mgr inż. Dariusz Rzońca, mgr inż. Jan Sadolewski, dr inż. Andrzej Stec,
dr hab. inż. Zbigniew Świder, dr inż. Bartosz Trybus, prof. dr hab. inż. Leszek Trybus,
Politechnika Rzeszowska, Katedra Informatyki i Automatyki

PROGRAMOWANIE W JĘZYKU ST STEROWNIKA SMC LUMEL DLA MINISYSTEMU ROZPROSZONEGO

Przedstawiono sposób programowania w języku ST minisystemu kontrolno-pomiarowego złożonego ze sterownika SMC LUMEL, rozproszonych modułów wejścia/wyjścia oraz nadrzędnego komputera PC, za pomocą pakietu inżynierskiego CPDev. Do komunikacji służy protokół Modbus. Składnikami pakietu CPDev są kompilator, symulator i konfigurator zasobów sprzętowych. Przykład projektowy dotyczy sterowania prostą instalacją c.o.

PROGRAMMING AN SMC LUMEL CONTROLLER FOR A MINI-DISTRIBUTED SYSTEM IN ST LANGUAGE

A technique for programming control-and-measurement minisystem involving an SMC LUMEL controller, distributed I/O modules and PC host computer, using ST language is presented. ST is available in CPDev engineering package. Modbus communication protocol is used. CPDev consists of a compiler, simulator and configurer of hardware resources. Control of simple central heating installation is a design example.

1. WPROWADZENIE

Krajowy przemysł aparatury kontrolno-pomiarowej produkuje przetworniki pomiarowe, urządzenia wykonawcze, sterowniki, regulatory rejestratory, karty do PC itd. Po połączeniu w systemy rozproszone służą one do automatyzacji małej i średniej wielkości procesów. Narzędzia inżynierskie do programowania tych urządzeń są jednak dość ubogie i nie odpowiadają normie IEC 61131-3 obowiązującej w Polsce od 2004 r. [1]. Prace nad pakietem, który być może usunąłby to ograniczenie rozpoczęto w Politechnice Rzeszowskiej z końcem 2006 r.[1] Pakiet ten nazwano CPDev (*Control Program Developer*).

Podstawowymi założeniami postawionymi przed CPDev były uniwersalność i otwartość. Uniwersalność oznacza, że generowany kod może być wykorzystany na różnych platformach sprzętowych. Natomiast otwartość wymaga, aby dostępne były:

- narzędzia do tworzenia przez użytkownika własnych bloków funkcjonalnych i funkcji, będących jednostkami oprogramowania przeznaczonymi do wielokrotnego użycia.
- specyfikacje interfejsów wejścia/wyjścia i komunikacyjnych definiujące ogólne prototypy procedur obsługi interfejsów dla różnych platform i rozwiązań układowych.

Procedury te mogą oczywiście różnić się wewnętrzną treścią odpowiednio do platformy i rozwiązań układowych, ale formy ich wywołań pozostają identyczne. Niedawno we Francji opracowano pakiet IDE [2], który w założeniach jest trochę zbliżony do CPDev.

Obecnie pakiet CPDev umożliwia programowanie w tekstowych językach ST i IL [3, 4]. Skompilowany kod mogą wykonywać 8-bitowe mikrokontrolery AVR oraz 32-bitowe procesory komputerów PC. W mikrokontroler AVR jest wyposażony nowy sterownik SMC

[1] Praca realizowana w ramach projektu rozwojowego MNiSzW nr R02 058 03.

z LUMELu Zielona Góra (rys. 1), do programowania którego po raz pierwszy zastosowano CPDev. PC z kartami wejść/wyjść National Instruments lub InTeCo Kraków służy jako laboratoryjny *Soft Controller*.

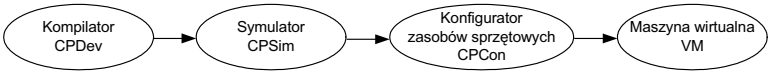
Rys. 1. Plansza LUMELu ze sterownikiem SMC i modułami serii SM na ubiegłorocznych Targach AUTOMATICON

Celem niniejszej pracy jest w miarę szczegółowe przedstawienie sposobu programowania sterownika SMC w języku ST dla minisystemu rozproszonego z modułami wejścia/wyjścia serii SM oraz nadrzędnym komputerem PC. Komunikacja jest prowadzona według protokołu Modbus. Jako obiekt sterowania służy prosta instalacja c.o.

