは、ネットワークコマンドを扱う上で、必要事項について述べます。

TCP/IP および各種用語のより詳細な解説については、専門の書籍を参照ください。

ネットワークコマンドを扱う上で、出てくる主要な用語として、以下が挙げられます。

用語	解説
IPアドレスとポート番号	<p>IPアドレスはネットワーク通信を行う機器を識別する為の一意の数値です。パソコン等に搭載されているLANは、一般的に一つのLANポートに1つのIPアドレスが割り当てられます（ソフト的に複数のIPアドレスを割り当てる事も可能）。</p> <p>「192.168.1.1」や「127.0.0.1」のような表記のIPv4と、「fe80::280:62ff:fe82:b66f」のような表記のIPv6が 使用されます。</p> <p>ポート番号は、ネットワーク通信を行う際の識別用の番号です。IPアドレスとプロトコル毎に、65536個の番号を持ちます。</p> <p>例えるなら、IPアドレスは住所で区別できる1軒1軒のマンションで、ポート番号はマンションの各部屋番号のようなものです。</p>
サーバとクライアント	<p>サーバは、データ通信を行うサービスを提供するものであり、クライアントからの要求を受けてサービスを返す事が肝要です。</p> <p>クライアントは、サーバに対して接続して、サービスを要求します。</p> <p>ネットワークコマンドのNWOPEN命令は、オープンする際、必ず サーバとクライアントを指定して通信を行います。</p>
プロトコル	<p>プロトコルは、データ通信を行う際の規約を指します。</p> <p>例えるなら、会話を行う際に、双方が理解できる言語(日本語で会話など)を選択する事に相当します。</p>

ネットワークコマンドは、用途別にプロトコルを選択して使用できるようになっています。

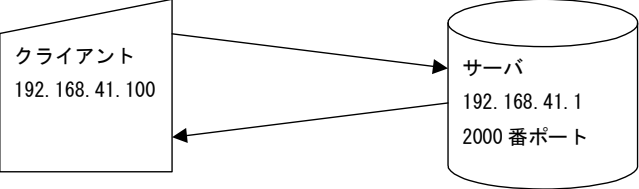
プロトコル	特徴
TCPプロトコル	<ul style="list-style-type: none"><li>通信相手先に、重複したり喪失する事なく、データが配送されます。通信データの信頼性が求められる場合は、こちらのプロトコルを使用します。</li><li>通信データは、ストリームと呼ばれる区切り無しのデータが送られます。(紙テープのような情報を流しているようなイメージ)</li><li>通信は、1対1(ユニキャスト)のみで、同時に双方向可能です。</li><li>片端がサーバ、もう片端がクライアントとなります。サーバが切断すると通信エラーとなり、双方が再度オープンする必要があります。(通信状態の把握は、内部でOSがハンドシェイクする事で行われています)</li></ul>
UDPプロトコル	<ul style="list-style-type: none"><li>通信相手先に、データを配送しますが、順序間違いや喪失の可能性があります。その代わり、TCPプロトコルより一般的には速いとされています。動画の視聴(ストリーミング配信)のように一部コマ落ちしても問題なく、速度が優先されるようなケースでは、こちらのプロトコルが向いています。</li><li>通信データは、データグラムと呼ばれる ひと塊のデータ(一般にパケットと呼ばれる)を単位に送られます。(手紙のような大きさの決まった情報を飛ばすようなイメージ)</li><li>通信は、1対1、ブロードキャスト、マルチキャストが選択可能です。ブロードキャスト、マルチキャストは、片方向のみです。</li><li>片端がサーバ、もう片端がクライアントとなります。サーバが切断すると通信エラーとなりますが、サーバを再起動すると再び受信可</li></ul>

プロトコル	特徴
	能です。
MQTTプロトコル	<ul style="list-style-type: none"><li>• TCP/IPを元にした、より上位のプロトコルで、IoTやM2Mでの利用に向いています。</li><li>• ブローカーと呼ばれるサーバを介して、あるホストがメッセージを送信(出版：Publishと呼びます)、別のホストがメッセージを受信(購読：Subscribeと呼びます)します。</li><li>• 通信データは、ひと塊のデータを、トピックと呼ばれる 一種のキーに対して、送信または受信します。</li><li>• 通信は、1対1、1対n、n対mが可能です。</li><li>• 送信者と受信者は、ある種クライアントと捉える事ができます。ブローカー(サーバ)がある限り、クライアントは自由に切断と再開が出来ます。</li></ul>

## 2.2 TCPによるデータ送受信

TCPプロトコルによるデータ送受信は、サーバ側とクライアント側で処理が一部異なります。  
データの送受信は、NWSENDとNWRCV\$をしますが、サーバ側は複数のクライアントからの接続に対応する為に、NWACCEPTでクライアントからの接続を受け取ります。

この事例では、仮にサーバのIPアドレス：192.168.41.1、クライアントのIPアドレス：192.168.41.100とします。



サーバは、自身のIPアドレス：192.168.41.1の2000番ポートをオープンして、クライアントからの接続を受け付けるようにし、クライアントは、サーバのIPアドレス：192.168.41.1の2000番ポートに対して接続して、文字列を送受信します。

クライアント側		サーバ側
		指定した IP アドレスとポート番号でオープンする NWOPEN "TCP:192.168.41.1:2000" FOR SERVER AS #1
サーバの IP アドレスとポート番号を指定してオープンする NWOPEN "TCP:192.168.41.1:2000" FOR CLIENT AS #1	<div><div></div><div></div></div>	クライアントからの接続待ちと接続確立。 CNUM% = NWACCEPT (1)
サーバに対して、メッセージを送信する NWSEND #1, "hello world"		
	<div><div></div><div></div></div>	クライアントからのメッセージを受け取る A\$ = NWRCV\$ (CNUM%, 11)
		クライアントに対して、メッセージを返信する NWSEND CNUM%, "HELLO WORLD"
サーバから、メッセージを受信する A\$ = NWRCV\$ (1, 11)	<div><div></div><div></div></div>	
		クライアントを切断 NWCLOSE CNUM%
クローズ NWCLOSE #1		クローズ NWCLOSE #1

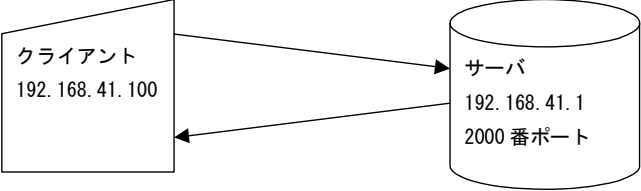
プログラム例を以下にまとめます。

クライアント側	サーバ側
<pre>' オープン NWOPEN "TCP:192.168.41.1:2000" FOR CLIENT AS #1  ' 送信 NWSEND #1, "hello world"  ' 受信 A\$ = NWRECV\$(1, 11)  ' #1 を閉じる NWCLOSE #1</pre>	<pre>' オープン NWOPEN "TCP:192.168.41.1:2000" FOR SERVER AS #1  ' 接続待ち CNUM% = NWACCEPT(1)  ' 受信 A\$ = NWRECV\$(CNUM%, 11)  ' 返信 NWSEND CNUM%, "HELLO WORLD"  ' CNUM%を閉じる NWCLOSE CNUM%  ' #1 を閉じる NWCLOSE #1</pre>

## 2.3 UDPによるデータ送受信

UDPプロトコルによるデータ送受信は、サーバ側とクライアント側で処理が同一です。  
データの送受信は、両方共 **NWSEND**と**NWRECV\$**を使います。

この事例では、仮にサーバのIPアドレス：192.168.41.1、クライアントのIPアドレス：192.168.41.100とします。



サーバは、自身のIPアドレス：192.168.41.1の2000番ポートをオープンして、クライアントからの受信待ちとし、クライアントは、サーバのIPアドレス：192.168.41.1の2000番ポートに対して、接続して文字列を送信受信します。

クライアント側		サーバ側
サーバの IP アドレスとポート番号を指定してオープンする NWOPEN "UDP:192.168.41.1:2000" FOR CLIENT AS #1		指定した IP アドレスとポート番号でオープンする NWOPEN "UDP:192.168.41.1:2000" FOR SERVER AS #1
サーバに対して、メッセージを送信する NWSEND #1, "hello world"		
		クライアントからのメッセージを受け取る A\$ = NWRECV\$(1, 11)
		クライアントに対して、メッセージを返信する NWSEND #1, "HELLO WORLD"
サーバから、メッセージを受信する A\$ = NWRECV\$(1, 11)		
クローズ NWCLOSE #1		クローズ NWCLOSE #1

