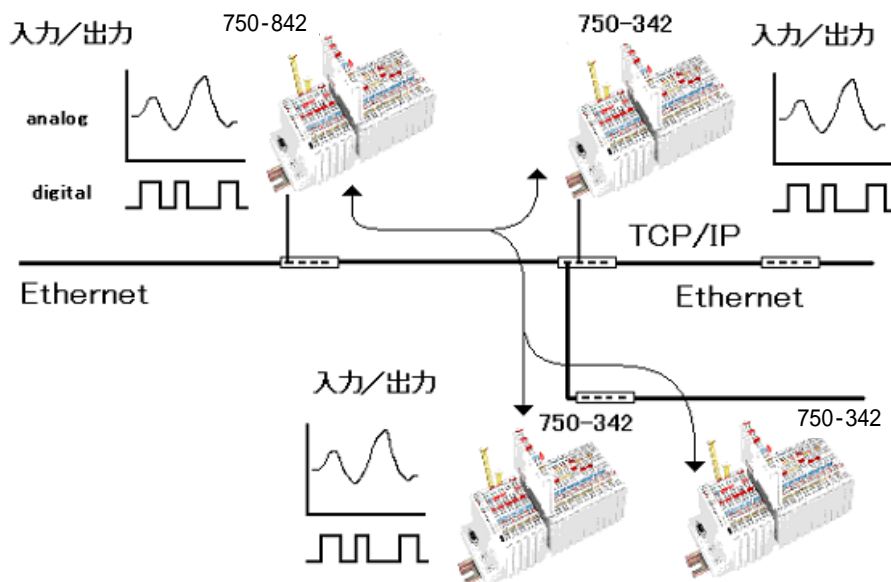


1. 概要

ワゴ I/O システムの Ethernet バスコントローラとバスカプラとの間で Modbus/TCP あるいは Modbus/UDP で相互通信を行い、データ交換を行うことができます。交換されたデータを各々のコントローラ/カプラの先頭アドレスからデータをアサインすることによって、手元で入力したデータを離れた場所で出力したり、離れた場所を入力したデータを手元で出力するなどのピアツーピア通信が実現できます。

データ通信をさせるための上位のパソコンやコントローラは不要で、WAGO-I0-PRO でプログラミングすることによって複数のノードを順番に接続してデータ通信を行う Polling 通信や自分の状態が変化したときだけパケット通信を行う COS 通信 (Change Of State) などが可能です。



2. ライブラリと使用関数名

通信に使用する `ETHERNET_MODBUSMASTER_TCP` または `ETHERNET_MODBUSMASTER_UDP` の関数 (FB) は、`ModbusEthernet_03.lib` の中に収められています。ライブラリとサンプルプログラムについては、

<http://www.wago.co.jp/io>

のソフトウェアダウンロードのページからダウンロード出来ます。

代表的な FB である ETHERNET_MODBUSMASTER_TCP 関数に用いられる入出力の内容を以下に説明します。

| ETHERNET_MODBUSMASTER_TCP | |
|---------------------------|-------------------|
| -strIP_ADDRESS | wERROR |
| -bUNIT_ID | xREADY |
| -bFUNCTION_CODE | bRESPONSE_UNIT_ID |
| -wREAD_ADDRESS | |
| -wREAD_QUANTITY | |
| -ptREAD_DATA | |
| -wWRITE_ADDRESS | |
| -wWRITE_QUANTITY | |
| -ptSEND_DATA | |
| -xSTART | |
| -xRESET | |
| -tTIME_OUT | |