2. STRUKTURA PAKIETU CPDev I JĘZYK ST

W skład pakietu CPDev wchodzą cztery programy pokazane na rys. 2. Po stronie komputera PC są to:
– kompilator CPDev języka ST,
– symulator oprogramowania CPSim,
– konfigurator zasobów sprzętowych CPCon.

Programy wymieniają dane w formie plików. Kompilator CPDev (jest to zarazem nazwa pakietu) generuje uniwersalny kod wykonywalny, który po stronie sterownika wykonuje maszyna wirtualna VM (*Virtual Machine*). Kod ten jest listą elementarnych instrukcji języka tej maszyny nazywanego asemblerem VMASM (*VM Assembler*). Nie jest on związany z żadnym procesorem, choć bliski nieco rozszerzonemu typowemu asemblerowi [5]. Maszyna wirtualna jest programem w języku ANSI C kompilowanym na daną platformę (np. AVR lub PC). Ogólnie biorąc przypomina to koncepcję maszyn wirtualnych Javy [6]. Symulator CPSim zawiera również maszynę wirtualną, która w tym przypadku funkcjonuje po stronie komputera PC.



Rys. 2. Struktura pakietu CPDev

Konfigurator CPCon specyfikuje typy i obszary pamięci, liczby i typy wejść/wyjść oraz kanałów komunikacyjnych, adresy fizyczne, sygnalizację poprawności/niepoprawności itp.

(część tych zadań wykonuje się automatycznie). Przydział zasobów sprzętowych ma postać *mapy* wiążącej adresy lokalne z adresami fizycznymi. Maszyna wirtualna platformy docelowej otrzymawszy kod i mapę przydziału zasobów jest w stanie prowadzić wymagane obliczenia.

Norma IEC 61131-3 definiuje pięć języków programowania – tekstowe ST, IL oraz graficzne LD, FBD i SFC [1]. Najwszechstronniejszy jest język ST (zbliżony do Pascala), który w CPDev służy jako język bazowy. Spośród 20 elementarnych typów danych najczęściej stosowane są `BOOL`, `INT`, `REAL` i `TIME`. Norma określa trzy zakresy dostępności zmiennych – `LOCAL`, `GLOBAL` i `ACCESS`. Zmienne `LOCAL` są dostępne tylko w ramach programu, bloku funkcjonalnego lub funkcji. Zmiennych `GLOBAL` można używać w całym projekcie z tym, że w programie muszą być zadeklarowane jako `EXTERNAL`. Zmienne `ACCESS` są wymieniane z innymi systemami (nie ma ich w CPDev).

Programy napisane w ST rozpoczynają się od deklaracji zmiennych i egzemplarzy bloków funkcjonalnych (instancji) zawartych między `VAR` a `END_VAR`. Po nich następuje lista instrukcji.

Język ST udostępnia pięć rodzajów instrukcji:
- podstawienie `:=` (symbol z Pascala),
- wybór `IF`, `CASE`,
- pętle `FOR`, `WHILE`, `REPEAT`,
- sterowanie `RETURN`, `EXIT`, `END`,
- wywołanie bloku funkcjonalnego.

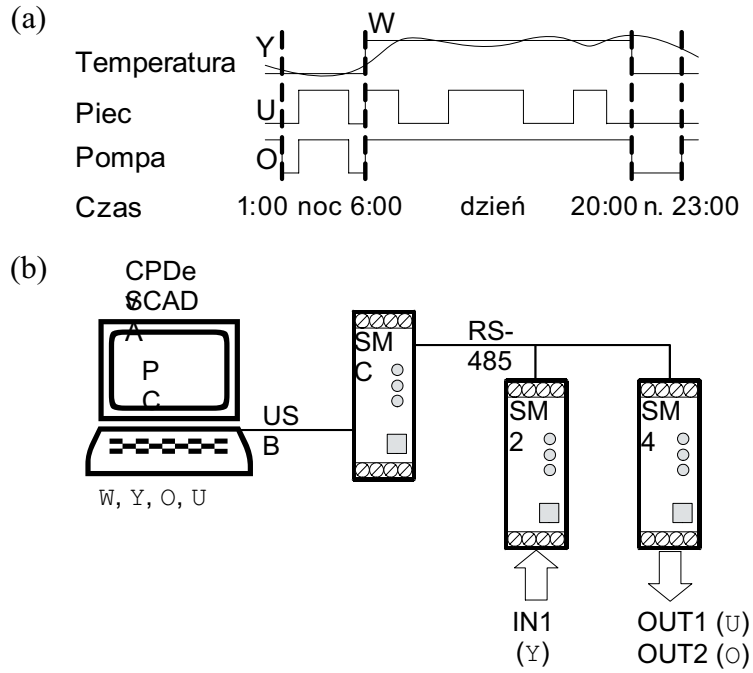
Pakiet CPDev zezwala na deklarowanie tablic, ale tylko jednoargumentowych. Dobrym źródłem dotyczącym programowania jest książka Kasprzyka [7].

