XX Jubileuszowe Seminarium

ZASTOSOWANIE KOMPUTERÓW W NAUCE I TECHNICE' 2010
Oddział Gdański PTETiS
Referat nr 10

GRAFICZNE PROGRAMOWANIE MIKROKONTROLERÓW W ŚRODOWISKU REALIZER

Krzysztof GÓRSKI[1], Krystyna Maria NOGA[2]

1. student AM w Gdyni, Wydział Elektryczny
   tel: (0 55) 2302538, e-mail: krzysztof.gorski@ep.com.pl
2. Akademia Morska, Katedra Automatyki Okrętowej, ul. Morska 81-87
   81-225 Gdynia, tel: (058) 6901471, e-mail: jagat@am.gdynia.pl

Streszczenie: Artykuł zawiera krótki opis programu Realizer, na bazie którego autorzy zaproponowali zestaw ćwiczeń dydaktycznych do wykorzystania w laboratorium techniki cyfrowej. W artykule przedstawiono kilka praktycznych przykładów zastosowań pakietu Realizer do graficznego programowania mikrokontrolerów rodziny ST62, ST72, ST7.

Słowa kluczowe: programowanie, mikrokontrolery, technika cyfrowa.

1. WSTĘP

Postęp techniki sprawia, że mikrokontrolery są coraz częściej stosowane w wielu urządzeniach elektronicznych. W związku z tym rosną wymagania stawiane współczesnemu inżynierowi, który powinien znać przynajmniej podstawy jednego z języków programowania mikro-kontrolerów. Do najbardziej popularnych języków programowania należą język C oraz asembler. Dobre opanowanie każdego z tych języków wymaga dużej ilości czasu, którego coraz częściej brakuje. Przyjaznym rozwiązaniem, zapewniającym szybkie pisanie programów dla mikrokontrolerów, są programy graficzne. Obecnie programowanie mikrokontrolerów w środowisku graficznym nie cieszy się zbyt dużą popularnością. Nawet firmy produkujące mikrokontrolery podchodzą do tego zagadnienia dość sceptycznie. Jednak co pewien czas na rynku informatycznym pojawiają się programy umożliwiające tworzenie oprogramowania mikrokontrolerów w edytorze graficznym, tj. przy pomocy schematu lub grafu. Jednym z takich przykładów jest pakiet Realizer, którego producentem jest firma Actum [1]. Możliwość tworzenia oprogramowania mikrokontrolerów bez znajomości języka wyższego rzędu oraz prosta i intuicyjna obsługa pakietu Realizer sprawia, że może on być niezwykle przydatny w dydaktyce podstaw techniki mikroprocesorowej. Pakiet ten ułatwia niezbyt doświadczonemu informatykowi poznanie świata techniki mikroprocesorowej. Praktycznie każdy posiadający podstawową wiedzę z zakresu techniki cyfrowej nie powinien mieć trudności z tworzeniem oprogramowania w tym środowisku. Umiejętności nabyte podczas pracy z tego typu programami ułatwiają dalsze zgłębianie techniki mikroprocesorowej.

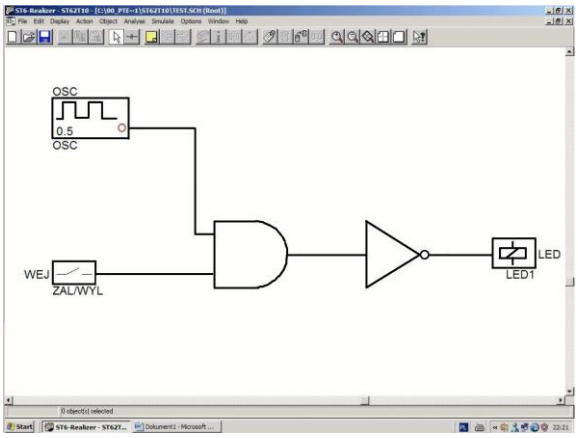
Artykuł zawiera krótki opis programu Realizer 2.1 [2, 3], na bazie którego autorzy zaproponowali zestaw ćwiczeń dydaktycznych do wykorzystania w laboratorium techniki cyfrowej. W artykule przedstawiono kilka praktycznych przykładów zastosowań tego programu do graficznego programowania mikrokontrolerów rodziny ST62, ST72, ST7.