プログラム例を以下にまとめます。

クライアント側	サーバ側
' オープン NWOPEN "UDP:192.168.41.1:2000" FOR CLIENT AS #1	' オープン NWOPEN "UDP:192.168.41.1:2000" FOR SERVER AS #1
' 送信 NWSEND #1, "hello world"	' 受信 A\$ = NWRECV\$(1, 11)
' 受信 A\$ = NWRECV\$(1, 11)	' 返信 NWSEND #1, "HELLO WORLD"
' #1 を閉じる NWCLOSE #1	' #1 を閉じる NWCLOSE #1

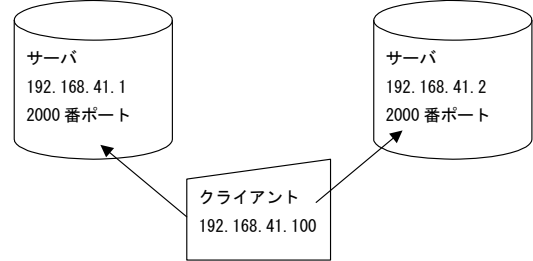


2.4 UDPによるブロードキャスト送信

UDPプロトコルでは、IPアドレスにブロードキャストアドレスを指定する事で、複数の端末に対してブロードキャスト送信が可能です。

!	ブロードキャスト送信は、複数の端末に対して一斉に通信を行う為、ネットワークの帯域全体を大きく消費します。 用途上、ブロードキャストが必要な時以外は、使わない事が推奨されます。
!	一般に売られているルーター(ネットワーク間を中継する装置)は、ブロードキャスト通信を中継しないように初期設定されています。
!	ブロードキャスト送信は、IPv4でのみ使用可能です。

この事例では、仮に、2台のサーバのIPアドレス：192.168.41.1 と 192.168.41.2、クライアントのIPアドレス：192.168.41.100とします。



2つのサーバは、自身のIPアドレスの 2000番ポートをオープンして、クライアントからの受信待ちとし、クライアントは、ブロードキャストアドレス：192.168.41.255 の2000番ポートに対して、接続して文字列を送信します。

サーバとクライアントのプログラム例を以下に示します。  
特に重要な箇所を赤字にしています。

クライアント側	サーバ側
<pre>' オープン NWOPEN      [      "UDP:192.168.41.255:2000"; "SO_BROADCAST" ] FOR CLIENT AS #1  ' 送信 NWSEND #1, "hello world"  ' #1 を閉じる NWCLOSE #1</pre>	<pre>' オープン NWOPEN "UDP:0.0.0.0:2000" FOR SERVER AS #1  ' 受信 A\$ = NWRECV\$(1, 11)  ' #1 を閉じる NWCLOSE #1</pre>

サーバ側は、IPアドレス指定でIPv4時”0.0.0.0”を指定することで、利用可能な全てのネットワークインタフェースでクライアントからデータを受信することができます。それ以外は普通のUDP受信プログラムと変わりません。

- クライアント側で NWOPEN 命令を呼び出すとき、以下を指定する事が重要です。
- ・IPアドレス指定の添字の0番目は、ブロードキャストアドレスを指定する。
  - ・IPアドレス指定の添字の1番目に、“SO\_BROADCAST” オプションを指定する。

ブロードキャストアドレスとは、IPアドレスのホスト部の全ビットを1にしたものとされています。  
上の例では、送信したいIPアドレスが「192.168.41.x」で、サブネットマスクが「255.255.255.0」と仮定すると、以下のように求められます。

IPアドレス	192	168	41	x
サブネットマスク	255	255	255	0
ブロードキャストアドレス	192	168	41	255

なお、IPアドレス全てのビットを1、つまり「255.255.255.255」とした場合、全ての端末に対して送信するので、特にリミテッドブロードキャストアドレスと呼ばれます。

## 2.5 UDPによるマルチキャスト送信

UDPプロトコルでは、IPアドレスにマルチキャストアドレスを指定する事で、複数の端末に対してマルチキャスト送信が可能です。



一般に売られているルーター(ネットワーク間を中継する装置)は、マルチキャスト通信を中継しないように初期設定されています。

マルチキャスト送信をするには、マルチキャストアドレスと呼ばれる特定の範囲のIPアドレスを選択し、受信者がそのアドレスに参加、送信者がアドレスに対して送信する事で、受信者全員に対してメッセージが送信されます。

ブロードキャストの場合は、メッセージを受け取りたく無いもの関係なしに送られますが、マルチキャストの場合は、選択的に送る事が可能です。

マルチキャスト送信に用いるマルチキャストアドレスは、以下のIPアドレス範囲を指します。

IPv4	224. 0. 0. 0 ~ 239. 255. 255. 255
IPv6	ff00::/8 で始まるアドレスブロック

自身のコンピュータで使用可能なマルチキャストアドレスを知りたい場合、Linuxの端末上で「[ip maddress](#)」と入力してください。

以下に、例を示します。

(「inet」がIPv4のマルチキャストアドレス。「inet6」がIPv6のマルチキャストアドレスです)

```
$ ip maddress
... 略
2:      eth0
    link  01:00:5e:00:00:01
    link  33:33:00:00:00:01
    link  33:33:ff:ab:af:4e
    link  01:00:5e:00:00:fb
    link  33:33:00:00:00:fb
    inet  224.0.0.251
    inet  224.0.0.1
    inet6 ff02::fb
    inet6 ff02::1:ffab:af4e
    inet6 ff02::1
    inet6 ff01::1
... 略
```

サーバとクライアントのプログラム例を以下に示します。

特に重要な箇所を赤字にしています。

クライアント側	サーバ側
<pre>' オープン NWOPEN "UDP:224.0.0.251:2000" FOR CLIENT AS #1  ' 送信 NWSEND #1, "hello world"  ' #1 を閉じる NWCLOSE #1</pre>	<pre>' オープン NWOPEN [ "UDP:0.0.0.0:2000"; "IP_MULTICAST=224.0.0.251" ] FOR SERVER AS #1  ' 受信 A\$ = NWRECV\$(1, 10)  ' #1 を閉じる NWCLOSE #1</pre>

クライアント側は、NWOPEN 命令を呼び出すとき、以下を指定する事が重要です。

- IPアドレス指定で、マルチキャストアドレスを指定する。

サーバ側は NWOPEN 命令を呼び出すとき、以下を指定する事が重要です。

- IPアドレス指定の添字の0番目に、IPv4時"0.0.0.0"を指定する。
- IPアドレス指定の添字の1番目に、"IP\_MULTICAST" オプションを指定しつつ、マルチキャストアドレスを指定する。

## 2.6 MQTTによるデータ送受信

MQTT(Message Queuing Telemetry Transportの略)は、データ通信プロトコルの一つです。一般的に、Pub/Sub型データ配信モデルに準拠するといわれます。

Brokerと呼ばれるサーバに対して、データの送信(publish)およびデータの受信(subscribe)機能を持つクライアントとしてのコマンドを持ちます。

AJAN自体には、Broker機能はありません。別途 Broker機能を持つサーバソフトウェアを導入してください。AJANでは、mosquitto と呼ばれるパッケージを使用する事を推奨します。

mosquittoパッケージのインストール事例を、以下に示します。

```
$ sudo apt-get install mosquitto
$ sudo apt-get install mosquitto-clients
```

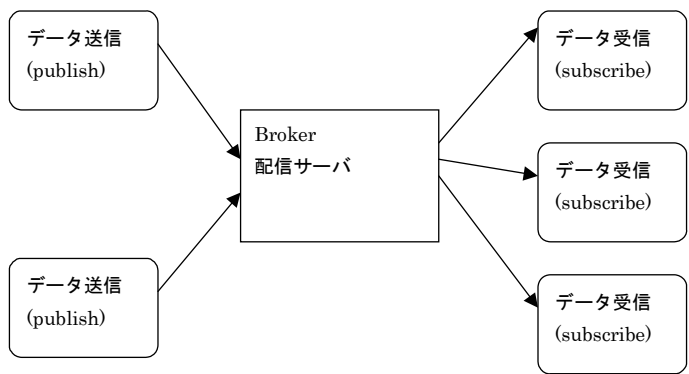
データの送信および受信は、トピックと呼ばれる一種の宛先のようなものに対して行います。トピックは「/」を使った階層構造で名前を指定します。AJANでは、MQOPEN 呼び出し時に、トピックIDで指定します。

データの送信は、MQOPEN で、動作モードに "OUTPUT" を指定してオープンした後、MQSEND で行います。

データの受信は、MQOPEN で、動作モードに "INPUT" を指定してオープンすると、バックグラウンドで、トピックに対して、データが送信される都度、内部キューに取り込まれます。