3. PRZYKŁAD PROJEKTU Z KOMUNIKACJĄ I CZASEM

Zastosowanie pakietu CPDev do programowania minisystemu rozproszonego zostanie teraz przedstawione na przykładzie prostego projektu dotyczącego ogrzewania. Przypuśćmy, że temperaturę w hali należy utrzymywać na zadanej wartości `W`, wyższej w ciągu dnia, np. 22 °C, niższej w ciągu nocy, 18 °C. Faktyczną temperaturę `Y` mierzy moduł wejść analogowych, np. SM2 z LUMELu [8]. Jeżeli `W>Y`, piec C.O. jest włączany sygnałem sterującym `U` przez moduł wyjść binarnych SM4, a jeżeli `W≤Y`, piec jest wyłączany. W celu zapobieżenia zbyt częstym włączeniom ponowne włączenie pieca może nastąpić jednak dopiero wtedy, gdy `Y` spadnie poniżej `W` przynajmniej o 0,5 °C. Pompa obiegowa `O` tłocząca wodę przez grzejniki jest włączona stale w ciągu dnia, a w nocy wtedy, gdy piec pracuje, oraz niezależnie od pieca między godzinami 23.00 a 1.00. Jako dzień rozumie się okres od 6.00 do 20.00. Przykładowe przebiegi odpowiadające takiemu sterowaniu pokazano na rys. 3a.

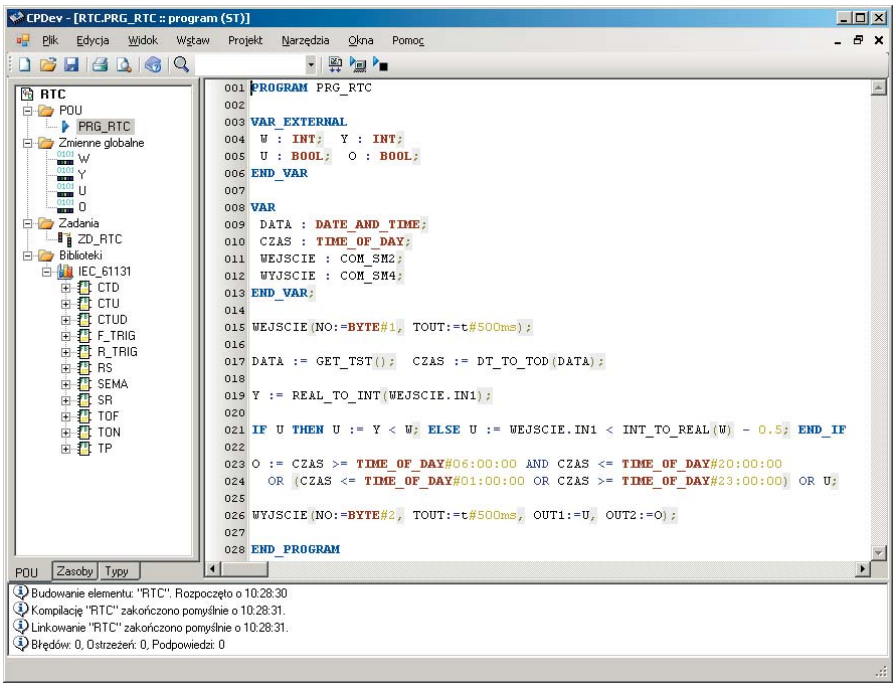
Przetwornik temperatury jest dołączony do wejścia IN1 modułu SM2, zaś styczniki włączające piec i pompę do wyjść OUT1, OUT2 modułu SM4. Zakłada się ponadto, że sterownik SMC komunikuje się z komputerem PC z zainstalowanym pakietem SCADA (np. InTouch, iFIX, LUMEL Control) w celu:
- zdalnego ustawiania zadanej temperatury `W`,
- obserwacji sygnałów `Y`, `U`, `O`, czyli temperatury mierzonej oraz sterowania piecem i pompą.