2. ŚRODOWISKO PROGRAMOWE REALIZER

Środowisko Realizer składa się z czterech programów, które są niezbędne w procesie tworzenia oprogramowania dla mikrokontrolera. Są to:

- Program główny Realizer,
- Symulator,
- Analyser,
- Sch2Lib Converter.

Realizer - jest głównym programem, na bazie którego przeprowadza się edycję projektowanego oprogramowania oraz uruchamia się pozostałe programy wchodzące w skład środowiska (rys. 1).



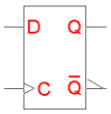
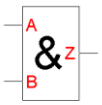
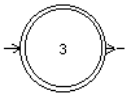
Rys. 1. Okno główne (pole edycyjne) programu Realizer

Symulator - umożliwia, po poprawnie przeprowadzonej kompilacji, symulację, czyli uzyskanie przebiegów czasowych. Analyser - generuje szesnastkowy plik wynikowy oraz raporty o błędach. Natomiast Sch2Lib – Converter jest przeznaczony do konwersji schematów do postaci modułów bibliotecznych. Realizer zawiera bogaty zestaw elementów bibliotecznych (Tab. 1), które możemy podzielić na dwie grupy, tj. symbole logiczne oraz moduły funkcjonalne. W grupie symboli logicznych zawarte są wszystkie podstawowe elementy logiczne, czyli bramki AND, NAND, NOT, EXOR, OR i NOR. Natomiast w grupie modułów funkcjonalnych zawarte są komparatory, liczniki rewersyjne, bloki

Recenzent: Prof dr hab. inż. Krzysztof Karwowski – Wydział Elektrotechniki i Automatyki Politechnika Gdańska

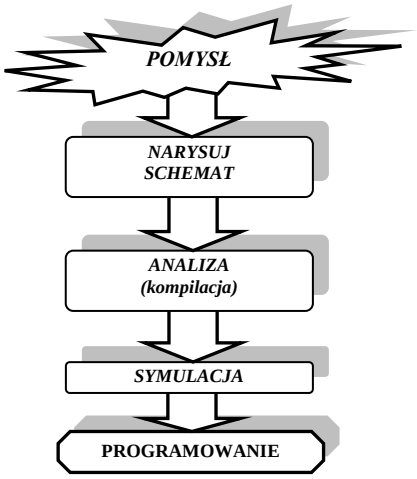
komutacyjne (multipleksery, demultipleksery), układy arytmetyczne (sumatory, multiplikatory, subtraktory), stałe (bit i słowo), przerzutniki (JK, D, RS), wejścia cyfrowe i analogowe, detektory zmian wartości zmiennych, przekaźniki czasowe oraz generatory fali prostokątnej. Ponadto środowisko Realizer pozwala użytkownikowi na tworzenie własnych elementów bibliotecznych.

Tab.1. Przykładowe elementy biblioteczne

Symbol	Nazwa	Opis
	DFF	Przerzutnik typu D
	ADC	8-bitowy przetwornik A/C
	AND2, AND3, AND4, AND6, AND8,	bramki typu iloczyn logiczny
	STATEINIT	rozpoczęcie programu (Initial state)