| 入出力端子名 | 変数の種類 | 変数の型 | 説明 |
|-------------------|-------|-------|--|
| Str IP_ADRESS | 入力 | 文字列 | 通信先のバスカプラの IP アドレスを指定します。 |
| bUNIT_ID | 入力 | バイト | 16#01 に設定しておきます。 |
| bFUNCTION_CODE | 入力 | バイト | 16#17 (データ交換用コード番号) に設定します。 |
| wREAD_ADDRESS | 入力 | ワード | 通信相手の入力アドレスを 16 進数で設定します。16#0000 の場合は相手のバスカプラ中の入力先頭アドレスを意味します。 |
| wREAD_QUANTITY | 入力 | ワード | 上で指定したアドレスから何ワード分読むかを 16 進数で設定します。16#0003 の場合は 3 ワード分読み出すことを意味します。 |
| ptREAD_DATA | 入力 | ポインタ | 通信相手から読み出したデータを受信バッファに書き込むときに受信バッファのアドレスのポインタです。オペレーション「ADR」を使用して入力してください。 |
| wWriteAddress | 入力 | ワード | 通信相手の出力アドレスを 16 進数で設定します。16#0000 の場合は相手のバスカプラ中の出力先頭アドレスを意味します。 |
| wWriteQuantity | 入力 | ワード | 上で指定したアドレスから何ワード分書き込むかを 16 進数で設定します。16#0003 の場合は 3 ワード分書き込むことを意味します。 |
| ptSEND_DATA | 入力 | ワード | 通信相手の出力アドレスにデータ書き込むときに自分の送信バッファアドレスのポインタです。オペレーション「ADR」を使用して入力してください。 |
| xSTART | 入力 | ビット | 通信開始の指示をビットで指定します。FALSE→TRUE の立ち上がりで通信動作を開始します。 |
| xRESET | 入力 | ビット | 通信状態をリセット (初期状態に戻す) ときに使います。通常は FALSE にしておいてください。 |
| tTIME_OUT | 入力 | 時間データ | タイムアウト処理時間を指定します。この時間内で通信が完了しない場合は自動的に処理を中止します。 |
| wERROR | 出力 | バイト | 通信エラーが発生したときにエラーコードを出力します。通常は使用しません。 |
| xREADY | 出力 | ビット | 通信動作が終了するとこの出力は FALSE から TRUE に変化し、動作が始まると再び FALSE に戻ります。 |
| bRESPONSE_UNIT_ID | 出力 | バイト | 通常は使用しません。 |

3. ライブラリとサンプルプログラムのダウンロードとコピー

750-842 のピアツーピア通信には ModbusEthernet_03.lib を使用します。

これを利用したプログラムを得るには以下のページにアクセスします。

http://www.wago.co.jp/io/download/D_00044.htm

上記のページにおいてプログラムおよびライブラリファイルは自己解凍形式で圧縮されていますので、この圧縮ファイルをダウンロードします。それを解凍すると以下の各ファイルが得られますので下記に示したフォルダにコピーします。

「プログラム」

842_trans_TCP.pro

842_trans_UDP.pro

842_cos_UDP.pro

842-poll-UDP.pro

} → C:\Program files\WAGO Software\CoDeSys V2.3\Projects

「ライブラリ」

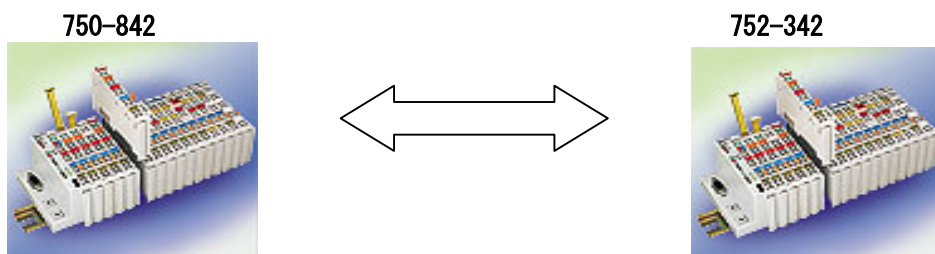
Ethernet.lib

ModbusEthernet_03.lib

} → C:\Program files\WAGO Software\CoDeSys V2.3\Targets
WAGO\Libraries\16_Bit

4. プログラム内容の解説（1対1通信）

ここでは、842_trans_TCP.pro および 842_trans_UDP.pro のプログラム内容について解説します。



受信バッファの定義

通信相手の入力データを読み出してこのデータを格納しておくバッファを自己定義します。サンプルプログラム中では **Rec_Buffer** という名前で3ワード定義しています。データ ARRAY で定義していますので、メモリ容量が許す限り何ワード分でも定義可能です。

