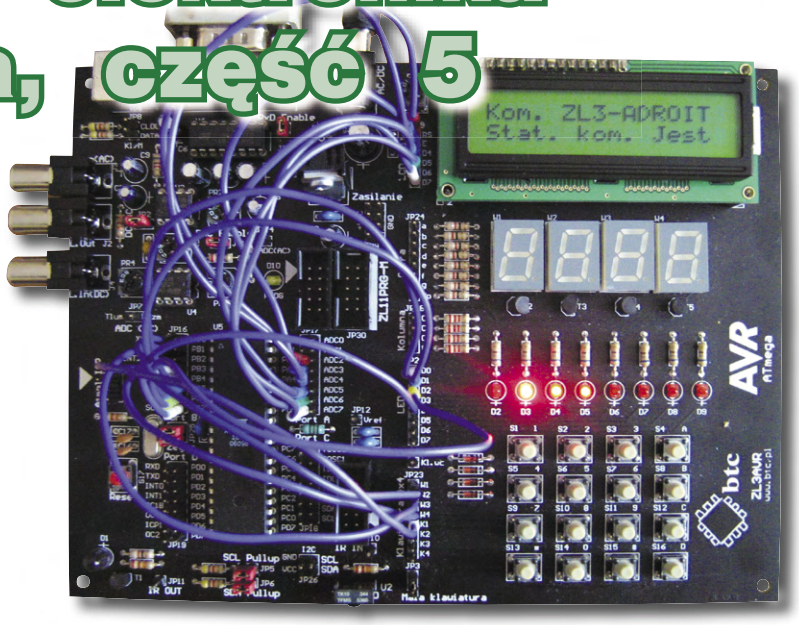
Oprogramowanie SCADA w praktyce elektronika–automatyka, część 5

