BECKHOFF

Komunikacja między sterownikami przez protokół ADS

Poziom trudności: łatwy

Wersja dokumentacji: 1.0

Aktualizacja: 20.03.2015

Beckhoff Automation Sp. z o. o.

Spis treści

1. Komunikacja ADS

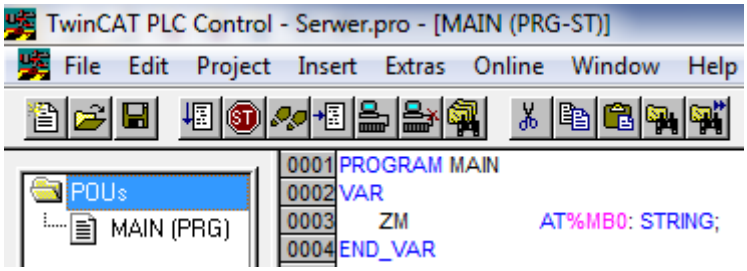
Protokół ADS to protokół komunikacyjny wykorzystywany w systemie TwinCAT. Definiuje on komunikację typu klient – serwer. Serwerem nazywamy komputer (sterownik), który udostępnia pewne dane, natomiast klientami nazywamy komputery (sterowniki), które odczytują bądź zapisują dane na serwerze. Sieć oparta na protokole ADS jest siecią typu multimaster, co oznacza, że w sieci może występować wiele jednostek typu klient i serwer. W przypadku sterowników z serii CX możliwe jest nawiązanie nieograniczonej liczby połączeń ADS. W przypadku sterowników serii BX możliwe jest nawiązanie 4 połączeń, w praktyce 3, ponieważ jedno jest zarezerwowane na programowanie sterownika.

2. Konfiguracja sterowników

W celu przesyłania danych pomiędzy dwoma sterownikami za pomocą protokołu ADS, każdy z nich musi mieć dodany drugi sterownik jako route'a. Informacje jak to zrobić można znaleźć w odpowiednim dokumencie na ftp://ftp.beckhoff.com/poland/Pomoc/ . Realizacja wymiany zmiennych za pomocą protokołu ADS odbywa się w programie PLC.

3. Serwer – program PLC

Możliwy jest odczyt/zapis danych znajdujących się w przestrzeni Memory (zadeklarowanych … AT %M …). W pokazanym przykładzie z serwera odczytywana jest zmienna typu STRING zadeklarowana w sposób pokazany poniżej.



4. Klient – program PLC

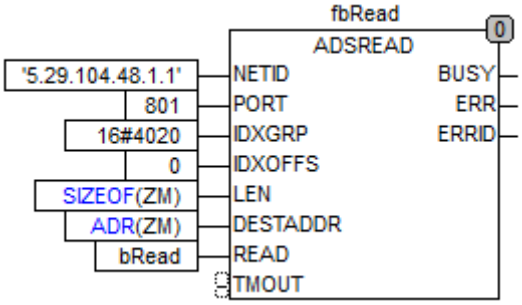
W programie sterownika pełniącego rolę klienta konieczne jest dodanie odpowiedniej biblioteki – TcSystem.lib dla PC i CX lub TcSystemBX.lbx dla sterowników serii BX.
W programie uruchomianym na kliencie używamy bloku funkcyjnego **ADSREAD**. W bloku tym wsytępują następujące wejścia:

- **NETID (T_AmsNetId) -** Adres ADS sterownika, który jest serwerem,
- **PORT (T_AmsPort) -** Numer portu wykorzystywanego przez PLC Runtime. W przypadku PC i CX dla Runtime 1 jest to port 801, dla Runtime 2 jest to port 811 itd. BX i BCxx50 port 800.
- **IDXGRP (UDINT) -** Indeks grupy usługi ADS. Wskazuje on na obszar pamięci z którego dokonujemy odczytu. W przypadku odczytu/zapisu zmiennych przestrzeni Memory (flaga %M) jest to zawsze wartość 16#4020,
- **IDXOFFS (UDINT) -** Indeks offsetu usługi ADS. Wskazuje on od którego bajtu chcemy odczytywać pamięć,
- **LEN (UDINT) -** Ilość bajtów które chcemy odczytać. Jest to jednocześnie rozmiar zmiennej do której dokonujemy zapisu (w bajtach),