MQSTAT 関数で、内部キューの件数を知る事ができ、キューに取り込まれた受信データを、MQRECV\$ 関数で取り出す事ができます。

MQTTは、TCPと異なり、1対1通信だけでなく n対mの通信が可能です。受信者なしにデータ送信が可能ですし、複数の受信者が同じデータを受信可能です。



## 2.7 Tips

### 1. IPv6を使った通信を行う際の注意

ネットワークコマンドは、IPv6アドレスを使った通信をサポートしています。  
従来のネットワーク通信で使われていた、IPv4はアドレス表現に32bit幅を使用するのに対して、IPv6は128bit幅を用います。

この為、IPv4とIPv6では、アドレス表現方法に違いがあります。

例えば、Linuxで「ip a」コマンドを使って、自分の端末のアドレスを確認してみます。

以下は、呼び出し事例です。紫文字がIPv4のアドレス。青文字がIPv6のアドレスです。

```
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
   inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
   link/ether 00:80:62:82:b6:6f brd ff:ff:ff:ff:ff:ff
   inet 10.80.40.66/24 brd 10.80.40.255 scope global dynamic enp1s0
       valid_lft 208812sec preferred_lft 208812sec
   inet6 fe80::280:62ff:fe82:b66f/64 scope link
       valid_lft forever preferred_lft forever
```

NWOPEN(P.18)で、IPアドレスとポート番号を指定しますが、幾つか注意事項があります。

<注意1> 「[]括弧」を使ってIPアドレスとポート番号を明示的に指示する。

IPアドレスに、IPv6を指定したい場合、IPアドレスとポート番号の境を明示的に指示する為に、例えば以下のように[]括弧を使って指定してください。

```
' IP アドレスにローカルアドレス、ポート番号:12345 を指定する事例
NWOPEN "TCP:127.0.0.1:12345" FOR SERVER AS #1 ' IPv4 の場合
NWOPEN "TCP:::1]:12345"      FOR SERVER AS #1 ' IPv6 の場合
```

<注意2> リンクローカルのアドレスを使用する場合は、ゾーンインデックスの指定が必要

IPv6のアドレスの先頭部分が「fe80」とあるのは、リンクローカルアドレスと規定されています。

これは、インターネット上の機器と直接やり取りできるグローバルアドレスと異なり、例えばイントラネットのように、特定域内でのみ通信可能なアドレスです。

このアドレスを用いるときは、Linuxの場合、ネットワークインターフェース名をゾーンインデックスに付加して使用する必要があります。

先のアドレス事例で言えば、例えば「fe80::280:62ff:fe82:b66f」を用いたい場合、ネットワークインタフェース名「enp1s0」を付加します。区切り記号は「%」です。

```
$ ip a
...
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
   link/ether 00:80:62:82:b6:6f brd ff:ff:ff:ff:ff:ff
   inet 10.80.40.66/24 brd 10.80.40.255 scope global dynamic enp1s0
       valid_lft 208812sec preferred_lft 208812sec
   inet6 fe80::280:62ff:fe82:b66f/64 scope link
       valid_lft forever preferred_lft forever
```

以下に事例を示します。

```
' IP アドレスにリンクローカルアドレス、ポート番号:12345 を指定する事例
NWOPEN "TCP:10.80.40.66:12345" FOR SERVER AS #1 ' IPv4 の場合
NWOPEN "TCP:[fe80::280:62ff:fe82:b66f%enp1s0]:12345" FOR SERVER AS #1 ' IPv6 の場合
```

## 2.TCP>相手先のクローズを検知するには？

TCPプロトコルで通信を行っている中、相手先がクローズした事を以下の方法で検知できます。

受信時	NWRECV\$ で、受信バイト数に0以外を指定したときに、空文字が返ってくる。
送信時	NWSEND で、エラーが発生し、ERRSUB関数の返り値が 44である。 または、 NWSEND\$ で、返り値のエラーコードに 32が返ってくる。

相手先がクローズした事を確認したら、自分もクローズします。

そうする事で、双方が完全にネットワークの接続がクローズ状態になります。

それまでの間、ネットワークは クローズしようとする中間的な状態となります。専門用語では TIME\_WAIT状態といいます。

基本的には、双方がクローズを呼び出す事が肝要ですが、相手先のクローズを待たずに 強制的に接続を落とした場合、NWCLOSE のオプション値に "SO\_LINGER" を指定して呼び出します。そうすると、強制的にネットワークの接続がクローズ状態になります。

## 3.TCP>受信タイムアウトか、相手先クローズの違いの見分け方

先の説明で、相手先がクローズすると、NWRECV\$ から空文字が返ってくると説明しました。タイムアウト時間を設定したとき、受信タイムアウトが発生したのか、相手先がクローズしたのか、どう見分けるかは、NWSTAT の 状態取得IDに "ERR" を用います。

以下に事例を掲載します。

```

A$ = NWRECV$ (FNUM%, 10, 1)  ' 10 バイトの受信要求かつ 1 秒のタイムアウト待ち
IF A$ = "" THEN
  ST = NWSTAT (FNUM%, "ERR")  ' 受信時エラー値を得る
  IF ST = 0 THEN
    PRINT "相手先がクローズしました"
  ELSE
    PRINT "受信タイムアウトが発生しました"
  END IF
END IF

```

#### 4.TCP>可変長のデータを送受信したい時の注意事項

TCPプロトコルは、ストリームと呼ばれる 紙テープのように途切れの無いデータを扱うプロトコル形式です。

この為、基本的には、送信側と受信側で、どのような形式のデータを送受信するか、決めておく事が肝要です。

例えば、受信側が 10バイト受け取る命令を発行しているのに、送信側が 5バイトしか送っていないと、受信側は残り5バイトが送られるまで延々と待ち続ける事になります。(タイムアウトを指定していると、エラーとなって中断できる)

すなわち、TCPプロトコルで通信を行うには、**送信者と受信者の間で、アプリケーション独自のプロトコル(通信規約)** を取り決める必要があります。



他のプロトコル、例えば UDPプロトコルは、データサイズに影響されず 送信と受信の呼び出し回数と一致するので、比較的単純に扱えます。

固定長でデータを送受信するならば、話は単純です。送信側が xバイト送るなら、受信側も xバイト受信すれば良いだけです。(xバイト送受信するプロトコルを決めた事になります)

可変長でデータを送受信する場合、既存の通信プロトコル、例えば Telnet、HTTP、SMTP などの事例が参考になります。

既存の通信プロトコルで多いのは、テキストベースで送受信を行うものです。1バイトずつ送受信したり、改行単位で送受信するケースが多いです。

AJANで改行単位にデータを送受信したいならば、送信側は改行コード(CHR\$(10))を付加して送り、受信側は 受信バイト数の指定で、-1 を指定します。

送信側	受信側
' 改行コードを付けて送信 NWSEND #1, "Hello world"+CHR\$(10)	' 改行コードが来るまで受信待機 A\$ = NWRECV\$(1, -1)

テキストベースでない方式となると、一般的には 固定長のヘッダ部分と可変長のデータ部分を送受信する、というものです。

ヘッダ部分で、データ部分のバイトサイズを教える事で、受信側は 一旦ヘッダ部分を読み取った後、残りのデータ部分に必要なデータサイズを知る事が可能となります。

以下は、4バイトのヘッダに、データ本体のサイズ情報を教える。という単純な事例です。

送信側	受信側
' 送信したい文字列 A\$ = "Hello world" ' 送信したい文字列のサイズを求める SZ = LEN(A\$) ' サイズ値を、4 バイトのバイナリ文字列へ H\$ = MKI\$(SZ)    ' これがヘッダ ' ヘッダとデータ部分を合体して送信 NWSEND #1, H\$ + A\$	' ヘッダ部分(4 バイト)を受信 H\$ = NWRECV\$(1, 4) ' ヘッダからデータサイズ値を取り出す SZ = CVI(H\$) ' 残りのデータ部分を受信する A\$ = NWRECV\$(1, SZ)

## 5.UDP>送信するサイズの注意事項

UDPプロトコルでデータを送信する場合、一度に送れる理論上の限界とは別に、送受信する端末間のルーター(中継器)の仕様やOSなど各種要素に注意する必要があります。

まず、一度に送れる理論上の限界は、IPv4で通信する場合は 65507 バイトまでです。

これは、UDPの理論上の限界が 65535バイトで、そのうち データを送受信する際に自動的にシステムが付加するヘッダ情報(IPヘッダ：20バイト、UDPヘッダ：8バイト)を除いた値となります。