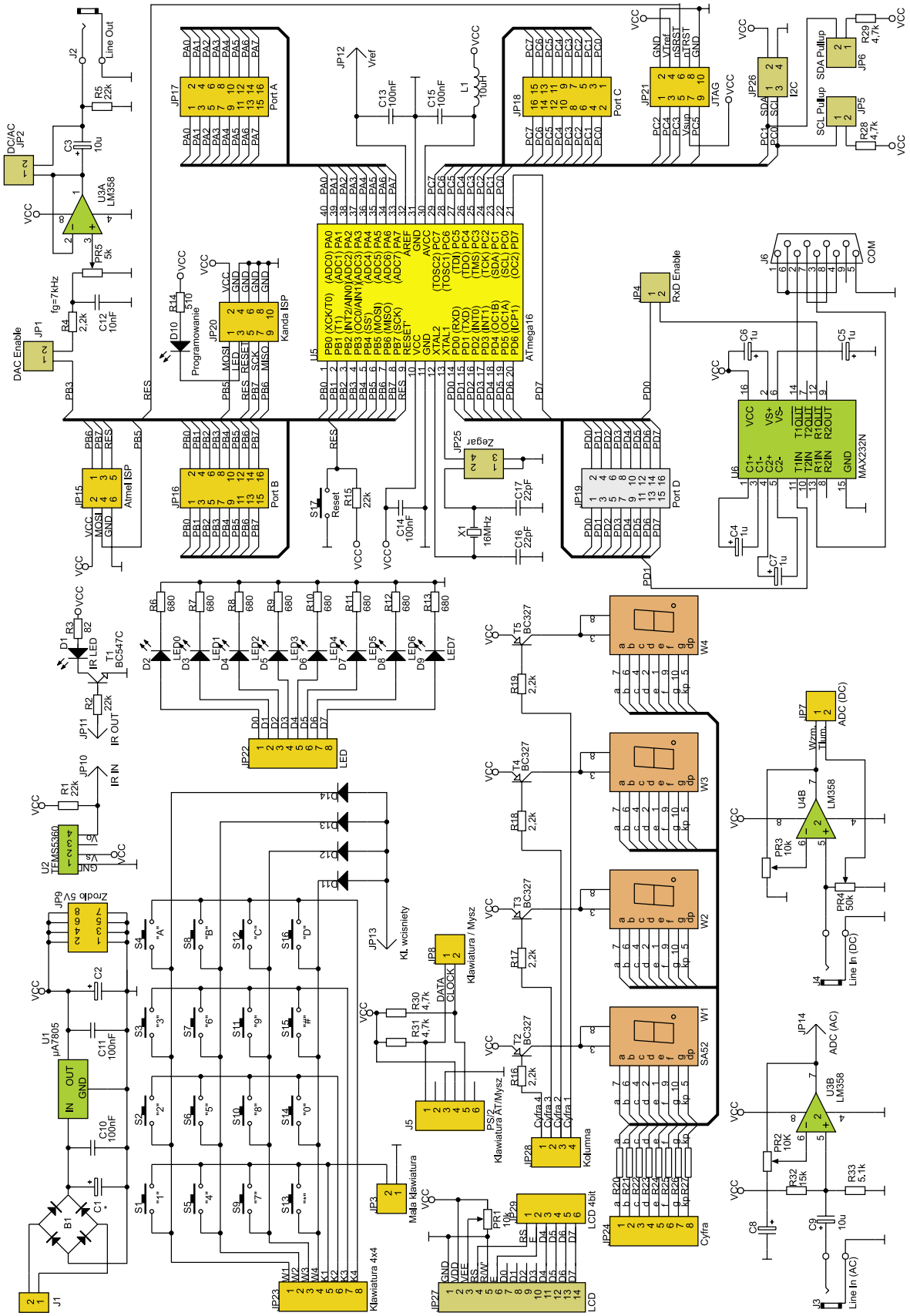


Elementy układów automatyki, nawet te najbardziej skomplikowane, składają się z mniejszych podzespołów na których są umieszczone elementy elektroniczne, zarówno te podstawowe (rezystory, kondensatory, itp.) jak i bardzo skomplikowane (układy cyfrowe, mikroprocesory). Obecnie moduły takie może zaprojektować prawie każdy elektronik. Kwestią dyskusyjną może się okazać opłacalność. Nie ma natomiast przeciwwskazań aby samemu zacząć eksperymenty z własnymi modułami automatyki. My zaczniemy od oprogramowania zestawu ZL3AVR z wykorzystaniem protokołu Modbus.

Zestaw startowy dla mikrokontrolerów AVR – ATmega

Praktyczną część kursu oprogramowania Adroit oparto o sterownik programowalny PLC serii NX7 firmy OEMax Controls oraz zestaw startowy ZL3AVR firmy Kamami. Zestaw ten został przygotowany dla czytelników książki „Mikrokontrolery AVR ATmega w praktyce", stanowi on sprzętową bazę dla wszystkich ćwiczeń opisanych w tej książce. Jest on również bardzo uniwersalny w budowie, dzięki czemu wykorzystano go w przykładowej aplikacji z oprogramowaniem Adroit.

Aby ułatwić Czytelnikom zrozumienie zasady działania oprogramowania, w poniższym przykładzie nie zostały wykorzystane wszystkie możliwości zestawu. Można oczywiście rozbudowywać oprogramowanie oraz część sprzętową o kolejne bloki funkcjonalne. Schemat elektryczny zestawu ZL3AVR z niezbędnymi połączeniami elektrycznymi w tej aplikacji przedstawiono na **rys. 1**. W przykładzie zostały wykorzystane następujące bloki: moduł zasilania, mikrokontroler ATmega32, układ taktujący