Rec_Buffer ARRAY [1..3] OF WORD

●受信バッファをワード ARRAY で定義します。ここでは3ワード分を定義しています。

ReadAddress 16#0000

ReadQuantity 16#0003

●接続先の I/O システムの入力バッファを先頭アドレスから3ワード分読み込むことを意味します。

送信バッファの定義

通信相手の出力バッファに送るデータを格納しておくバッファを自己定義します。サンプルプログラム中では **Send_Buffer** という名前で3ワード定義しています。データ ARRAY で定義していますので、メモリ容量が許す限り何ワード分でも定義可能です。

Send_Buffer ARRAY [1..3] OF WORD

●送信バッファをワード ARRAY で定義します。ここでは3ワード分を定義しています。

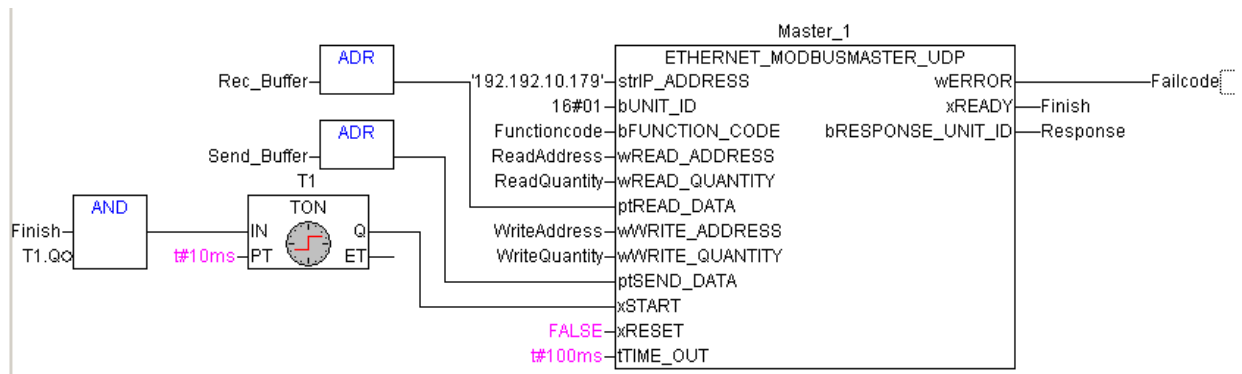
WriteAddress 16#0000

WriteQuantity 16#0003

●接続先の I/O システムの出力バッファへ先頭アドレスから3ワード分書き込むことを意味します。

送信相手の IP アドレス、送信時間間隔

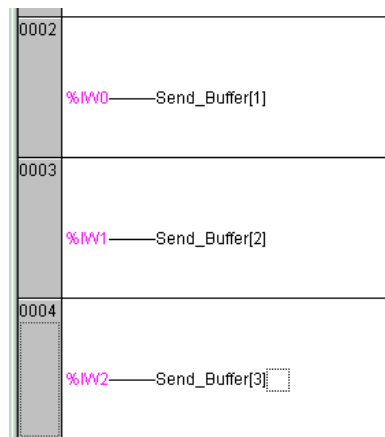
送信相手の IP アドレス（ここでは 192.192.10.179）、送受信の繰り返し時間間隔を設定します。以下の UDP 通信のサンプルプログラムではタイマー関数を用いて時間インターバル 10ms を設定しています。タイムアウトは 100ms に設定しています。TCP 通信を行う場合、時間インターバルは 100ms 以上、タイムアウトは 1000ms 以上に設定してください。



この通信用ファンクションブロックは通信一回ごとに Ethernet ポートのオープン→送受信→クローズを繰り返します。ポートオープンは「xSTART」が「TRUE」になったタイミングで行われ、通信が終了してポートをクローズした後は Finish が「TRUE」になります。