しかし、実際にデータを送信する際、MTUと呼ばれる通信回線を1回に送れる最大送信サイズを考慮する必要があります。

LAN すなわち イーサネットでは、MTUは 1500バイトが一般に使われます。

先の例で言うと、MTUのサイズを越えるデータを送ろうとした時、実際には送信データがMTUのサイズに分割して送られます。これを一般に断片化と言います。

UDPで断片化を起こすようなサイズのデータを送信する事は、一般には避けるべき事項とされています。理由としては幾つかあり、送信側・受信側ともに負荷が高まる事、1つの断片化されたデータを失うと全てのデータを失う事、ルーターやOSの設定によって断片化されたデータを中継・送信できない場合がある事、などが挙げられます。

まとめると、MTUを越えないサイズのデータを送れば、概ね良いとなります。


(IPv4使用して MTUが1500バイトの場合、 $1500 - 20(\text{IPヘッダ分}) - 8(\text{UDPヘッダ分}) = 1472$  バイト)



## 第3章 リファレンス

使用できるコマンドの使い方について記載します。

### 3.1 コマンド一覧

	本コマンドは、使用状況によって、管理者権限(スーパーユーザー)で実行する必要があります。
---	--

コマンド名	機能
TCP/UDP通信に関する関数・命令	
NWOPEN	指定したIPアドレスおよびポート番号で、オープンします。
NWCLOSE	ネットワークをクローズします。
NWACCEPT	サーバ動作時、クライアントからのネットワーク接続を受け付けます。
NWFREEFILE	使用可能なネットワークのネットワーク番号を得ます。
NWSEND	データを送信します。
NWSENDADR	UDP通信時、相手先を指定してデータを送信します。
NWSENDNR	データを送信します。(NWSENDの関数[戻り値あり]版)
NWSENDRADR	UDP通信時、相手先を指定してデータを送信します。 (NWSENDADRの関数[戻り値あり]版)
NWRECV\$	ネットワークに対して、指定したバイト数のデータを受信します。
NWRECVADR\$	ネットワークに対して、指定したバイト数のデータおよび相手先のIPアドレスを受信します。
NWSTAT	指定したネットワークの状態を得ます。
NWIPADR\$	自身のネットワークが認識しているIPアドレス情報を、文字列配列で得ます。
MQTT通信に関する関数・命令	
MQOPEN	MQTT通信を行うためにオープンします。
MQCLOSE	MQTT通信をクローズします。
MQSEND	MQTT通信に対してデータを送信します。
MQRECV\$	MQTT通信に対して受信データを受け取ります。
MQSTAT	MQTT通信に対して受信データの取り出し可能件数を得ます。

## 3.2 コマンド個別説明(TCP/UDP通信)

### 1.NWOPEN

命令			
機能	指定したIPアドレスおよびポート番号で、オープンします。		
書式	NWOPEN <①IPアドレスとポート番号> FOR <②動作モード> AS [#]<③ネットワーク番号>		
パラメータ	①	<IPアドレスとポート番号>	文字列
	文字列形式で「[プロトコル:]IPアドレス：ポート番号」の形式で記述します。 プロトコルは、「UDP」でUDP通信。「TCP」または省略で TCP通信に対応します。 IPアドレスとポート番号は、 サーバ時、クライアントから接続してもらう際のIPアドレスとポート番号を指定します。 クライアント時、接続したいサーバへのIPアドレスとポート番号を指定します。 ポート番号を省略すると、0番を指定するのと同じ効果です。  <IPアドレスとポート番号>を文字列配列形式にすることで、オプションが設定できます。  書式：NWOPEN [<添字0>;<添字1>;<添字2>;...] FOR ~ 添字0には、前述の<IPアドレスとポート番号>を指定します。 添字1には、以下のオプションが設定できます。複数のオプションを指定する場合、添字2,3と追加してください。		
	オプション	説明	
	"SO_KEEPALIVE"	TCPのサーバ接続時、長時間通信を行わないときに一定時間が経過したら、通信可能な状態かOSが応答確認を行い、応答がない場合は切断します。(一定時間は、デフォルトでは約2時間です)	
	"TCP_KEEPIDLE=数値"	"SO_KEEPALIVE"指定時、キープアライブパケットを最初に投げ始めるまでの時間を指定します。 数値の所を、秒単位で指定してください。	
	"TCP_KEEPINTVL=数値"	"SO_KEEPALIVE"指定時、キープアライブパケットを投げる時間間隔を指定します。 数値の所を、秒単位で指定してください。	
	"TCP_KEEPCNT=数値"	"SO_KEEPALIVE"指定時、キープアライブパケットを投げる際、リトライする回数を指定します。ここで指定した回数を投げて失敗するならば、通信を切断します。 数値の所を、回数単位で指定してください。	
	"SO_REUSEADDR"	TCPのサーバ接続時、socketに対してSO_REUSEADDRがデフォルトで有効になっており、ポート番号を使用中でも、OSに対して再利用するよう指示を出します。 本オプションを指定することでSO_REUSEADDRを無効にします。	
	"TCP_NODELAY"	TCP接続時、送信データをバッファに溜めてから送信せず、即送るように OSに対して指示します。 全体的なパフォーマンスは落ちるので、即時性を求めない限り、お勧めしません。	
	"SO_BROADCAST"	UDP接続時、ブロードキャスト通信を行いたい時、指定します。 この時、IPアドレスはブロードキャストアドレスを指定してください。	
	"IP_MULTICAST=アドレス"	UDP接続時、マルチキャスト通信を行いたい時、指定します。 アドレスの所に、マルチキャストアドレスを指定してください。	

	<div> <div> "SO_BINDTODEVICE=インタフェース名" </div> <div> オープン時、指定したインタフェース名(eth1など)のデバイスにバインドします。  指定可能なインタフェース名は、Linuxコマンドで「ip link」を指定した際の、「eth0」や「eth1」などの名前です。  本オプションの呼び出しには、管理者権限(スーパーユーザ)が必要です。 </div> </div>	
	<div> <div>②</div> <div>&lt;動作モード&gt;</div> <div>キーワード</div> </div> <p>サーバとして動作させたい場合は「SERVER」、クライアントとして動作させたい場合は「CLIENT」を指定してください。</p>	
	<div> <div>③</div> <div>&lt;ネットワーク番号&gt;</div> <div>数値</div> </div> <p>作成したネットワークインタフェースに関連付けるネットワーク番号を1～15の範囲で指定します。ネットワークインタフェースの制御はこの&lt;ネットワーク番号&gt;を指定して行います。複数のネットワークインタフェースを同時に利用する場合、NWFREEFILEで使用可能なネットワーク番号を取得することができます。</p>	
備 考	<ul style="list-style-type: none"> <li>サーバ動作時、ネットワークのソケットを作成し、使用するIPアドレスとポート番号を bindします。</li> <li>クライアント動作時、ネットワークのソケットを作成し、サーバに connect します。</li> <li>IPアドレスに、IPv6を指定したい場合、「1.IPv6を使った通信を行う際の注意(P.13)」を参考にしてください。IPv4の場合と記述方法が異なります。</li> <li>サーバ動作時、自分の端末以外に接続を受けるつもりがない場合は、IPアドレスの記名に、ローカルループバックアドレス(IPv4時、"127.0.0.1")を指定します。</li> <li>サーバ動作時、どの別端末からでも接続を受けたい場合、IPアドレスの記名に、デフォルトルートアドレス(IPv4時 "0.0.0.0")を指定する事が可能です。これは、「利用可能な全てのネットワークインタフェースで、クライアントからの接続を受け付ける」の意となります。</li> <li>クライアント動作時、サーバとの接続動作は、おおよそ以下です。  TCP通信時、サーバとの通信が確立してから戻ります。  UDP通信時、サーバとの通信の確立を待たずに、戻ります。 </li> </ul>	
注 意	本コマンドは、サーバ動作時、ポート番号に1023以下を指定して使用する場合、管理者権限(スーパーユーザ)で実行する必要があります。	
使用例	『2.2 TCPによるデータ送受信』および『2.3 UDPによるデータ送受信』を参照してください。	

## 2.NWCLOSE

命令			
機能	ネットワークをクローズします。		
書式	NWCLOSE #<①ネットワーク番号またはクライアント番号> [, <②オプション>]		
パラメータ	①	<ネットワーク番号またはクライアント番号>	数値
	NWOPEN命令でオープンしたネットワーク番号を指定します。 または、NWACCEPT命令で受け取ったクライアント番号を指定します。		
パラメータ	②	<オプション>	文字列
	クローズ時の挙動を変更できます。		
	オプション	挙動	
	"SO_LINGER"	特に TCP 接続時、相手先とのクローズ連携処理を行わず、接続を中断するように OS に指示します。 一般には、あまり指定しない方が良いと言われています。	
備考	・		
使用例	『2.2 TCPによるデータ送受信』および『2.3 UDPによるデータ送受信』を参照してください。		