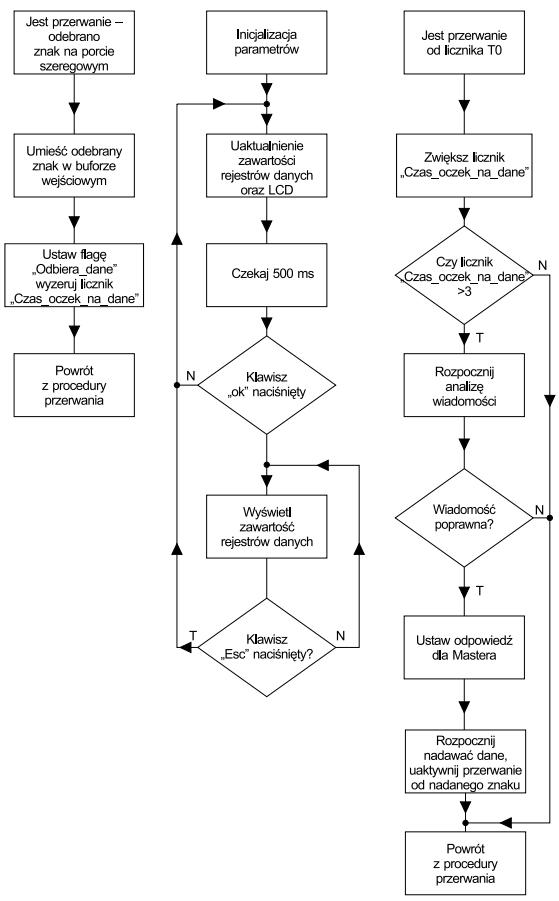


Rys. 1. Schemat elektryczny zestawu ZL3AVR, wykorzystanego do implementacji protokołu MODBUS w przykładowym projekcie. Połączenia do wykonania na schemacie elektrycznym zestawu ZL3AVR: Port A PA2 – JP29 RS, Port A PA3 – JP29 E, Port A PA4 – JP29 D4, Port A PA5 – JP29 D5, Port A PA6 – JP29 D6, Port A PA7 – JP29 D7, Port B PB4 – JP22 D0, Port B PB5 – JP22 D1, Port B PB6 – JP22 D2, Port B PB7 – JP22 D3, Port D PD4 – JP23 W1, Port D PD5 – JP23 W2, Port D PD6 – JP23 W3, Port D PD7 – JP23 W4, Zwora na JP3 (Mała klawiatura), Zwora na JP4 (RXD Enable)

(należy ustawić pracę z zewnętrznym rezonatorem kwarcowym 16 MHz), moduł klawiatury 4x1 (dodatkowo należy zewrzeć zworkę JP3), złącze JTAG, diody LED, alfanumeryczny wyświetlacz LCD, interfejs RS232. Połączenie z komputerem jest zrealizowane poprzez port RS232 oraz kabel połączeniowy 1:1. Aby uaktywnić linię RXD należy zewrzeć zworkę JP4. Do programowania mikrokontrolera został użyty programator JTAG ICE ale może to być również inne urządzenie kompatybilne ze standardem z ISP lub JTAG. Aplikacja z protokołem Modbus pozwalająca na komunikację z systemem SCADA została napisana w języku Basic pod kompilatorem BASCOM–AVR. Przed analizą algorytmu programu mikrokontrolera przybliżymy trochę teorii związanej z protokołem Modbus.

Protokół Modbus

Protokół komunikacyjny Modbus został opracowany w firmie Modicon. Jest on powszechnie stosowany w aplikacjach automatyki przemysłowej tam gdzie są niskie wymagania dotyczące szybkości i częstości transmisji danych. Jest on standardem przyjętym przez większość producentów sterowników przemysłowych. Modbus swą popularność zyskał dzięki prostocie zastosowanych w nim rozwiązań, jawności specyfikacji protokołu, a ponadto takim cechom jak: prosty dostęp do łącza na zasadzie „Master – Slave", zabezpieczenie komunikatów przed przekłamaniami, potwierdzenie wykonania rozkazów i sygnalizacji błędów oraz mechanizmom zapobiegającym zawieszaniu się systemu. Modbus jest najczęściej stosowany z interfejsem RS485. Kontrolery urządzeń pracujących w systemie MODBUS komunikują się ze sobą przy wykorzystaniu protokołu typu *master–slave*, w którym tylko jedno urządzenie może inicjować transakcje (jednostka nadrzędna – *master*), a pozostałe (jednostki podrzędne – *slave*) odpowiadają jedynie na zdalne zapytania *mastera*. Transakcja składa się z polecenia (*query*) wysyłanego z jednostki *master* do *slave* oraz z odpowiedzi (*response*) przesyłanej z jednostki *slave* do *mastera*. Odpowiedź zawiera dane żądane przez *master* lub potwierdzenie realizacji jego polecenia. *Master* może adresować indywidualnych odbiorców (jednostki *slave*) lub też przesyłać wiadomości „rozgłoszeniowe" (*broadcast*), przeznaczone dla wszystkich urządzeń podrzędnych w systemie. Na polecenia rozgłoszeniowe jednostki *slave* nie przesyłają odpowiedzi.