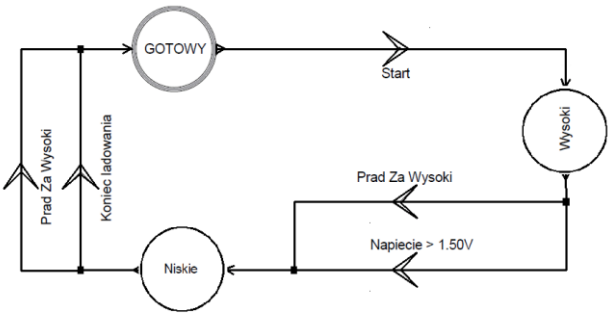
3. TWORZENIE PROGRAMU GRAFICZNEGO

Proces tworzenia oprogramowania dla mikrokontrolera w środowisku Realizer podzielić możemy na pięć podstawowych etapów, tj. tworzenie koncepcji, rysowanie schematu, analiza (generowanie plików wynikowych), testowanie układu (symulacja) oraz programowanie mikrokontrolera (rys. 2). Cykl tworzenia oprogramowania w środowisku Realizer nie wymaga zbyt dużo czasu. Niewielki program można napisać w ciągu kilku minut. Przed przystąpieniem do pracy należy jasno określić rolę programu. Na tym etapie prac należy wybrać typ mikrokontrolera tak, aby możliwości Realizera i mikrokontrolera sprostały wymaganiom postawionym projektowanemu urządzeniu. Wersja 2.1 Realizera obsługuje mikrokontrolery rodziny ST62. Rysowanie schematu polega na umieszczaniu w polu edycyjnym elementów bibliotecznych oraz na wykonaniu odpowiednich połączeń między elementami.



Rys. 2. Proces tworzenia programu

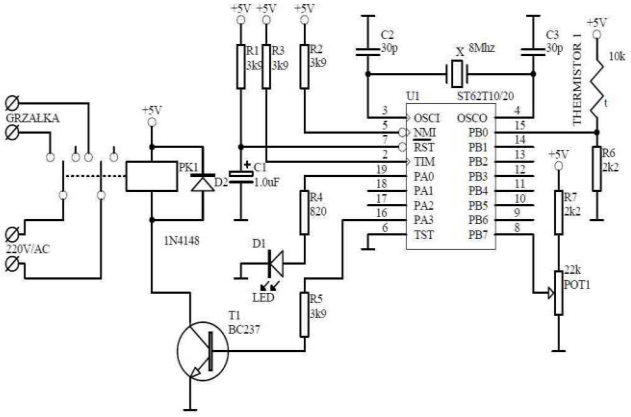
W omawianej wersji programu Realizer wymagane jest przygotowanie programu przy pomocy grafu przejść, który jest odpowiednikiem algorytmu. Graf jest budowany ze specjalnych elementów, tj. STAN POCZĄTKOWY (STATEINIT), WARUNEK (CONDITION), STAN-PRACY (STATE), które znajdują się w bibliotece Realizera. Na rysunku 3 został przedstawiony przykład grafu układu, którego zadaniem jest ładowanie ogniwa akumulatora NiCd. Po poprawnym narysowaniu schematu należy wejścia DIGOUT, ADC i wyjścia DIGINPUT przypisać fizycznym wyprowadzeniom mikrokontrolera. Następnie projekt należy poddać kompilacji, co uzyskujemy uruchamiając program Analyser. W wyniku końcowym, podobnie jak podczas programowania w języku niższego oraz wyższego poziomu, uzyskujemy plik wynikowy HEX, który jest wykorzystywany do zaprogramowania mikrokontrolera. Przed jego zaprogramowaniem projekt można poddać symulacji. Powstaje wówczas plik o rozszerzeniu *sef*, w którym do schematu dołączane są moduły pomiarowe. Moduły te wymuszają określone przebiegi czasowe na wejściach oraz umożliwiają obserwację sygnałów wyjściowych. Programowanie mikrokontrolera jest ostatnim etapem pracy nad projektem. Polega ono na umieszczeniu mikrokontrolera w programatorze oraz zaprogramowaniu go przy użyciu programu Windows Epromer [7].



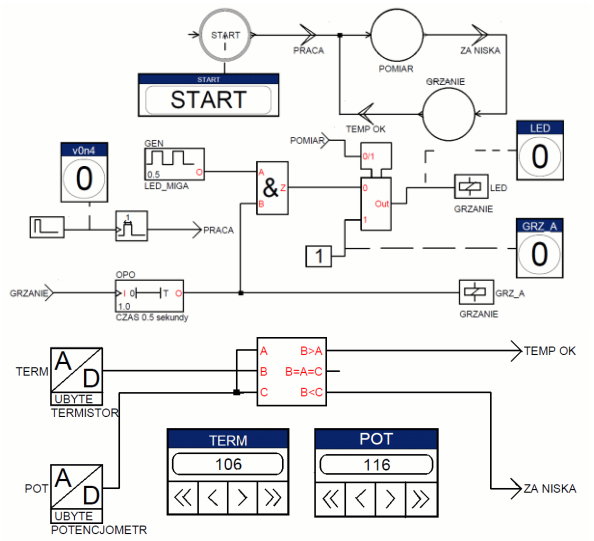
Rys. 3. Graf przejść układu ładującego akumulatory NiCd