## 3.NWACCEPT

関数			
機能	サーバ動作時、クライアントからのネットワーク接続を受け付けます。		
書式	<(戻り値)クライアント番号>=NWACCEPT(<①ネットワーク番号> [, <②タイムアウト時間>])		
戻り値	戻り値	<クライアント番号>	数値
	クライアントからのネットワーク接続に対して、割り振られた番号を返します。 クライアントとの通信は、この番号に対して、送受信可能となります。 なお、タイムアウト時間を設定して、クライアントから接続が無い場合、-1 が返ります。		
パラメータ	①	<ネットワーク番号>	数値
	NWOPEN命令でオープンしたネットワーク番号を指定します。		
	②	<タイムアウト時間>	数値
	ネットワーク接続待ちを行う最大時間(秒)を指定します。 省略または0を指定すると、受信完了まで待ち続けます。		
備考	本関数は、サーバーのTCP通信時に使用します。UDP通信では使用しません。		
使用例	『2.2 TCPによるデータ送受信』を参照してください。		

## 4.NWFREEFILE

関数			
機 能	使用可能なネットワークのネットワーク番号を得ます。		
書 式	<(戻り値)ネットワーク番号>=NWFREEFILE()		
戻り値	戻り値	<ネットワーク番号>	数値
	NWOPEN で使用可能な、ネットワークのネットワーク番号が得られます。 全て使用中で空きが無い場合、0が得られます。		
使用例	FNUM% = NWFREEFILE() PRINT "使用可能なネットワーク 番号="; FNUM% NWOPEN "TCP:192.168.41.1:2000" FOR CLIENT AS #FNUM%		

## 5.NWSEND

命令			
機 能	ネットワークに対して、データを送信します。		
書 式	NWSEND #<①ネットワーク番号またはクライアント番号>, <②送信データ> [, <③タイムアウト時間>]		
パラ メータ	①	<ネットワーク番号またはクライアント番号>	数値
	NWOPEN命令でオープンしたネットワーク番号を指定します。 または、NWACCEPT命令で受け取ったクライアント番号を指定します。		
	②	<送信データ>	文字列
	データを送信するデータを文字列形式で指定します。		
	③	<タイムアウト時間>	数値
	送信待ちを行う最大時間(秒)を指定します。 省略または0を指定すると、送信完了まで待ち続けます。		
備 考	<ul style="list-style-type: none"> <li>UDP通信でサーバ動作時、直近の「NWRECV\$」で受け取ったクライアントからのアドレスに対して、送信を行おうとします。</li> <li>UDP通信では、MTUを越える大きなサイズのデータを送信しないでください。 (参考「5.UDP＞送信するサイズの注意事項」)</li> <li>TCP通信で相手先がクローズすると、エラーが発生し、ERRSUB関数の値が 44となります。</li> </ul>		
使用例	『2.2 TCPによるデータ送受信』および『2.3 UDPによるデータ送受信』を参照してください。		

## 6.NWSENDADR

命令		
機 能	UDP通信時、ネットワークに対して、相手先を指定してデータを送信します。	
書 式	NWSENDADR #<①ネットワーク番号>, <②IPアドレスとポート番号>, <③送信データ> [, <④タイムアウト時間>]	
パラ メータ	①	<ネットワーク番号> 数値
	NWOPEN命令でオープンしたネットワーク番号を指定します。	
	②	<IPアドレスとポート番号> 文字列
	通信相手先を、「IPアドレス：ポート番号」の形式で指定します。	
	③	<送信データ> 文字列
備 考	データを送信するデータを文字列形式で指定します。	
	④	<タイムアウト時間> 数値
	送信待ちを行う最大時間(秒)を指定します。 省略または0を指定すると、送信完了まで待ち続けます。	
使用例	<ul style="list-style-type: none"> <li>・本コマンドは、UDP通信でのみ使用してください。</li> <li>・UDP通信では、MTUを越える大きなサイズのデータを送信しないでください。 (参考「5.UDP＞送信するサイズの注意事項」)</li> </ul>	
	NO = 1 'NWOPENで、オープン済みのネットワーク番号と仮定します NWSENDADR NO, "192.168.1.1:2000", "Hello world" '「Hello world」というメッセージを、「192.168.1.1:2000」の宛先に向けて送信します	

## 7.NWSENDNR

関数			
機能	ネットワークに対して、データを送信します。(NWSENDの関数[戻り値あり]版)		
書式	<(戻り値)エラー値> = NWSENDNR( <①ネットワーク番号またはクライアント番号>, <②送信データ> [, <③タイムアウト時間> ] )		
戻り値	戻り値	<エラー値>	数値
	送信に成功したら 0が返り、失敗したら 0以外が返ってきます。 エラー値の代表例を、以下に示します。		
	エラー値	意味	
	0	送信成功	
	4	送信前に、シグナルが発生した。	
	11	要求された操作が停止された。	
	32	接続が閉じられている。	
	41	要求された操作が停止された。	
	104	接続が接続相手によりリセットされた。	
	105	OS の出力キューが一杯である。	
パラメータ	①	<ネットワーク番号またはクライアント番号>	数値
	NWOPEN命令でオープンしたネットワーク番号を指定します。 または、NWACCEPT命令で受け取ったクライアント番号を指定します。		
	②	<送信データ>	文字列
	データを送信するデータを文字列形式で指定します。		
	③	<タイムアウト時間>	数値
備考	送信待ちを行う最大時間(秒)を指定します。 省略または0を指定すると、送信完了まで待ち続けます。		
	<ul style="list-style-type: none"> <li>UDP通信でサーバ動作時、直近の「NWRECV\$」で受け取ったクライアントからのアドレスに対して、送信を行おうとします。</li> <li>UDP通信では、MTUを越える大きなサイズのデータを送信しないでください。 (参考「5.UDP＞送信するサイズの注意事項」)</li> <li>TCP通信で相手先がクローズすると、エラー値は 32 が得られます。</li> </ul>		
使用例	『2.2 TCPによるデータ送受信』および『2.3 UDPによるデータ送受信』を参照してください。		

## 8.NWSENDADR

関数			
機 能	UDP通信時、ネットワークに対して、相手先を指定してデータを送信します。(NWSENDADRの関数[戻り値あり]版)		
書 式	<(戻り値)エラー値>= NWSENDADR( <①ネットワーク番号>, <②IPアドレスとポート番号>, <③送信データ> [, <④タイムアウト時間> ] )		
戻り値	戻り値	<エラー値>	数値
	送信に成功したら 0が返り、失敗したら 0以外が返ってきます。 エラー値の代表例を、以下に示します。		
	エラー値	意味	
	0	送信成功	
	4	送信前に、シグナルが発生した。	
	11	要求された操作が停止された。	
	32	接続が閉じられている。	
	41	要求された操作が停止された。	
パラメータ	①	<ネットワーク番号>	数値
	NWOPEN命令でオープンしたネットワーク番号を指定します。		
	②	<IPアドレスとポート番号>	文字列
	通信相手先を、「IPアドレス：ポート番号」の形式で指定します。		
	③	<送信データ>	文字列
備考	④	<タイムアウト時間>	数値
	送信待ちを行う最大時間(秒)を指定します。 省略または0を指定すると、送信完了まで待ち続けます。		
	<ul style="list-style-type: none"> <li>・本コマンドは、UDP通信でのみ使用してください。</li> <li>・UDP通信では、MTUを越える大きなサイズのデータを送信しないでください。 (参考「5.UDP＞送信するサイズの注意事項」)</li> </ul>		
使用例	NO=1 'NWOPENで、オープン済みのネットワーク番号と仮定します PRINT "送信="; NWSENDADR(NO, "192.168.1.1:2000", "Hello world") ' 「Hello world」というメッセージを、「192.168.1.1:2000」の宛先に向けて送信します		