Rys. 2. Algorytm działania programu wykonywanego przez mikrokontroler

Ramka protokołu Modbus określa format przesyłanych wiadomości. Zawiera ona: adres odbiorcy, kod funkcji reprezentujący żądane polecenie, dane dotyczące funkcji oraz słowo kontrolne zabezpieczające przesyłaną wiadomość.

Odpowiedź urządzenia slave wysyłana jest również zgodnie z formatem zdefiniowanym w protokole MODBUS. Zawiera ona pole potwierdzenia realizacji rozkazu, dane żądane przez mastera oraz słowo kontrolne zabezpieczające odpowiedź przed błędami. Jeżeli urządzenie slave wykryje błąd przy odbiorze wiadomości, lub nie jest w stanie wykonać polecenia, przygotowuje specjalny komunikat o wystąpieniu błędu i przesyła go jako odpowiedź do mastera.

Protokół Modbus definiuje dwa tryby transmisji: ASCII (znakowy – każdy bajt w wiadomości przesyłany jest w postaci dwóch znaków ASCII), RTU (binarny – wiadomości rozpoczynają się odstępem czasowym trwającym minimum 3,5*TZ, gdzie TZ – czas trwania pojedynczego znaku oraz cała ramka musi zostać przesłana w postaci ciągłej tzn. odstęp pomiędzy kolejnymi znakami nie może być większy od 1,5*TZ).

Poszczególne znaki są przesyłane w formacie od najmłodszego do najstarszego bitu. Sposób ułożenia bitów w jednym bajcie w trybie RTU:

- 1 bit startu,
- 8 bitów pola danych, jako pierwszy wysyłany jest najmniej znaczący bit,
- 1 bit kontroli parzystości (nieparzystości) lub brak bitu kontroli parzystości,
- 1 bit stopu przy kontroli parzystości lub 2 bity stopu przy braku kontroli parzystości.

Kontrola parzystości jest dołączana do bajtu danych w urządzeniu nadającym. Urządzenie odbierające sprawdza bit parzystości znaku i na tej podstawie generuje odpowiedni bit kontrolny. W większości przypadków kontrola parzystości jest wyłączona, gdyż pewniejsze zabezpieczenie danych jest generowane przez sumę kontrolną CRC.

Typowe funkcje zdefiniowane w protokole Modbus:

- Read Coil Status „00" – odczyt bieżącego stanu grupy wyjść cyfrowych,
- Read Input Status „01" – odczyt stanu grupy wejść cyfrowych,
- Read Holding Register „03" – odczyt zawartości grupy rejestrów wyjściowych,
- Read Input Register „04" – odczyt zawartości grupy rejestrów wejściowych,
- Force Single Coil „05" – ustawienie stanu jednego wyjścia cyfrowego,
- Preset Single Register „06" – zapis do pojedynczego rejestru wyjściowego,
- Force Multiple Coils „0F" – ustawienie stanu grupy wyjść cyfrowych,
- Preset Multiple Register „10"– zapis do grupy rejestrów wyjściowych,

Generacja sumy kontrolnej

W protokole Modbus jako zabezpieczenie ramki wiadomości stosuje się sumę kontrolną. Jej wartość wyznacza urządzenie nadające dla zawartości przesyłanego komunikatu i umieszcza w ramce po części informacyjnej. Urządzenie odbiorcze oblicza sumę kontrolną dla odebranego komunikatu i porównuje jej wartość z wartością otrzymaną.

Niezgodność sum świadczy o wystąpieniu błędu.