3.1. Przykładowy program

Przykładowe oprogramowanie, przygotowane w środowisku Realizer, wykonano dla mikrokontrolera ST62T10. Projekt przedstawia prosty regulator (rys. 4), który z powodzeniem można wykorzystać do regulacji temperatury, np. w akwarium. Pomiar temperatury dokonywany jest za pomocą sondy z termistorem NTC 10k. Załączanie elementu grzejnego zostało zrealizowane przy użyciu przekaźnika PK1. Stan pracy regulatora sygnalizuje dioda LED D1. Wartość wymaganej temperatury jest zadawana potencjometrem POT1. Po włączeniu zasilania program mikrokontrolera realizuje pierwszy stan pracy, tj. POMIAR (rys. 5).



Rys. 4. Schemat ideowy regulatora temperatury



Rys. 5. Schemat programu regulatora temperatury

Gdy wartość temperatury mierzonej jest niższa od zadanej, czyli spełniony jest warunek ZA_NISKA, program przechodzi w stan pracy GRZANIE. W tym stanie pracy zostaje załączony przekaźnik PK1, którego styki załączają zasilanie grzałki oraz zaczyna pulsować dioda D1. Program znajduje się w stanie GRZANIE tak długo, aż zostanie spełniony warunek TEMP_OK, po czym ponownie przechodzi w stan POMIAR.

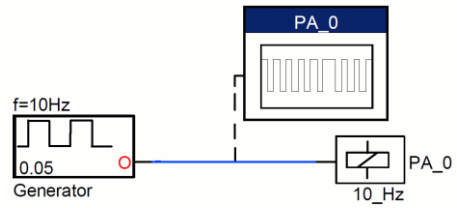
4. PROPONOWANY ZESTAW ĆWCZEŃ

Dla celów edukacyjnych autorzy zaproponowali zestaw ćwiczeń do wykorzystania w laboratorium techniki cyfrowej. Podczas wykonywania każdego ćwiczenia studenci powinni utworzyć pliki projektu z rozszerzeniem *ini* oraz *sch*, które zawierają odpowiednio dane konfiguracyjne oraz plik roboczy programu. Należy również wybrać typ mikrokontrolera, przy czym do wyboru są mikrokontrolery rodziny ST62, a wybór typu mikrokontrolera zależy od stopnia skomplikowania programu. Po opracowaniu projektu należy dokonać kompilacji programu, czyli uruchomić program Analyser oraz uruchomić Simulator i utworzyć plik *sef*. Należy także dołączyć moduły testowe symulatora, tj. próbnik cyfrowy, oscyloskop, wskaźnik stanu. Wymienione stałe elementy programowania w dalszym opisie zostały pominięte.

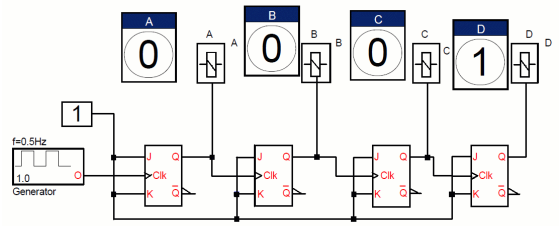
4.1. Generatory, licznik dwójkowy synchroniczny i asynchroniczny

