Dorota Rabczuk
Akademia Morska w Gdyni

PODSUMOWANIE DOŚWIADCZEŃ NAUCZANIA PROGRAMOWANIA MIKROKONTROLERÓW 8-BITOWYCH AVR W ŚRODOWISKU AVRSTUDIO

W artykule podsumowano doświadczenia z nauczania programowania mikrokontrolerów 8-bitowych AVR w układzie ewaluacyjnym EVB-503 PROPOX na platformie AVRStudio z kompilatorem języka C AVRGCC. Doświadczenia dotyczą m.in. wykorzystania symulacji środowiskowych oraz emulacji działania programu w czasie rzeczywistym po magistrali JTAG. Zwrócono uwagę na praktyczne aspekty pracy ze środowiskiem AVRStudio.

WSTĘP

Nauka programowania 8-bitowch mikrokontrolerów (na przykładzie rodziny AVR z firmy ATMEL) leży u podstaw szkolenia inżynierów na kierunku elektronika i telekomunikacja.

Mikrokontrolery 8-bitowe mają zaledwie zalążek systemu RTOS w postaci zestawu protokołów, do których programista może dodawać nowe, potrzebne w danej aplikacji. Ich specjalizacją są wbudowane interfejsy wielu magistral: UART, SPI, I2C, CAN, przetwornik ADC przy ograniczonych możliwościach obliczeniowych (do sumy i iloczynu liczb całkowitych 8-bitowych).

1. PRACA Z MIKROKONTROLEREM AVR W ŚRODOWISKU AVRSTUDIO

Rodzina AVR skupia 8-bitowe mikrokontrolery o architekturze RISC, wydajności 1 MIPS/MHz wykonujących większość rozkazów w jednym cyklu zegara kwarcowego o częstotliwości nieprzekraczającej 16 MHz.

Mikrokontrolery AVR mają rozdzielone przestrzenie pamięci: kod programu zapisywany jest w pamięci Flash programowalnej szeregowo, dane w pamięci ulotnej RAM o dostępnie rzędu ns lub nieulotnej pamięci EEPROM o dostępie rzędu ms.

Mikrokontrolery AVR obsługują wektory przerwań wewnętrznych (np. liczników/timerów) oraz przerwań zewnętrznych (np. sygnalizują stany linii wejściowych, raportują zdarzenia komunikacyjne na magistralach).

1.1. Platforma sprzętowa

Ćwiczenia są wykonywane w układzie ewaluacyjnym EVB-503 w wersji ADVANCED firmy PROPOX na trzech wymiennych procesorach: AT90S8515 (nowszy zamiennik ATMega8515), ATMega8, ATMega128. Procesory różnią się nieco wyposażeniem w moduły peryferyjne. ATMega8515 zaopatrzony jest w interfejs magistrali równoległej, na której w zewnętrznej przestrzeni adresowej umieszczony jest na płytce wyświetlacz LCD oraz zewnętrzna pamięć RAM. ATMega8 nie ma interfejsu magistrali równoległej (w konsekwencji LCD trzeba podłączać bezpośrednio do linii mikrokontrolera), za to ma wbudowany przetwornik ADC oraz interfejs magistrali I2C. ATMega128 jako jedyny z wymienionych zaopatrzony jest w interfejs JTAG, który umożliwia programowanie oraz debuggowanie w czasie rzeczywistym. Zamiana procesorów na płytce ewaluacyjnej poszerza liczbę możliwych do zaproponowania ćwiczeń laboratoryjnych.

Zestaw ewaluacyjny EVB-503 w wersji ADVANCED składa się z urządzeń zewnętrznych do podłączenia na liniach magistral mikrokontrolera, w tym z dwóch zegarów czasu rzeczywistego na magistralach SPI i I2C, zewnętrznej pamięci EEPROM na magistrali I2C, wyprowadzonego złącza magistrali 1-wire do podłączenia termometru cyfrowego lub pamięci, buczka piezoelektrycznego. Na liniach interfejsu USART znajduje się konwerter zmiany poziomów napięć ze standardu TTL na RS-232 umożliwiający bezpośrednią komunikację w portem COM komputera PC. W zestawie brakuje wyjścia na złącze kart pamięci SD/MMC oraz CF, które trzeba samodzielnie dolutować na karcie rozszerzeń.