Wartości teemperatury transmitowane do/z komputera są typu `INT`. Minisystem rozproszony pokazany na rys. 3b mógłby służyć do realizacji takiego zadania, dalej nazywanego projektem `RTC`. Należy dodać, że moduł SM2 przekazuje do PC dane typu `REAL`, zaś SM4 odbiera dane typu `BOOL`.



Rys. 3. (a) Przykładowe przebiegi sterowania piecem i pompą; (b) realizacja przez minisystem rozproszony

Na rys. 4 pokazano główne okno interfejsu projektanta w pakiecie CPDev [3, 5] z projektem RTC. Drzewo po lewej stronie zawiera program PRG_RTC jako jednostkę POU (*Program Organization Unit*), cztery zmienne globalne W, Y, U, O, zadanie ZD_RTC oraz bibliotekę IEC_61131 dołączoną standardowo.



Rys. 4. Interfejs projektanta w pakiecie CPDev – projekt RTC

Program PRG_RTC widoczny w głównym polu jest napisany według reguł syntaktycznych języka ST [1, 7]. Pierwszą część stanowią deklaracje zmiennych globalnych (EXTERNAL), drugą deklaracje zmiennych lokalnych CZAS, DATA oraz instancji bloków komunikacyjnych

WEJSCIE: COM_SM2, WYJSCIE: COM_SM4, a trzecią instrukcje programu. Instrukcje w kolejnych liniach oznaczają:

15 – komunikacja z modułem SM2 za pomocą bloku WEJSCIE; moduł ma numer NO równy 1 (BYTE#1), a maksymalny czas *timeout* (TOUT) przeznaczony na komunikację wynosi 500ms.

17 – ustawienie zmiennej DATA na wartość zwracaną przez funkcję GET_TST, odczytującą stan zegara RTC sterownika w momencie rozpoczęcia zadania (*Get Task Time*). Wyodrębnienie CZASU dobowego przez funkcję konwersji DT_TO_TOD (*Date_and_Time To Time_of_Day*).

19 – konwersja wartości typu REAL odczytanej z wejścia IN1 modułu SM2 na zmienną Y typu INT.

21 – wyznaczenie sterowania U piecem na podstawie porównania temperatury mierzonej Y z zadaną W uwzględniając spadek temperatury po wyłączeniu przynajmniej o 0.5°C.

23, 24 – wyznaczenie sterowania O pompą obiegową - włączona w ciągu dnia, nocą w godzinach 23.00 ... 1.00 oraz wtedy, gdy piec pracuje.

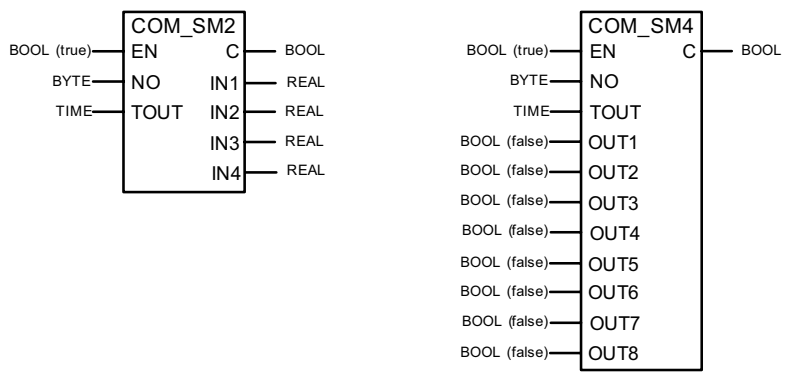
26 – komunikacja z modułem SM4 będącym urządzeniem nr 2, którego wyjścia OUT1, OUT2 są ustawiane przez sygnały U i O.

Potrzeba konwersji REAL_TO_INT (linia 21) bierze się stąd, że komputer przesyła do sterownika daną typu INT, a moduł SM2 daną typu REAL. Zadanie ZD_RTC, któremu przyporządkowano program PRG_RTC jest wykonywane cyklicznie co 200 ms (okno doboru cyklu pokazano np. w [3]).