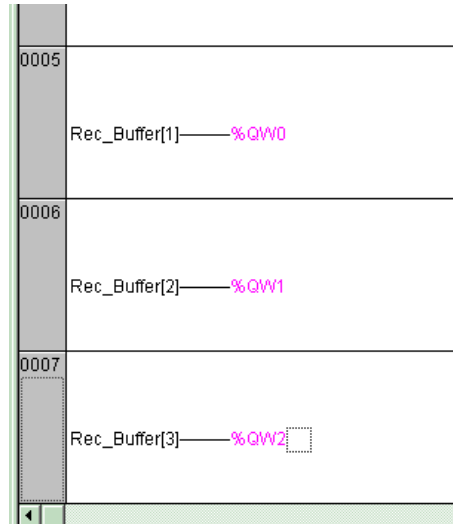
Send_Buffer へのデータのアサイン

毎回送信している Send_Buffer には自分の入力データをアサイン（割り当て）します。



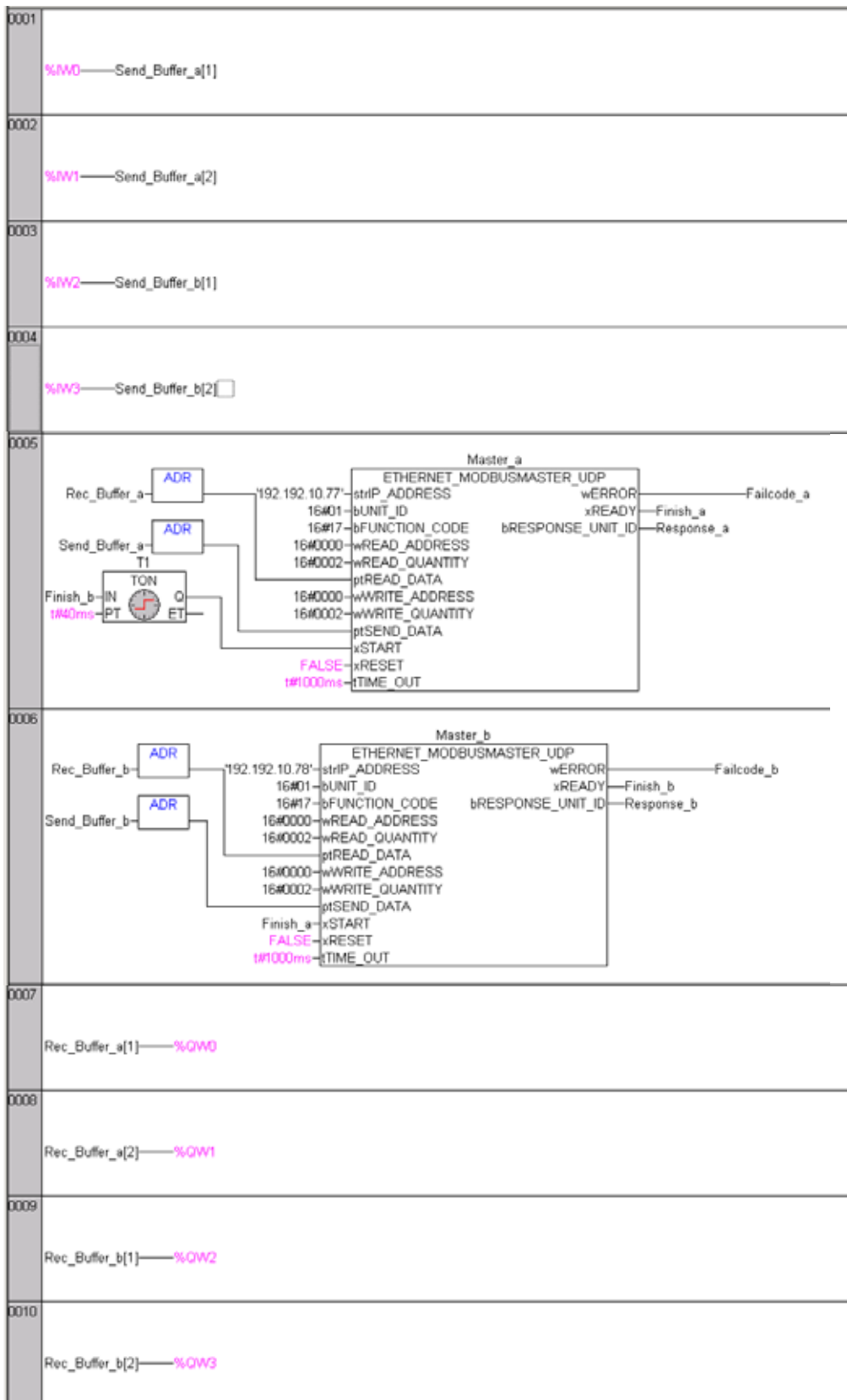
Rec_Buffer 内のデータのアサイン

受信されたデータは **Rec_Buffer** に記録されますが、これを自分の出力バッファにアサイン（割り当て）プログラムが必要になります。順番を逆にアサインすることによって相手の入力順と逆の順番で出力させることも可能です。