## 9.NWRECV\$

関数		
機 能	ネットワークに対して、指定したバイト数のデータを受信します。	
書 式	<(戻り値)受信データ>=NWRECV\$( <①ネットワーク番号またはクライアント番号>, <②受信バイト数> [, <③タイムアウト時間> ] )	
戻り値	戻り値	<受信データ> 文字列
パラ メータ	受信したデータを取得します。 データを受信する前にタイムアウトすると、それまで受信したデータか空の文字列が得られます。	
	①	<ネットワーク番号またはクライアント番号> 数値
	NWOPEN命令でオープンしたネットワーク番号を指定します。 または、NWACCEPT命令で受け取ったクライアント番号を指定します。	
	②	<受信バイト数> 数値
	データを受信するバイト数を指定します。 TCP通信時、負数(-1)を指定すると、改行コード(CHR\$(10))を検知するまで、データを受信します。(得られる文字列に、改行コードは含まれます) UDP通信時、負数(-1)を指定すると、呼び出された時点で受信可能な、1パケットの全バイト数のデータを受信します。受信可能サイズが0の時、タイムアウト設定如何に依らず、空文字列が返ります。	
	③	<タイムアウト時間> 数値
	受信待ちを行う最大時間(秒)を指定します。 省略または0を指定すると、受信完了まで待ち続けます。	
	<ul style="list-style-type: none"> <li>通信相手先がクローズしていた場合、空文字列が得られる場合があります。</li> <li>UDP通信時、受信バイト数に負数を指定した時、呼び出された時点の1パケットの受信データを得るため、タイムアウト時間の指定は無視されます。</li> <li>UDP通信時、パケット単位で通信を行う為、受け取る受信データのサイズが、指定した 受信バイト数を下回る事があります。</li> </ul>	
使用例	『2.2 TCPによるデータ送受信』および『2.3 UDPによるデータ送受信』を参照してください。	

## 10. NWRECVADR\$

関数							
機 能	ネットワークに対して、指定したバイト数のデータおよび相手先のIPアドレスを受信します。						
書 式	<(戻り値)受信データと相手先IPアドレス>=NWRECVADR\$(<①ネットワーク番号またはクライアント番号>,<②受信バイト数>[,<③タイムアウト時間>])						
戻り値	戻り値	<受信データと相手先IPアドレス>					
	配列						
	以下のような形式の文字列配列が得られます。						
	<table><tr><th>添字</th><th>内容</th></tr><tr><td>(0)</td><td>受信したデータを取得します。 データを受信する前にタイムアウトすると、それまで受信したデータか空の文字列が得られます。</td></tr><tr><td>(1)</td><td>通信相手先のIPアドレスとポート番号が得られます。 「IPアドレス：ポート番号」の形式です。</td></tr></table>		添字	内容	(0)	受信したデータを取得します。 データを受信する前にタイムアウトすると、それまで受信したデータか空の文字列が得られます。	(1)
添字	内容						
(0)	受信したデータを取得します。 データを受信する前にタイムアウトすると、それまで受信したデータか空の文字列が得られます。						
(1)	通信相手先のIPアドレスとポート番号が得られます。 「IPアドレス：ポート番号」の形式です。						
パラメータ	①	<ネットワーク番号またはクライアント番号>					
	数値						
	NWOPEN命令でオープンしたネットワーク番号を指定します。 または、NWACCEPT命令で受け取ったクライアント番号を指定します。						
	②	<受信バイト数>					
	数値						
	データを受信するバイト数を指定します。 TCP通信時、負数(-1)を指定すると、改行コード(CHR\$(10))を検知するまで、データを受信します。(得られる文字列に、改行コードは含まれます) UDP通信時、負数(-1)を指定すると、呼び出された時点で受信可能な、1パケットの全バイト数のデータを受信します。受信可能サイズが0の時、タイムアウト設定如何に依らず、空文字列が返ります。						
	③	<タイムアウト時間>					
	数値						
受信待ちを行う最大時間(秒)を指定します。 省略または0を指定すると、受信完了まで待ち続けます。							
備 考	・ 本コマンドは、戻り値が文字列配列で、通信相手先のIPアドレス情報が得られる以外、制限事項および挙動は、NWRCV\$ と同じです。 UDP通信時、通信相手先の情報が必要な時に使用ください。						
使用例	NO=1 'NWOPENで、オープン済みのネットワーク番号と仮定します LIST A\$ A\$=NWRECVADR\$(NO, -1) ' 受信および相手先情報を得ます PRINT "受信データ="; A\$(0) PRINT "相手先情報="; A\$(1)						

## 11. NWSTAT

関数														
機 能	指定したネットワークの状態を得ます。													
書 式	<(戻り値)状態値>=NWSTAT(<①ネットワーク番号またはクライアント番号>,<②状態取得ID>[,<③オプション値>])													
戻り値	戻り値	<状態値> 数値 状態取得IDにより、ネットワークの状態値が得られます。												
パラ メータ	①	<ネットワーク番号またはクライアント番号> 数値 NWOPEN命令でオープンしたネットワーク番号を指定します。 または、NWACCEPT命令で受け取ったクライアント番号を指定します。												
	②	<状態取得ID> 文字列 ネットワークのどんな状態を得たいか、文字列で指定します。												
	<table><tr><th>状態取得ID</th><th>オプション値</th><th>戻り値</th></tr><tr><td>"RCVSIZE"</td><td>—</td><td>呼び出した時点で、受信可能なバイトサイズを数値で得られます。</td></tr><tr><td>"COND"</td><td>—</td><td>送受信可能な状態を、ビット単位の組み合わせで得られます。 &amp;H01：受信可能 &amp;H04：送信可能  上記以外のビット値が得られる事があるので、判定を行うには AND 演算を使用して下さい。</td></tr><tr><td>"ERR"</td><td>—</td><td>送受信が成功したら 0 を。 エラーが発生したら 0以外がセットされます。  タイムアウトエラーが発生したか否かの判定等に用います。</td></tr></table>		状態取得ID	オプション値	戻り値	"RCVSIZE"	—	呼び出した時点で、受信可能なバイトサイズを数値で得られます。	"COND"	—	送受信可能な状態を、ビット単位の組み合わせで得られます。 &H01：受信可能 &H04：送信可能  上記以外のビット値が得られる事があるので、判定を行うには AND 演算を使用して下さい。	"ERR"	—	送受信が成功したら 0 を。 エラーが発生したら 0以外がセットされます。  タイムアウトエラーが発生したか否かの判定等に用います。
	状態取得ID	オプション値	戻り値											
	"RCVSIZE"	—	呼び出した時点で、受信可能なバイトサイズを数値で得られます。											
"COND"	—	送受信可能な状態を、ビット単位の組み合わせで得られます。 &H01：受信可能 &H04：送信可能  上記以外のビット値が得られる事があるので、判定を行うには AND 演算を使用して下さい。												
"ERR"	—	送受信が成功したら 0 を。 エラーが発生したら 0以外がセットされます。  タイムアウトエラーが発生したか否かの判定等に用います。												
③	<オプション値> 値 指定した、状態取得IDにより、オプション値を指定可能です。													
備 考	・TCPサーバ動作時のネットワーク番号(クライアント番号ではない)を使って、"RCVSIZE"を取得しようとする事は出来ません。													
使用例	DO WHILE TRUE SZ = NWSTAT(1, "RCVSIZE") IF SZ <> 0 THEN A\$ = NWRECV(1, SZ)    ' 受信可能サイズ分のみ受信するようにします END IF LOOP													

## 12. NWIPADDR\$

関数			
機 能	自身のネットワークが認識しているIPアドレス情報を、文字列配列で得ます。		
書 式	<(戻り値)IPアドレス>=NWIPADDR\$([ <①オプション値> ])		
戻り値	戻り値	<IPアドレス>	配列
	自身のネットワークが認識しているIPアドレスを、文字列配列形式で取得します。		
パラ メータ	①	<オプション値>	数値
	取得するIPアドレスの形式を指定できます。 1を指定すると IPv4形式のIPアドレス。2を指定すると IPv6形式のIPアドレスが得られます。 両方を OR して、3を指定する事も可能です。 省略時、1が採用されます。		
備 考	本コマンドは、Linuxのコマンドで「ip a」コマンドを呼び出して、IPアドレス情報を得るのと、機能的にほぼ等しいです。		
使用例 1	PRINT NWIPADDR\$() ' ネットワークが認識しているIPアドレスが得られます。 ' 例えば、「[ 127.0.0.1, 10.80.40.61 ]」のような表示結果が得られます。		
使用例 2	PRINT NWIPADDR\$(3) ' IPv4アドレスとIPv6アドレスの両方を得る事も可能です。 ' 例えば、「[ 127.0.0.1, 10.80.40.61, fe80::3c:9580:6d1e:c251%eth1 ]」のような表示結果が得られます。		