4. KOMUNIKACJA Z MODUŁAMI ROZPROSZONYMI

W pakiecie CPDev dostępne są specjalne bloki dla komunikacji według protokołu Modbus z niektórymi urządzeniami LUMELu, w tym przede wszystkim z rozproszonymi modułami serii SM. Strukturę bloków COM_SM2, COM_SM4 wykorzystanych w projekcie RTC pokazano na rys. 5. Wejścia i wyjścia oznaczają:

EN – zezwolenie na komunikację (*Enable*); domyślnie TRUE.
NO – numer urządzenia w komunikacji Modbus; tylko wartość stała, np. BYTE#2.
TOUT – *timeout* (oczekiwanie na odpowiedź); tylko wartość stała, np. T#500ms.
C – poprawność/niepoprawność komunikacji.
IN1,...,IN4 – wyjścia analogowe (REAL) bloku COM_SM2.
OUT1,...,OUT8 – wejścia binarne (BOOL) bloku COM_SM4.

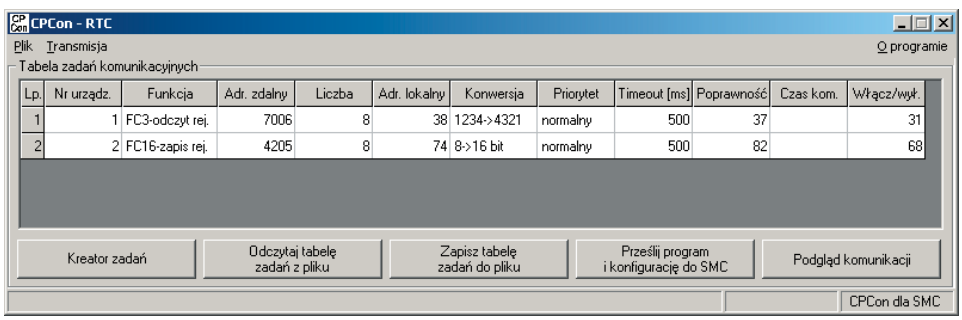


Rys. 5. Bloki komunikacyjne COM_SM2, COM_SM4

Typowym zastosowaniem wyjścia C jest przełączanie sterowania na wartość bezpieczną w razie przerwania komunikacji.

Bloki COM_SM odczytują lub zapisują wszystkie wejścia/wyjścia modułów w pojedynczych transakcjach pytanie↔odpowiedź lub polecenie↔potwierdzenie. Transakcje są tutaj nazywane *zadaniami komunikacyjnymi*. Deklaracja instancji bloku COM_SM automatycznie tworzy odpowiednie zadanie. Wystarczy tylko ustawić parametry transmisji w konfiguratorze CPCon, przesłać skompilowany projekt do sterownika SMC i włączyć urządzenia.

Wypełnioną automatycznie tabelę zadań komunikacyjnych konfiguratora CPCon dla projektu RTC pokazano na rys. 6. Parametry transmisji, tzn. prędkość, parzystość i bity stopu, ustawia się w zakładce *Transmisja* (nie otwieranej tutaj). Służy ona również do synchronizacji zegara sterownika z czasem komputera PC. Pierwszy wiersz tabeli reprezentuje blok WEJSCIA, a drugi WYJSCIA. *Numery urządzeń* i *timeout* pochodzą z wejść NO, TOUT w programie z rys. 4. *Adresy zdalne* są adresami pierwszego wejścia modułu SM2 i pierwszego wyjścia w SM4 [8]. *Liczba* jest liczbą rejestrów 2-bajtowych przy komunikacji rejestrowej lub liczbą bitów przy komunikacji bitowej. Moduł SM2 przesyła 4 liczby REAL, każdą w dwóch rejestrach, potrzeba więc na nie 8 rejestrów. Do modułu SM4 wysyła się 8 bitów, każdy w osobnym rejestrze. *Adresy lokalne* są wyrażonymi w bajtach adresami wyjścia IN1 bloku COM_SM2 i wejścia OUT1 w COM_SM4, które w skompilowanym programie odpowiadają adresom zdalnym. Dane REAL w module SM2 i sterowniku SMC są formatu IEEE-754, ale bajty ich zapisane są w odwrotnej kolejności (*Big Endian* w SM2, *Little Endian* w SMC). Konwersja 1234→4321 polega więc na przestawieniu bajtów. Konwersja 8→16 bit dla SM4 bierze się stąd, że w SMC zmienna BOOL zajmuje jeden bajt, a do SM4 jest transmitowana jako rejestr.