1.2. Cechy środowiska AVRStudio ATMEL

Producent mikrokontrolerów AVR – firma ATMEL udostępnia dla swoich produktów język assembler oraz środowisko AVRStudio (na licencji freeware) do pisania programów w tym języku. Środowisko ma edytor i debugger – tworzy kod wynikowy i umożliwia symulacje środowiskowe. Platforma AVRStudio współpracuje z kompilatorem języka C AVRGCC dostępnym również na licencji freeware w pakiecie WINAVR. Zainstalowanie obu środowisk zapewnia pełne narzędzie do pisania programów w języku C, ich kompilacji i debuggowania. W zintegrowanym środowisku opcje kompilacji można ustawiać w programie okienkowym lub edytować plik Makefile w notatniku. Należy tu zwrócić uwagę, że wybranie zbyt dużego stopnia optymalizacji długości kodu wynikowego skutkuje niewykonywaniem przez program niektórych pętli. Efekt ten jest widoczny nie tylko po załadowaniu programu do pamięci Flash, ale również w trybie symulacji środowiskowych działania programu. W takich przypadkach zmniejszenie poziomu optymalizacji do wartości 1 lub całkowita rezygnacja z optymalizacji (poziom 0) zapewniają prawidłowe wykonanie kodu.

1.3. Assembler kontra język C

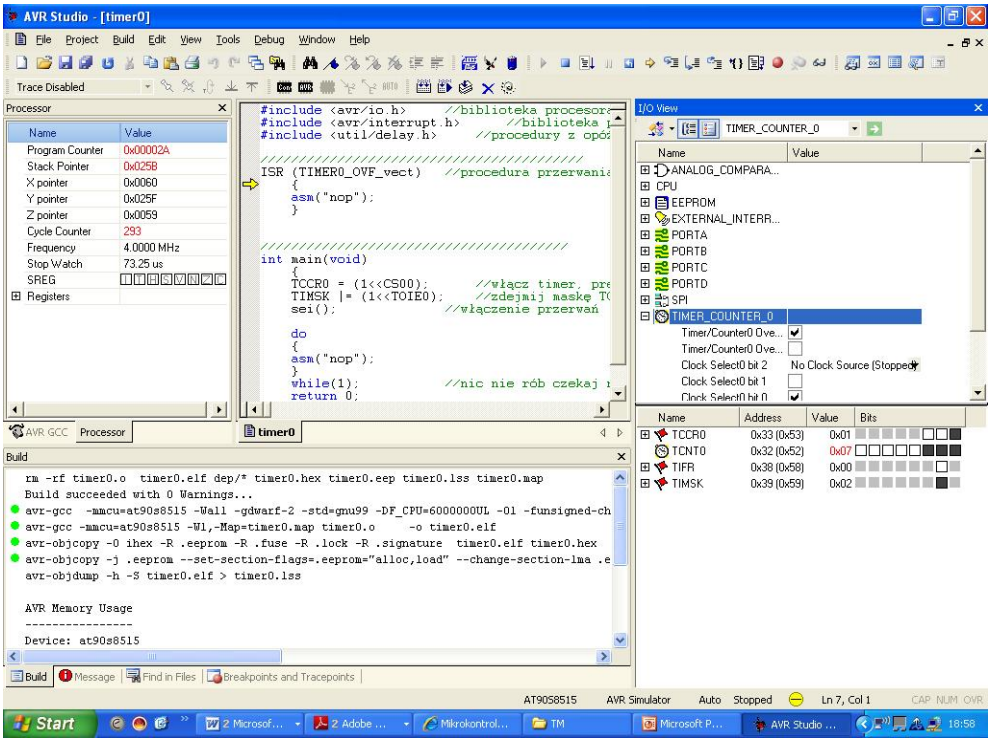
Mikrokontrolery 8-bitowe AVR można z powodzeniem programować zarówno w języku assembler, jak i w języku C. Język assembler daje programiście panowanie nad czasem z dokładnością do 1 cyklu zegarowego, ale nie zapewnia przejrzystości kodu w programach, które mają wiele odwołań warunkowych i nie umożliwia korzystania z gotowych procedur dostarczanych przez wielu producentów oprogramowania głównie w języku C. Z tego powodu praca ze środowiskiem w języku assembler (zakładanie projektu, pisanie kodu, assemblacja, debuggowanie) w laboratorium jest ograniczona do trzech reprezentatywnych projektów. Projekty assemblerowe ilustrują precyzyjne odliczanie czasu w pętli, zapis/odczyt wewnętrznej pamięci EEPROM – proces wymagający reżimu czasowego z dokładnością do czterech cykli zegara systemowego oraz programową obsługę interfejsu I2C z optymalizacją prędkości transmisji po magistrali I2C (istotne dla procesorów nieposiadających interfejsu I2C). Pozostałe zagadnienia są realizowane w projektach zakładanych w języku C pod kompilatorem AVRGCC. Student przystępujący do laboratorium musi mieć określoną wiedzę z programowania w języku C. W szczególności istotne i często wykorzystywane są następujące elementy języka C:

- instrukcje warunkowe *if else,* pętle: *for, while, do while, switch case*, w tym pętla oczekująca na ustawienie się flagi o konstrukcji *while* (*!flaga*);
- arytmetyka binarna (suma |, iloczyn &, negacja ~ i przesunięcie binarne <<) do selektywnego ustawiania/zerowania wybranych bitów w rejestrach;
- umiejętność definiowania funkcji pobierających parametry i zwracających wynik;
- umiejętność tworzenia plików nagłówkowych w celu uporządkowania rozbudowanych programów.

1.4. Nauczanie w trybie symulacji środowiskowych

Tryb symulacji środowiskowych przydaje się szczególnie w początkowej fazie nauczania programowania mikrokontrolerów [2]. Pierwsze ćwiczenia obejmują konfigurację linii portów, sterowanie urządzeniami wyjściowymi (ON/OFF) oraz odczyt linii wejściowych. Kolejnym nauczanym zagadnieniem są pojęcia wektorów przerwań zewnętrznych i wewnętrznych, aktywacja związanych z nimi procedur przerwań przez operacje na maskach przerwań.

Debuggowanie środowiskowe (w trybie symulacji) umożliwia wykonanie kodu programu krok po kroku. W trakcie debuggowania symulacyjnego rozkaz aktualnie wykonywany jest wskazywany strzałką w okienku kodu źródłowego napisanego w języku C (a nie w okienku dissasemblera), co znakomicie ułatwia orientację początkującym programistom. Ponadto debuggowanie w trybie symulacji zapewnia automatyczny podgląd (symulacyjny) zawartości rejestrów mikrokontrolera zaangażowanych w wykonanie kodu. Symulacyjny tryb debuggowania nie zawsze precyzyjnie odpowiada rzeczywistości, za to ze względu na przejrzysty i poglądowy sposób prezentacji działania programu ma walory dydaktyczne.



Rys.1. Okno debuggowania – przykład symulacji środowiskowych

1.5. Ładowanie programu wykonywalnego

Mikrokontroler AVR akceptuje kod programu w postaci pliku *.hex. Kod może zostać załadowany po magistrali SPI lub JTAG (w zależności od modułów peryferyjnych mikrokontrolera) przy wykorzystaniu odpowiedniego programatora. Środowisko AVRStudio ma odpowiednie oprogramowanie narzędziowe do ładowania programów wykonywalnych po obu magistralach.