### 3.3 コマンド個別説明(MQTT通信)

#### 1.MQOPEN

命令	
機 能	MQTT通信を行うためにオープンします。
書 式	MQOPEN <①IPアドレスとポート番号とトピック名> FOR <②入出力モード> AS [#]<③MQTT番号>
パラ メータ	<div>①</div> <div>&lt;IPアドレスとポート番号とトピック名&gt;</div> <div>文字列</div> <p>文字列で「[プロトコル:] IPアドレス [: ポート番号] / トピック名」の形式で記述します。 プロトコルは「mqtt」と記述します(省略可能)。</p> <p>IPアドレスとポート番号は、 MQTTの用語で言う、broker へのIPアドレスとポート番号を指定します。 ポート番号を省略すると、1883番を指定するのと同じ効果です。</p> <p>トピック名は、メッセージを送受信する宛先を指定します。 受信時、「hoge/+」のように「+」を付加して指定すると、「hoge/1」、「hoge/2」などの複数の宛先からのメッセージを受信できます。</p>
	<div>②</div> <div>&lt;入出力モード&gt;</div> <div>キーワード</div> <p>メッセージを送信したい場合は「OUTPUT」、 メッセージを受信したい場合は「INPUT」 と指定してください。</p>
	<div>③</div> <div>&lt;MQTT番号&gt;</div> <div>数値</div> <p>作成したネットワークインタフェースに関連付けるMQTT番号を1～15の範囲で指定します。 MQTT通信の制御はこの&lt;MQTT番号&gt;を指定して行います。</p>
備 考	・
使用例	MQOPEN "localhost:1883/hoge/1" FOR OUTPUT AS #1 MQSEND #1, "hello world" ' トピック名「hoge/1」で、MQTT通信用にオープンします。

## 2.MQCLOSE

命令		
機能	MQTT通信をクローズします。	
書式	MQCLOSE [#]<①MQTT番号>	
パラメータ	①	<MQTT番号> 数値
	MQOPEN命令でオープンしたMQTT番号を指定します。	
使用例	MQOPEN "localhost:1883/hoge/1" FOR OUTPUT AS #1 MQCLOSE #1 'MQTT番号 1 でオープンした後、クローズします。'	

## 3.MQSEND

命令		
機能	MQTT通信に対してメッセージを送信します。	
書式	MQSEND [#]<①MQTT番号>, <②送信メッセージ>	
パラメータ	①	<MQTT番号> 数値
	MQOPEN命令でオープンしたMQTT番号を指定します。	
	②	<送信メッセージ> 文字列
送信したいメッセージを文字列形式で指定します。		
備考	<ul style="list-style-type: none"> <li>このコマンドを呼び出すと、MQOPEN で指定したトピック名に対して、メッセージを送信します。</li> <li>MQTT通信的には、メッセージを、broker に対して、publishします。</li> <li>送信メッセージの最大サイズは、MQTTの規格では256MBとなっています。実際には、AJANでのメモリ容量、MQTTの broker の仕様などに依存します。</li> </ul>	
使用例	MQOPEN "localhost:1883/hoge/1" FOR OUTPUT AS #1 MQSEND #1, "hello world" '「hello world」というメッセージを送信します。'	

4.MQSTAT

関数					
機 能	MQTT通信に対して受信データの取り出し可能件数を得ます。				
書 式	<(戻り値)状態値> = MQSTAT( <①MQTT番号>, <②状態取得ID> )				
戻り値	戻り値      <状態値>      数値				
	状態取得IDにより、MQTT通信の状態値が得られます。				
パラ メータ	①      <MQTT番号>      数値				
	MQOPEN命令でオープンしたMQTT番号を指定します。				
	②      <状態取得ID>      文字列				
	MQTT通信のどんな状態を得たいか、文字列で指定します。				
<table><tr><td>状態取得ID</td><td>戻り値</td></tr><tr><td>"RECVCNT"</td><td>受信可能なメッセージ件数が得られます。</td></tr></table>		状態取得ID	戻り値	"RECVCNT"	受信可能なメッセージ件数が得られます。
状態取得ID	戻り値				
"RECVCNT"	受信可能なメッセージ件数が得られます。				
備 考	・				
使用例	MQOPEN "localhost:1883/hoge/1" FOR INPUT AS #1 ? "件数="; MQSTAT(1, "RECVCNT") ' 受信可能なメッセージ件数を取得して表示します。				

## 5.MQRECV\$

関数													
機能	MQTT通信に対して受信メッセージを受け取ります。												
書式	<(戻り値)受信メッセージ>=MQRECV\$( <①MQTT番号>, <②タイムアウト秒> )												
戻り値	戻り値	<受信メッセージ>	配列										
	受信したメッセージを取得します。 メッセージを受信する前にタイムアウトすると、配列要素1の空文字列が得られます。 メッセージを受信した時、以下の形式の文字列配列が得られます。												
	<table><tr><th>配列の添字</th><th>内容</th></tr><tr><td>0</td><td>受信メッセージ</td></tr><tr><td>1</td><td>トピック名</td></tr><tr><td>2</td><td>状態値(基本：0)</td></tr><tr><td>3</td><td>受信時の日時情報</td></tr></table>			配列の添字	内容	0	受信メッセージ	1	トピック名	2	状態値(基本：0)	3	受信時の日時情報
	配列の添字	内容											
	0	受信メッセージ											
1	トピック名												
2	状態値(基本：0)												
3	受信時の日時情報												
パラメータ	①	<MQTT番号>	数値										
	MQOPEN命令でオープンしたMQTT番号を指定します。												
	②	<タイムアウト秒>	数値										
受信待ちを行う最大時間(秒)を指定します。 省略または0を指定した時、受信待ちを行わずに その時点のメッセージまたは受信してなければ、空文字列が得られます。													
備考	・MQTT通信的には、broker から、subscribeしたメッセージを受け取ります。												
使用例	LIST ITEMS\$  MQOPEN "localhost:1883/hoge/1" FOR INPUT AS #1 IF MQSTAT(1, "RECVcnt") > 0 THEN ITEMS\$ = MQRECV\$(1) ? "受信データ:"; ITEMS\$ END IF MQCLOSE #1  'MQSTATで、受信可能なメッセージ件数が1以上あれば、MQRECV\$ でメッセージ受信します。												



## 第4章 サンプルプログラム

サンプルプログラムは「/usr/share/interface/AJANPro/samples/NWL/」に格納されています。

AJAN統合開発環境を起動すると、左ペインのエクスプローラウィンドウ内の「Samples/NWL/」に、ファイルが取り込まれて配置されます。

### 4.1 サンプルプログラム一覧

ファイル名	内 容
<b>●TCP</b>	
NWTCPSRV.AJN	TCP通信によるサーバー処理のサンプルプログラムです。 NWTCPSRV.AJNを実行してから、別の端末でNWTCPCLIAJNを実行します。 NWACCEPT関数で、クライアントからの接続を待ち、NWRECV\$関数で、クライアントから送られたデータを11バイト受信し、NWSEND命令で、受信したデータを返します。これを3回繰り返した後、終了します。
NWTCPCLIAJN	TCP通信によるクライアント処理のサンプルプログラムです。 NWTCPSRV.AJNを実行してから、別の端末でNWTCPCLIAJNを実行します。 NWSEND命令で「hello world」(11バイト分)を送信した後、NWRECV\$関数で、サーバーから送られるデータを受信します。 これを3回繰り返した後、終了します。
NWTCPv6SRV.AJN	NWTCPSRV.AJNのサーバー処理のIPv6対応版です。 NWTCPv6SRV.AJNを実行してから、別の端末でNWTCPv6CLIAJNを実行します。 NWOPENでオープンする際、IPv6のIPアドレスでオープンします。その後の処理は、NWTCPSRV.AJNと同じく、クライアントからのデータを受信したら返すよう送信する処理を、3回繰り返して終了します。
NWTCPv6CLIAJN	NWTCPCLIAJNのクライアント処理のIPv6対応版です。 NWTCPv6SRV.AJNを実行してから、別の端末でNWTCPv6CLIAJNを実行します。 NWOPENでオープンする際、IPv6のIPアドレスでオープンします。その後の処理は、NWTCPCLIAJNと同じく、サーバーに対してデータを送信したら受信する処理を、3回繰り返して終了します。
NWTCPMSRV.AJN	NWTCPSRV.AJNのサーバー処理の、マルチスレッド対応版です。 NWTCPMSRV.AJNを実行してから、別の端末でNWTCPMCLIAJNを実行します。 複数のクライアントからの接続に同時対応できます。 NWACCEPT関数で、クライアントから接続されると、スレッドを生成して、クライアントとの通信処理を別スレッドに任せます。スレッド内では、データを受信したら返すのを、3回繰り返して終了します。 15回、クライアントから接続されると、プログラムを終了します。
NWTCPMCLIAJN	NWTCPMSRV.AJNを動かす際の、クライアント処理対応版です。 NWTCPMSRV.AJNを実行してから、別の端末でNWTCPMCLIAJNを実行します。 NWSEND命令で、改行付きの文字列を送信した後、NWRECV\$関数で、サーバーから送られるデータを、改行単位で受信します。 これを3回繰り返した後、終了します。
NWTCPWEBCLIAJN	TCP通信を使って、簡単なWebクライアント機能のサンプルプログラムです。 実行すると、ホスト名とURLのパス名の入力を求めます。 入力後、指定したホスト名のWebサーバー(HTTP 80番ポート)にアクセスし、HTMLデータを取得&表示します。 例えば、Webブラウザで「http://127.0.0.1/list.ajin」を表示している時の、HTMLデータを取得したい場合、最初のホスト名入力で「127.0.0.1」、次のURLパス名入力で「/list.ajin」を入力します。
<b>●UDP</b>	
NWUDPSRV.AJN	UDP通信によるサーバー処理のサンプルプログラムです。 NWUDPSRV.AJNを実行してから、別の端末でNWUDPCLIAJNを実行します。 NWRECV\$関数で、クライアントから送られるデータを11バイト受信するまで待機し、受信後、NWSEND命令で、受信したデータを返します。 これを3回繰り返した後、終了します。