Rys. 6. Tabela zadań komunikacyjnych konfiguratora CPCon

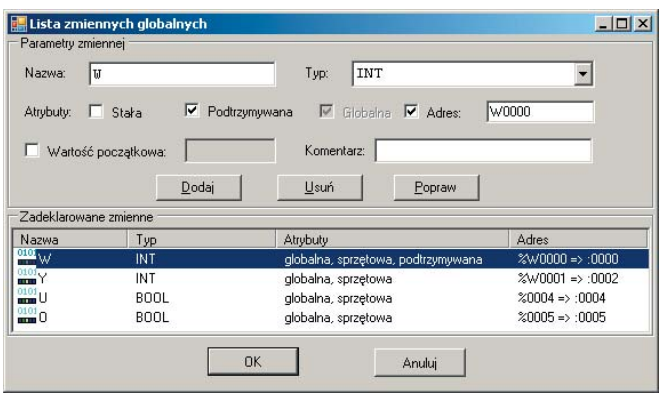
Zadaniu komunikacyjnemu można przyporządkować jeden z trzech *priorytetów*. Zadanie o priorytecie normalnym jest wykonywane dwukrotnie rzadziej niż zadanie o priorytecie wysokim, a zadanie o priorytecie niskim trzykrotnie rzadziej. *Poprawność* w tabeli zadań określa adres wyjścia C bloku COM_SM, a *Włącz/wyłącz* adres wejścia EN. *Czas komunikacji* (tutaj niewykorzystany) podaje adres zmiennej typu TIME, w której zostanie umieszczony czas wykonania zadania komunikacyjnego (tzn. czas wymiany komunikatów z danym urządzeniem). Czas ten można wykorzystać do diagnostyki komunikacji.

W przypadku bloków komunikacyjnych wiersze tabeli zadań są wypełniane automatycznie. Jeżeli jednak urządzenie zastosowane w systemie nie ma takiego bloku, to projektant wypełnia odpowiedni wiersz tabeli „komórka po komórce", bądź korzysta z interaktywnego kreatora zadań (przycisk na rys. 6). Kreator przechodzi przez kilka okien pytając o dane do tabeli zadań. Zmienne określające poprawność, czas i włączenie/wyłączenie komunikacji są potrzebne w zaawansowanych projektach.

5. KOMUNIKACJA Z KOMPUTEREM NADRZĘDNYM

Podczas komunikacji z komputerem PC sterownik SMC funkcjonuje jako *slave*. Parametry transmisji ustawia się również w zakładce *Transmisja* konfiguratora CPCon (rys. 6). Jednak skonfigurowanie komunikacji pakietu SCADA po stronie PC wymaga znajomości adresów zmiennych w sterowniku. Adresy te projektant może sam wpisać na liście zmiennych globalnych, a gdy tego nie uczyni, pakiet CPDev przydzieli je automatycznie.

Listę zmiennych globalnych projektu RTC z adresami wpisanymi przez projektanta poprzez komórkę *Adres* pokazano na rys. 7 (u góry). Kolumna *Adres* (na dole) zawiera po lewej stronie to co wpisał projektant według formatu IEC 61131-3, a po prawej stronie odpowiadający adres wyrażony w bajtach. Norma IEC wymaga, aby adresy poprzedzać przedrostkiem `%`, po którym następuje identyfikator rozmiaru, tzn. `B`, `X` lub jego brak dla zmiennych typu `BOOL` (brak na rys. 7), `W` dla `INT`, `D` dla `REAL` itd. [1, 7]. Jak widać zmienne `W`, `Y` zajmują po dwa bajty, a `U` i `O` po jednym. Gdyby lista zawierała zmienną typu `REAL`, przeznaczone byłyby na nią cztery bajty. Zmiennymi, którym projektant nie nadał adresów (kolumna *Adres* pozostaje wtedy pusta), kompilator CPDev przydziela adresy automatycznie, ale w taki sposób, że zmienne typu `INT` i inne 2-bajtowe mają adresy parzyste, a zmienne typu `REAL` i inne 4-bajtowe – adresy podzielne przez 4. Ewentualne puste obszary pomiędzy zmiennymi kompilator wypełnia w miarę możliwości nowo deklarowanymi zmiennymi.