5. Polling 通信のプログラム内容の解説

ここでは、842_poll-UDP.pro のプログラム内容について解説します。例えば複数のノードと通信してデータの送受信を行う場合は xReady（通信動作終了）の信号を利用して複数のファンクションブロックが時系列的に送受信を繰り返すようにプログラムしていきます。ここでは 750-842 マスタが IP アドレスが 192.192.10.77（スレーブ 1）および 192.192.10.78（スレーブ 2）の各ノードとデータ交換を行います。



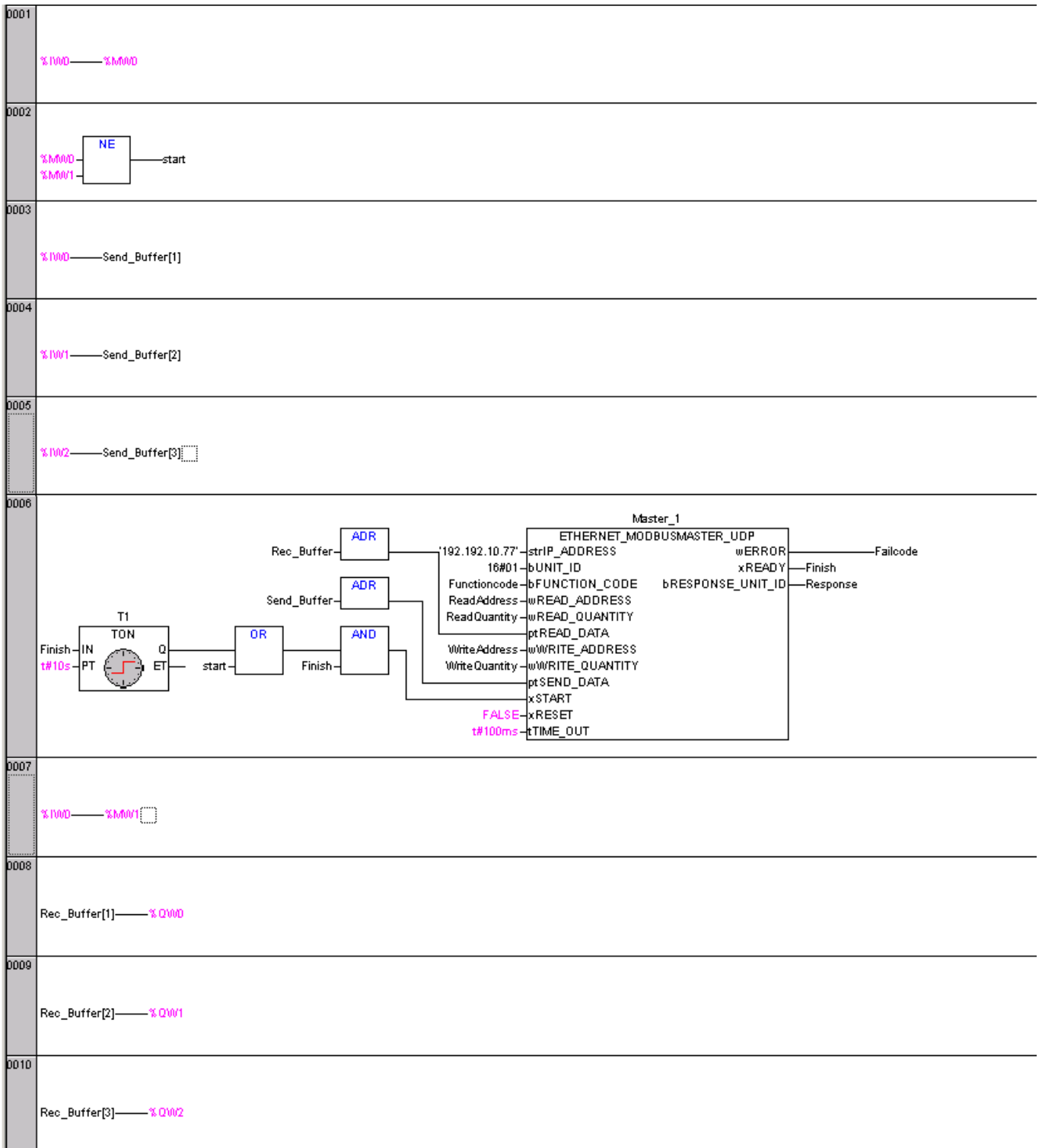
(842_pol |_UDP. pro の内容)