## ネットワークコマンドリファレンス

NWUDPCLI.AJN	UDP通信によるクライアント処理のサンプルプログラムです。 NWUDPSRV.AJNを実行してから、別の端末でNWUDPCLI.AJNを実行します。 NWSEND命令で、「hello world」(11バイト分)を送信した後、NWRECV\$関数で、サーバーから送られるデータを受信します。 これを3回繰り返した後、終了します。
NWUDPSRV2.AJN	NWUDPSRV.AJNのサーバー処理を、パケット単位で送受信するサンプルプログラムです。 NWUDPSRV2.AJNを実行してから、別の端末でNWUDPCLI2.AJNを実行します。 NWRECV\$関数で、クライアントからデータを受信する際、第2引数に -1 を指定して、全データ受信しています。これにより、有効な受信データが得られるまで、繰り返しNWRECV\$を呼びます。 データを受信したら、NWSEND命令で、受信したデータを送り返します。 これを3回繰り返した後、終了します。
NWUDPCLI2.AJN	NWUDPCLI.AJNのクライアント処理を、パケット単位で送受信するサンプルプログラムです。 NWUDPSRV2.AJNを実行してから、別の端末でNWUDPCLI2.AJNを実行します。 NWSEND命令で、文字列を送信した後、NWRECV\$関数で、サーバーから送られるデータを受信します。受信する際、NWRECV\$関数の第2引数に -1 を指定して全データしています。 これにより、有効な受信データが得られるまで、繰り返しNWRECV\$を呼びます。 これを3回繰り返した後、終了します。
NWUDPSRV_BROADCAST.AJN	UDPブロードキャスト通信のサーバー処理のサンプルプログラムです。 NWUDPSRV_BROADCAST.AJN を実行してから、別の端末でNWUDPCLI_BROADCAST.AJN を実行します。 NWUDPSRV_BROADCAST.AJNを実行する端末は、複数台可能です。 NWRECV\$関数で、クライアントからパケット単位に全データを受信して表示する事を、複数回繰り返して終了します。
NWUDPCLI_BROADCAST.AJN	UDPブロードキャスト通信のクライアント処理のサンプルプログラムです。 NWUDPSRV_BROADCAST.AJN を実行してから、別の端末でNWUDPCLI_BROADCAST.AJNを実行します。 NWUDPSRV_BROADCAST.AJNを実行する端末は、複数台可能です。 NWOPEN命令でオープンする際、ブロードキャストアドレスを指定しています。 これにより、NWSEND命令で送信したデータは、一度に複数台の端末に対して送られます。 データの送信は、複数回行われた後に終了します。
NWUDPSRV_MULTICAST.AJN	UDPマルチキャスト通信のサーバー処理のサンプルプログラムです。 NWUDPSRV_MULTICAST.AJN を実行してから、別の端末でNWUDPCLI_MULTICAST.AJNを実行します。 NWUDPSRV_MULTICAST.AJNを実行する端末は、複数台可能です。 NWOPEN命令でオープンする際、クライアントから送られるマルチキャストアドレスを受け取れるように指定します。 次に、NWRECV\$関数で、クライアントからパケット単位に全データを受信して表示する事を、複数回繰り返して終了します。
NWUDPCLI_MULTICAST.AJN	UDPマルチキャスト通信のクライアント処理のサンプルプログラムです。 NWUDPSRV_MULTICAST.AJN を実行してから、別の端末でNWUDPCLI_MULTICAST.AJNを実行します。 NWUDPSRV_MULTICAST.AJNを実行する端末は、複数台可能です。 NWOPEN命令でオープンする際、マルチキャストアドレスを指定しています。 これにより、NWSEND命令で送信したデータは、一度に複数台の端末に対して送られます。 データの送信は、複数回行われた後に終了します。
●MQTT	
MQTT.AJN	MQTT通信による、送受信処理のサンプルプログラムです。 実行すると、MQTT BrokerのIPアドレスの入力、送信か受信の選択、トピックIDの入力を行います。 送信を選択すると、キーボードから入力した1文字ずつ、MQSEND命令でデータを送信します。 受信を選択すると、MQSTAT関数で受信を確認しつつ、MQRECV\$で受信したデータを表示します。 どちらも、「Q」キーを押すと、プログラムが終了します。

## 第5章 索引

### **M**

MQCLOSE .....	30
MQOPEN .....	29
MQRECV\$ .....	32
MQSEND .....	30
MQSTAT .....	31

### **N**

NWACCEPT .....	20
NWCLOSE .....	20
NWFREEFILE .....	21

NWIPADDR\$ .....	28
NWOPEN .....	18
NWRECV\$ .....	25
NWRECVADR\$ .....	26
NWSEND .....	21
NWSENDADR .....	22
NWSENDNR .....	23
NWSENDNRADR .....	24
NWSTAT .....	27

## 第6章 重要な情報

---

### 保証の内容と制限

弊社は本ドキュメントに含まれるソースプログラムの実行が中断しないこと、またはその実行に誤りが無いことを保証していません。

本製品の品質や使用に起因する、性能に起因するいかなるリスクも使用者が負うものとします。

弊社はドキュメント内の情報の正確さに万全を期しています。万一、誤記または誤植などがあつた場合、弊社は予告無く改訂する場合があります。ドキュメントまたはドキュメント内の情報に起因するいかなる損害に対しても弊社は責任を負いません。

ドキュメント内の図や表は説明のためであり、ユーザ個別の応用事例により変化する場合があります。

### 著作権、知的所有権

弊社は本製品に含まれるおよび本製品に対する権利や知的所有権を保持しています。

本製品はコンピュータ ソフトウェア、映像/音声(例えば図、文章、写真など)を含んでいます。

### 医療機器/器具への適用における注意

弊社の製品は人命に関わるような状況下で使用される機器に用いられる事を目的として設計、製造された物では有りません。

弊社の製品は人体の検査などに使用するに適する信頼性を確保する事を意図された部品や検査機器と共に設計された物では有りません。

医療機器、治療器具などの本製品の適用により、製品の故障、ユーザ、設計者の過失などにより、損傷/損害を引き起こす場合が有ります。

### 複製の禁止

弊社の許可なく、本ドキュメントの全て、または一部に関わらず、複製、改変などを行うことはできません。

### 責任の制限

弊社は、弊社または再販売者の予見の有無にかかわらず発生したいかなる特別損害、偶発的損害、間接的な損害、重大な損害について、責任を負いません。

本製品(ハードウェア、ソフトウェア)のシステム組み込み、使用、ならびに本製品から得られる結果に関する一切のリスクについては、本製品の使用者に帰属するものとします。

本製品に含まれる不都合、あるいは本製品の供給(納期遅延)、性能もしくは使用に起因する付帯的損害もしくは間接的損害に対して、弊社に全面的に責がある場合でも、弊社はその製品に対する改良(有償サービスの利用)、代品交換までとし、製品の予防交換並びに、代金減額等、金銭面での賠償の責任は負わないものとします。

本製品は、日本国内仕様です。

### 商標/登録商標

本書に掲載されている会社名、製品名は、それぞれ各社の商標または登録商標です。

## 改訂履歴

Ver.	年 月	改 訂 内 容
0.90	2019年10月	新規作成
1.00	2022年1月	最新情報に更新
1.01	2023年3月	NWOPENの説明追加。

このマニュアルは、製品の改良その他により将来予告なく改訂しますので、予めご了承ください。