Celem ćwiczenia jest zapoznanie się z zasadami pracy generatora OSC oraz z licznikiem dwójkowym asynchronicznym i synchronicznym. W pierwszej części ćwiczenia należy wygenerować falę prostokątną o określonej częstotliwości (np. 10 Hz), a następnie poprzez zmiany parametru „time" OSC należy uzyskać inne częstotliwości, np. 50Hz, 100Hz, 200Hz. Natomiast w drugiej części ćwiczenia należy zbudować 4 bitowy synchroniczny i asynchroniczny licznik dwójkowy o określonej pojemności, a wygenerowany sygnał prostokątny należy wykorzystać jako sygnał zegarowy. W tym celu należy:

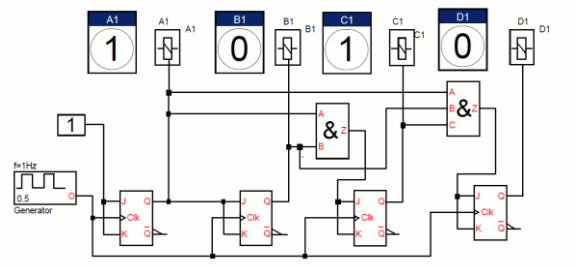
- w oknie głównym Realizera narysować schemat programu (rys. 6) pobierając elementy (OSC, DIGOUT) z biblioteki,
- przypisać wyjście DIGOUT do dowolnego wyprowadzenia portu PA lub PB, np. PA0 (push pull output),
- ustawić parametry OSCF (częstotliwość generowanych impulsów f(Hz)=1/2*time), gdzie *time* jest czasem trwania impulsu,
- sprawdzić działanie układu, zmienić parametr *time* OSCF dla f=50Hz, 100Hz, 200Hz;
- w oknie głównym Realizera narysować schemat programu (rys.7a i 7b) pobierając z biblioteki dodatkowe elementy (JKFF – przerzutnik JK, CONSTB – stała bitowa 0 lub 1).



Rys. 6. Generator impulsów prostokątnych



Rys. 7a. Asynchroniczny licznik dwójkowy modulo 16



Rys. 7b. Synchroniczny licznik dwójkowy modulo 16

4.2. Sterowanie wyświetlaczem 7 segmentowym

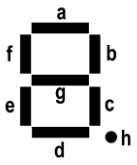
Celem ćwiczenia jest poznanie możliwości sterowania wyświetlaczem siedmiosegmentowym oraz zapoznanie się z elementami LOOKUPTABLE, COMP, COUNTF, EDGE. Ćwiczący powinien wyświetlić na wyświetlaczu liczby od 0-9. Podczas wykonywania ćwiczenia należy:

- w oknie głównym Realizera narysować schemat programu (rys. 8) pobierając z biblioteki odpowiednie elementy, np. DIGIN, 8xDIGOUT, BUNPACK-konwersja liczb z zakresu 0-255 na reprezentację dwójkową, LOOKUPTABLE – tablica zawierająca wartości wejściowe i przypisane im wartości wyjściowe, CONSTW - stała liczbowa 0-255, CONSTB – stała bitowa 0 lub 1, COMP – komparator porównuje wartość na wejściu z wartościami progowymi, COUNTF – licznik rewersyjny zliczający impulsy do określonej wartości, EDGE – wykrywa zbocze narastające impulsu prostokątnego,
- przypisać wyjścia DIGOUT do wyprowadzenia PA0-PA3, PB0-PB3,
- dla wyświetlacza o wyjściach wspólna anoda wyjścia cyfrowe DIGOUT należy skonfigurować jako otwarty dren (open-drain),
- dla wyświetlacza ze wspólną katodą, wyjścia cyfrowe DIGOUT należy skonfigurować jako push pull-output,
- wpisać dane do tablicy LOKUPTABLE według klucza zgodnego z tabelą 2,