Rys. 7. Lista zmiennych globalnych – projekt `RTC`

Warto jeszcze zwrócić uwagę, że zmiennej `W` przesyłanej z komputera nadrzędnego przypisano atrybut *podtrzymywana* (`RETAIN` w normie IEC). Spowoduje to, że w razie przerwy w zasilaniu sterownika SMC ostatnia wartość `W` zostanie przechowana w pamięci nieulotnej (RAM + bateria) i od niej zostaną wznowione obliczenia. Zmienne, którym nie przypisano atrybutu *podtrzymywana*, są inicjowane po wznowieniu zasilania.

Niezależnie od tego, czy projektant przydzielił, czy nie przydzielił adresy zmiennym globalnym, adresy bajtowe konieczne do skonfigurowania komunikacji z komputerem nadrzędnym są dostępne w raporcie projektu. Fragment raportu dotyczący adresów pokazano na rys. 8. W kolumnie *Adres* są powtórzone adresy z analogicznej kolumny listy zmiennych globalnych (rys. 7). Kolumna ModbusSMC zawiera adresy dla pakietów SCADA firmy LUMEL (Control, Ciepło). Przyjęto tam, że jeżeli w rejestrze jest przesyłana wartość typu `INT`, to adres bajtowy należy powiększyć o 4000 (*offset*). Pozwala to rozróżnić, czy w rejestrze jest przesyłany `BOOL` (bez *offsetu*), czy `INT`. Popularne pakiety SCADA podczas konfiguracji *drivera* komunikacyjnego stosują ponadto własny *offset*, który potem jest usuwany z komunikatów Modbusa. W InTouchu *offset* ten wynosi 40001, co widać w kolumnie Modbus InTouch raportu projektu.

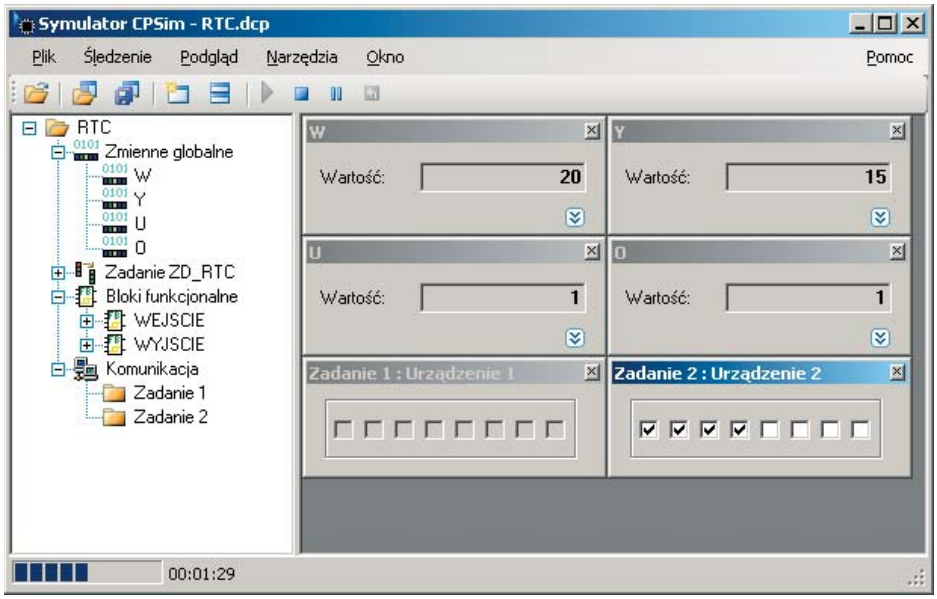
Lista zmiennych po kompilacji

	Nazwa	Pełna Nazwa	Adres	Rozmiar	Typ	Modbus SMC	Modbus INTOUCH
▶	W	RTC.W	0	2	INT	4000	44001
	Y	RTC.Y	2	2	INT	4001	44002
	U	RTC.U	4	1	BOOL	4	40005
	O	RTC.O	5	1	BOOL	5	40006

Rys. 8. Fragment raportu projektu z adresami zmiennych dla komputera PC

6. SYMULACJA PROJEKTU

Okno symulatora CPSim po skonfigurowaniu komunikacji pokazano na rys. 9. Po lewej stronie znajduje się drzewo zmiennych (zbliżone do drzewa projektu, rys. 4), a po prawej obszar podglądu z indywidualnymi panelami zmiennych `W`, `Y`, `U`, `O` oraz panelami grupowymi dwu zadań symulujących komunikację z modułami SM2 i SM4. Dobór paneli dla obszaru podglądu należy do użytkownika. Wartości zmiennych można ustawiać w obydwu typach paneli.