- **DESTADR (DWORD) -** Adres zmiennej do której wpisujemy odczytaną wartość, najczęściej odczytujemy go funkcją ADR,
- **READ (BOOL) -** Wejście reagujące na zbocze narastające, służy do wydawania polecenia odczytu,
- **TMOUT (TIME) -** Limit czasu odczytu, po jego przekroczeniu blok sygnalizuje błąd.

Wyjścia z bloku są następujące:

- **BUSY (BOOL) -** Wyjście sygnalizujące, że blok wykonuje operację odczytu,
- **ERR (BOOL) -** Wyjście sygnalizujące wystąpienie błędu. Przyjmuje ono wartość TRUE, gdy wyjście BUSY przyjmie wartość FALSE, a odczyt zakończy się niepowodzeniem,
- **ERRID (UDINT) -** ADS Return Code, znaczenie kodu można odnaleźć w dokumentacji protokołu.



5. Tips & tricks

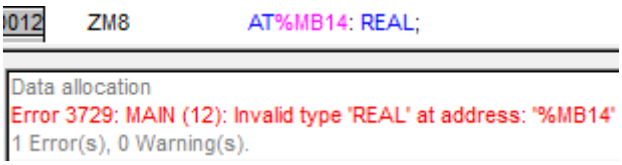
Adresowanie

W pokazanym wcześniej przykładzie przesyłana zmienna została zadeklarowana w pamięci jako AT %MB0, co oznacza że zostanie ona zapisana w pamięci zaczynając od zerowego bajtu. AT %MB4 oznaczałoby, że zmienna zostanie zapisana w pamięci zaczynając od piątego bajtu. Inne możliwe deklaracje to:

- AT %MX – umożliwia adresowanie pojedynczych bitów, np. AT %MX4.3 to 4 bit 5 bajtu,
- AT %MW – służy do adresowania zmiennych dwubajtowych,
- AT %MD – służy do adresowania zamiennych czterobajtowych.

W praktyce zasadne jest używanie jedynie zmiennych deklarowanych jako AT %MX oraz AT %MB, ponieważ ostatecznie wszystkie zmienne są umieszczane we wspólnym obszarze pamięci więc deklaracje AT %MB0, AT %MW0 i AT %MD0 dają identyczny rezultat.

Przy deklarowaniu zmiennych należy uważać, aby adres był podzielny przez rozmiar adresowanej zmiennej, np. dla zmiennej typu REAL dopuszczalne adresy to AT %MB0, AT %MB4, AT %MB8 itd. Podanie niedozwolonego adresu skutkuje błędem, przykład poniżej.



Nakładanie obszarów pamięci

Przydatną możliwością jest nakładanie jednej zmiennej na kilka innych i odczytywanie tym sposobem kilku zmiennych jednocześnie (za pomocą jednego bloku). Zostało to pokazane na rysunku poniżej. Na tablicę bajtów zostały nałożone różnego typu zmienne zmienne. Po stronie klienta i serwera układ zmiennych powinien być taki sam, wówczas rozkodowanie zmiennych odbędzie się samoczynnie.

```
0001 PROGRAM MAIN
0002 VAR
0003     ZM          AT%MB0: ARRAY[1..100] OF BYTE;
0004
0005     ZM1         AT%MB0: WORD;
0006     ZM2         AT%MB2: INT;
0007     ZM3         AT%MB4: REAL;
0008     ZM4         AT%MB8: BYTE;
0009     ZM5         AT%MB10: WORD;
0010     ZM6         AT%MB12: BYTE;
0011     ZM7         AT%MB13: BYTE;
0012     ZM8         AT%MB14: WORD;
```

W kliencie do bloku funkcyjnego odczytu/zapisu podpinamy tylko zmienną tablicową – ZM. Pozostałe zmienne uzupełnią swoje wartości samoczynnie. Jest to pokazane poniżej.