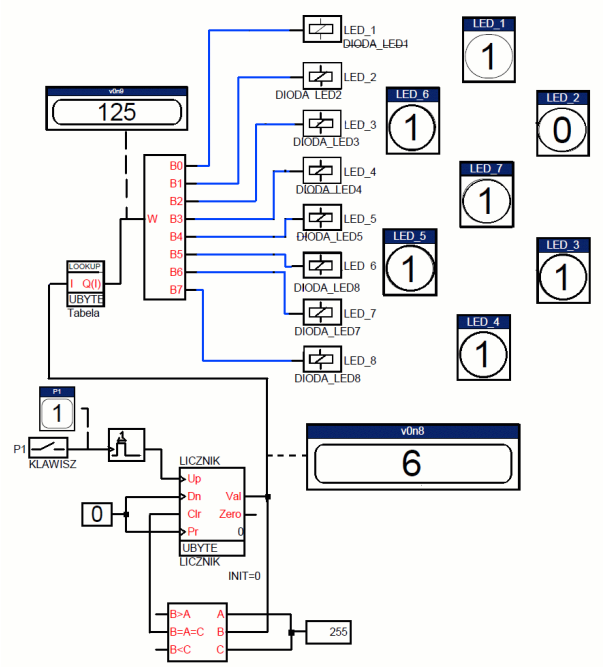
- sprawdzić zachowanie się układu dla dowolnych wartości wejściowych wpisanych do tablicy LOKUPTABLE.

Tab2. Zawartość tablicy LOKUPTABLE

wejście	wyjście	segmenty wyświetlacza
0	63	abcdef
1	6	bc
2	91	abged
3	79	abgcd
4	102	fgbc
5	109	afgcd
6	125	afgcde
7	7	abc
8	127	abcdefg
9	111	abcdfg



Każde wyjście B0-B6 BUNPACK steruje jednym segmentem wyświetlacza. Przykładowo, zadana wartość 63 powoduje włączenie segmentów a, b, c, d, e, f, czyli powoduje wyświetlenie zera.

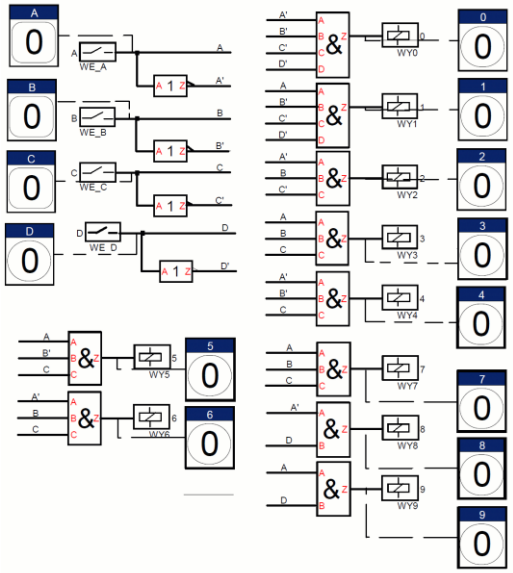


Rys .8. Sterowanie wyświetlaczem 7-segmentowym

4.3. Bloki komutacyjne

Celem ćwiczenia jest zapoznanie ćwiczących z blokami komutacyjnymi, przykładowo z możliwością realizacji dekodera kodu BCD na kod 1z 10. Ćwiczący powinien zbudować dekoder na bazie elementów bibliotecznych, np. bramki NOT oraz 2, 3 i 4 wejściowych bramek AND. Połączenia między elementami należy zrealizować przy użyciu etykiet LABEL. W tym celu należy:

- w oknie głównym Realizera narysować schemat programu (rys. 9) pobierając elementy z biblioteki,
- w poleceniu *Select hardware* dobrać typ mikrokontrolera do ilości użytych wejść i wyjść (ze względu na dużą ilość I/O wybrać ST6225),
- przypisać wejścia cyfrowe WE1-WE4 do wyprowadzeń mikrokontrolera,
- przypisać wyjścia cyfrowe D0-D9 do wyprowadzeń mikrokontrolera,
- sprawdzić zachowanie układu, zadając na wejściach A, B, C, D wartości od 0-1 należy sprawdzić działanie dekodera. W podobny sposób ćwiczący powinien zrealizować inny dekoder, np. kodu Aikena, Excess+3, kodu Gray'a, wybrany koder lub translator.