Oprogramowanie mikrokontrolera

Aplikacja napisana w oprogramowaniu Bascom–AVR nie zawiera pełnej implementacji protokołu Modbus RTU. Obsługuje komendy związane z odczytem rejestrów danych (funkcja 03 i 04) oraz zapisem rejestrów (funkcja 06 i 10). Obsługa błędów została ograniczona od wysyłania odpowiedzi związanych z błędnym adresowaniem zakresu odczytu i zapisu. Ograniczenie to zostało wprowadzone tak aby można było lepiej zrozumieć zasadę funkcjonowania algorytmu. Rozszerzenie o pozostałe funkcje nie będzie stanowiło problemu. Program ten oraz aplikacja do oprogramowania Adroit została umieszczona na płycie CD–EP. Algorytm pracy mikrokontrolera przedstawiono na **rys. 2**. Zawiera on trzy gałęzie. Taki podział algorytmu został wprowadzony ze względu na to, że większość programu jest wykonywana w przerwaniach mikrokontrolera. Przerwania są generowane zarówno przez układy czasowe jak i układ transmisji szeregowej. Gałąź główna programu zajmuje się inicjalizacją układów peryferyjnych, układów liczników oraz transmisji szeregowej. W części głównej program zajmuje się obsługą wyświetlacza LCD, uaktualnianiem danych w obszarze rejestrów danych „Data_IO" oraz obsługą klawiatury. Po naciśnięciu klawisza „OK–S1", moduł przechodzi do trybu wyświetlania zawartości rejestrów danych. Klawisze „Up–S5" i „Dn–S9" zwiększają i zmniejszają numer wyświetlanego rejestru. Naciśnięcie klawisza „Esc–S13" spowoduje wyświetlenie ekranu podstawowego informującego o stanie komunikacji.

W mikrokontrolerze zostały uruchomione przerwania od układu licznikowego T0, który generuje przerwania co 1 ms. Wartość ta wynika z czasu transmisji 1 znaku przy prędkości 9600 b/s. Jeśli zostanie odebrany znak przez port szeregowy to w procedurze „Recv_danych" zostanie ustawiona flaga „Odbiera_dane" informująca, że są odbierane dane przez port szeregowy. W przerwaniu T0 zostaje również zwiększany licznik „Czas_oczek_na_dane", który jest zerowany również w przerwaniu od portu szeregowego. Jeśli zostanie zakończony proces odbioru danych to licznik ten będzie zwiększany do wartości 3, po przekroczeniu tej wartości wiadomość zostaje zakwalifikowana jako kompletna. Minimalna wartość przerwy wynosi 3,5 czasu trwania pojedynczego znaku, a więc dla prędkości 9600 b/s jest to wartość około 3,5 ms. Dla pewności została przyjęta wartość 4 ms. Następnie wiadomość jest sprawdzana w podprogramie „Jest_wiadom". Sprawdzane są poszczególne bajty wiadomości, a więc adres, suma kontrolna CRC i kod funkcji. Jeśli jakiś bajt wiadomości (kod, CRC) nie będzie pasował do oczekiwanego jest to traktowane jako błąd. W przypadku innego adresu procedura analizy wiadomości zostaje wstrzymana i program przechodzi do oczekiwania na następną wiadomość. Po prawidłowej weryfikacji polecenia następnie jest układana odpowiedź dla układu nadrzędnego. W zależności od polecenia jest wykonywany podprogram odpowiedzi na zapytanie o zawartość rejestrów „Odcz_rej_fun_3_4" lub odpowiedź na żądanie zapisu rejestrów „Zapis_rej_fun_16" lub „Zapis_rej_fun_6". Następnie jest wysyłana odpowiedź poprzez podprogram „Rozpocz_nadaw", który ładuje do bufora nadawczego pierwszy bajt wiadomości i jednocześnie ustawia flagę „Nadaje_dane". Pozostałe dane są wysyłane w procedurze przerwania „Trans_danych". Procedura ta zostanie zakończona jeśli licznik aktualnie wysłanych bajtów „I" będzie większy od „Licznik_bajt_transm". Następnie port szeregowy zostaje ustawiony w tryb odbioru danych. Oprócz przerwania od licznika T0 jest również uruchomione przerwanie od T1, który służy do zwiększania liczników systemowych sekund i minut oraz wyłącza diody LED.

Obszar rejestrów do odczytu przez zewnętrzny program Modbus Master rozpoczyna się od rejestru 1, a więc pierwszym rejestrem, który można adresować jest 40001 (według specyfikacji Modbus). Program jednakowo reaguje na funkcję odczytu „03" i „04". W programie liczba rejestrów została ustalona na 50. Pierwszych pięć rejestrów jest tylko do odczytu i zawierają one: licznik sekund, minut, licznik zapytań, licznik poprawnych zapytań, licznik błędnych zapytań. Pozostałe rejestry można odczytywać oraz zapisywać dowolnymi wartościami.

Sławomir Kacprzak
AT Control System

Oprogramowanie przygotowane przez autora artykułu opublikujemy ma CD–EP6/2007B oraz na naszej stronie internetowej, w dziale Download.