Alternatywnym wygodnym sposobem ładowania programów jest wykorzystanie programu ładującego *bootloader* rezydującego w dolnej części pamięci programu Flash. Miejsce dla programu *bootloader* musi zostać zarezerwowane w pamięci Flash przez zaprogramowanie odpowiednich bitów konfiguracyjnych mikrokontrolera, a ładowanie programu *bootloader* odbywa się po magistrali JTAG lub ISP. Napisanie kodu programu *bootloader* na podstawie noty z firmy ATMEL jest doskonałym ćwiczeniem uzupełniającym. Po załadowaniu programu *bootloader* należy się z nim połączyć po magistrali UART (interfejs stanowi program narzędziowy środowiska AVRStudio) i wskazać nowy kod wykonywalny który należy załadować do pamięci Flash od adresu 0. Rozwinięciem ćwiczenia może być samodzielne stworzenie interfejsu na komputerze PC (np. w środowisku C++ Builder) do łączenia się z programem *bootloader*. Obszerność tego ostatniego za-

dania i konieczność wykorzystania dwóch różnych środowisk do jego wykonania wskazują, że zadanie to powinno być realizowane na zajęciach projektowych, a nie laboratoryjnych.

1.6. Nauczanie w trybie emulacji w czasie rzeczywistym

Emulacja w czasie rzeczywistym działa po magistrali JTAG i może być wykonana tylko w mikrokontrolerze wyposażonym w ten interfejs, np. ATMega128. Tryb emulacji w czasie rzeczywistym przydaje się szczególnie w trakcie uruchamiania programów z komunikacją po magistralach, np. UART, SPI lub I2C [1]. Jeśli wykonanie krokowe programu jest zbyt długie i nieefektywne, to można wykorzystać opcję „run" i należy założyć pułapki „breakpoints", w których zatrzymuje się wykonanie programu. Równocześnie należy założyć podgląd rejestrów interfejsu komunikacyjnego, który zadziała, gdy wykonanie programu zatrzyma się w linii „pułapki". Pozwala to sprawdzić, czy:

- program został wykonany aż do oczekiwanego punktu,
- w rejestrze interfejsu komunikacyjnego pojawiły się oczekiwane dane.

PODSUMOWANIE

Ćwiczenia powinny łączyć cechy zajęć projektowych i laboratoryjnych, i być poprzedzone przyswojeniem wiedzy teoretycznej. Ćwiczący przygotowują algorytm programu (który typowo składa się z konfiguracji interfejsów oraz zdefiniowania akcji w pętli głównej) i wypełniają go treścią, pisząc odpowiednie procedury. Po kompilacji sprawdzają wybrane fragmenty w trybie symulacji środowiskowych lub emulacji w czasie rzeczywistym. Efektywność nauczania zależy od liczby i różnorodności wykonanych ćwiczeń.

Ćwiczący mają obowiązek przechowywania samodzielnie napisanych kodów źródłowych, powiększając w ten sposób własne zasoby biblioteczne. Podczas przygotowywania rozbudowanych projektów możliwość korzystania z własnych plików nagłówkowych skraca wydatnie czas pisania kodu.

Naturalnym kierunkiem rozwoju tematyki zajęć jest wprowadzenie projektów z interfejsami ostatnio implementowanymi w mikrokontrolerach (nowych rodzin), takich jak Ethernet i ZigBee, oraz wprowadzenie projektów wielozadaniowych częściowo wykonywanych przez studenta w ramach pracy domowej i tylko sprawdzanych w trybie laboratoryjnych. Samodzielna praca nad projektem pozwala bowiem ugruntować wiedzę i nabrać w niej pewności, powinna być zatem elementem każdego szkolenia.

LITERATURA

1. Doliński J., *Mikrokontrolery AVR w praktyce,* BTC, Warszawa 2004.
2. Rabczuk D., *Technika mikroprocesorowa – ćwiczenia laboratoryjne,* Akademia Morska w Gdyni, Gdynia 2009.

SUMMARY OF EXPERIENCE IN TEACHING PROGRAMMING OF 8-BIT AVR MICROCONTROLLERS ON AVRSTUDIO PLATFORM

Summary

The article summarizes the author's experience in teaching programming of 8-bit AVR microcontrollers in EVB-503 evaluation board from PROPOX in AVRStudio programming environment with AVRGCC C language compiler. The role of debugging programs in laboratory exercises is discussed: on AVR Simulator platform and on JTAG emulation platform. Practical aspects of working with AVRStudio are emphasized.