| プログラムステップ | プログラム処理内容の説明 |
|-----------|---|
| 0001~0002 | Master_a に接続されている Send_Buffer_a [1] ~Send_Buffer_a [2] へ、入力値領域の値 %IW0~%IW1 をアサイン (代入) 処理。 |
| 0003~0004 | Master_b に接続されている Send_Buffer_b [1] ~Send_Buffer_b [2] へ、入力領域の値 %IW2~%IW3 をアサイン (代入) 処理。 |
| 0005 | Master_a にて '192.192.10.77' と UDP 通信して Send_Buffer_a の値を送信、受信した値を Receive_Buffer_a にか格納。終了後 Finish_a (xReady) が ON。 |
| 0006 | Finish_a が True に変化するとこれをスタート信号として Master_b にて '192.192.10.78' と UDP 通信して Send_Buffer_b の値を送信、受信した値を Receive_Buffer_b にか格納。終了後 Finish_b (xReady) が ON。Finish_b はステップ Master_a の通信スタート信号として使用。 |
| 0007~0008 | Recive_Buffer_a [1] ~Recive_Buffer_a [2] に格納されたデータを出力領域 %QW0 ~%QW1 へアサイン (代入) 処理。 |
| 0009~0010 | Recive_Buffer_b [1] ~Recive_Buffer_b [2] に格納されたデータを出力領域 %QW2 ~%QW3 へアサイン (代入) 処理。 |

上記の結果、マスタにおいてアドレス%IW0 から2ワード分の入力データがスレーブ1に、また%IW2 から2ワード分の入力データがスレーブ2に送信されます。一方、スレーブ1から送られたデータは先頭から2ワード分マスタのアドレス%QW0を始めとして、またスレーブ2から送られたデータは先頭から2ワード分マスタのアドレス%QW2を始めとして出力されます。

6. COS 通信 (Change Of State) のプログラム解説

ここでは、842_cos_udp.pro のプログラム内容について解説します。Polling 形式で通信を行わせる場合は一定時間間隔でパケット送信が行われます。これに対して当該ノードの入力に何か変化が発生したときだけ送受信の動作を行わせる (Change of State) のがこのサンプルプログラムです。



- ① ステップ 0001 入力領域%IWO のデータをまず内部領域の%MWO へ代入（アサイン）します。
- ② ステップ 0002 内部領域の%MWO と%MW1 のデータを比較して等しくない場合に' start' を True にします。
- ③ ステップ 0003~0005 Send_Buffer [1] ~Send_Buffer [3] へ%IWO~%IW2 の値をアサイン（代入）します。
- ④ ステップ 0006 ②で出力された' start' が ON (True) かあるいは 10 秒が経過するかどうか
どち
らかの条件で Master1 の通信 FB が動作します。つまり、入力の状態が変化しなくても 10 秒に一回は通信動作を行います。COS 通信の場合は通信ミスのカバー するためにこのような処理が必要となります。
- ⑤ ステップ 0007 通信動作直後に%IWO を%MW1 にアサイン（代入）します。これによって%MWO と%MW1 が等しくなります。プログラムはステップ 0001~0010 まで常に回っていますので、入力値%IWO が 1 ビットでも変化するとステップ 0002 で検出されて' start' が True になって通信動作が行われます。
- ⑥ ステップ 0008~0010 Rec_Buffer [1] ~ Rec_Buffer [3] に格納されたデータを%QW0~%QW3 にアサイン（代入）します。