Rys. 9. Okno symulatora CPSim dla projektu RTC

Gdyby symulację przeprowadzić przed skonfigurowaniem komunikacji, to z drzewa zmiennych zniknęłaby *Komunikacja* wraz z dwoma zadaniami. Pozostaną jednak bloki `WEJSCIE`, `WYJSCIE` ze zmiennymi `IN1`...`IN4`, `OUT1`...`OUT8`, `EN` i `C` (rys. 5), z których można utworzyć panele grupowe i przeprowadzić symulację w niewiele węższym zakresie.

Warto jeszcze zwrócić uwagę, że temperatury `W`, `Y`, na których sterownik SMC prowadzi obliczenia i które wymienia z komputerem PC wyrażono w pełnych stopniach, jako liczby typu `INT` (np. 20 °C). Ogranicza to możliwość ustawiania `W` z PC do dokładności 1 °C. Dokładność na poziomie 0,01 °C można byłoby uzyskać operując na zmiennych `W`, `Y` 100-krotnie większych. Linia 19 programu z rys. 4 przyjęłaby wtedy postać:

```
Y := REAL_TO_INT(WEJSCIE.IN1 * 100.0);
```

Analogiczne mnożenia należałoby wprowadzić w linii 21.

7. PODSUMOWANIE

Przedstawiono sposób programowania minisystemu kontrolno-pomiarowego złożonego ze sterownika SMC i rozproszonych modułów SM z zielonogórskiego LUMELu oraz komputera PC. Programowanie przeprowadzono w języku ST za pomocą pakietu CPDev, biorąc jako przykład prostą instalację c.o. Komunikację SMC z modułami konfiguruje się korzystając z gotowych bloków komunikacyjnych, a gdyby ich nie było, przez skonfigurowanie komunikatów Modbusa za pomocą odpowiedniego kreatora. Adresy zmiennych „widziane" przez komputer PC określane są na podstawie adresów wpisanych na liście zmiennych globalnych, a gdy ich brak, pakiet CPDev określa je automatycznie. Adresy te są dostępne w raporcie projektu służąc do konfiguracji *drivera* komunikacyjnego po stronie PC. Pakiet CPDev udostępnia cztery typy danych związanych z czasem, spośród których `DATE_AND_TIME` i `TIME_OF_DAY` wystąpiły w przykładzie. Autorzy mają nadzieję, że pakiet CPDev nie będzie stwarzał większych trudności projektantom małych systemów rozproszonych opartych na krajowym sprzęcie.

BIBLIOGRAFIA

[1] PN-EN 61131-3 – Sterowniki programowalne. Część 3: Języki programowania. Warszawa, 2004.

[2] Tisserant E., Bessard L., de Sousa M.: An Open Source IEC 61131-3 Integrated Development Environment. 5th Int. Conf. Industrial Informatics, Piscataway (NJ, USA) 2007.

[3] Rzońca D., Sadolewski J., Stec A., Świder Z., Trybus B., Trybus L.: Mini-DCS System Programming in IEC 61131-3 Structured Text. JAMRIS, v. 2, no. 3, s. 48-54, 2008.

[4] Sadolewski J., Szmyd E.: Polski kompilator oraz sterowniki programowane za pomocą języka IL normy IEC 61131-3. Zgłoszony na konf. Automation 2009.

[5] Rzońca D., Sadolewski J., Stec A., Świder Z., Trybus B., Trybus L.: Środowisko programistyczne dla rozproszonych minisystemów kontrolno–pomiarowych. Metody Informatyki Stosowanej, tom 13, nr 1, s. 199-221, 2008.

[6] Lindholm T., Yellim F.: Java Virtual Machine Specification 2nd Ed., Java Software, Sun Microsystems Inc., 2004.

[7] Kasprzyk J.: Programowanie sterowników przemysłowych. WNT, Warszawa 2006.

[8] Lubuskie Zakłady Aparatów Elektrycznych LUMEL S.A., Zielona Góra – www.lumel.com.pl .

[9] Modicon MODBUS Protocol Reference Guide. Modicon Inc., Massachusetts 1996.