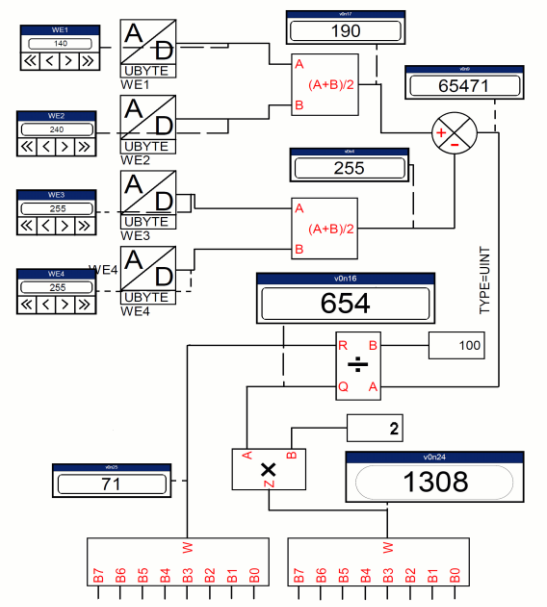


Rys. 9. Dekoder BDC na kod 1 z 10

4.4. Wybrane funkcje matematyczne

Celem ćwiczenia jest poznanie funkcji matematycznych dostępnych w środowisku Realizer, które są przykładowo wykorzystywane przy przetwarzaniu danych. Podstawowymi elementami z biblioteki elementów matematycznych są DIV, SUB2, MUL, AVERAGE. Ćwiczący powinien umieć zastosować podstawowe funkcje matematyczne. Podczas wykonywania ćwiczenia należy:

- w oknie głównym Realizera narysować schemat programu (rys. 10) pobierając z biblioteki odpowiednie elementy, np. DIV - dzielnik, MUL - mnożenie, AVERAGE - średnia, AD – przetwornik A/C, BUNPACK-konwersja z wartości 0-255 na reprezentację dwójkową,
- przypisać wejścia analogowe WE1-4 do wyprowadzeń mikrokontrolera PB0-PB3,
- dołączyć moduły testowe, zadajniki sygnału analogowego oraz próbniki cyfrowe,
- sprawdź zachowanie układu – zadając zadajnikami wartości od 0-255 należy śledzić zachowanie układu na próbnikach cyfrowych.

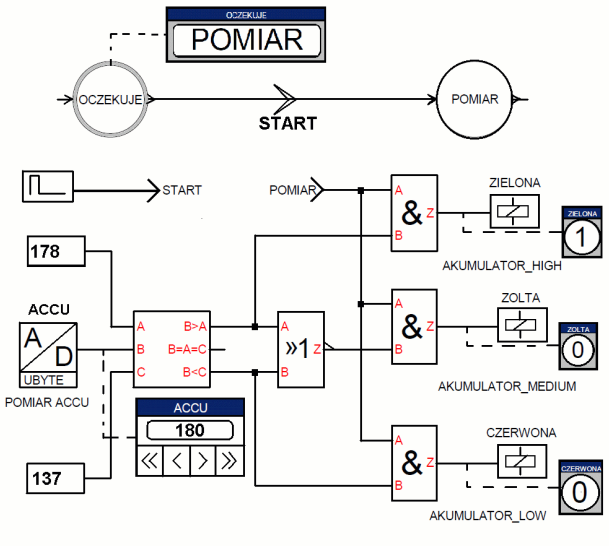


Rys. 10. Funkcje matematyczne

4.5. Wskaźnik napięcia akumulatora

Celem ćwiczenia jest zapoznanie ćwiczących z praktycznym wykorzystaniem programu Realizer. Ćwiczący powinni zbudować w pełni działający układ monitorujący poziom naładowania akumulatora. Podczas wykonywania ćwiczenia należy:

- w oknie głównym Realizera narysować schemat programu (rys. 11), należy wykorzystać elementy ADC – przetwornik A/C, COMP - komparator, DIGOUT – wyjście cyfrowe, bramki AND i NOR, STATEINIT, STATE, CONDITION, STATEIN, STATEOUT,
- przypisać wyjście analogowe POMIAR ACCU do wyprowadzeń mikrokontrolera PB0, oraz wyjścia cyfrowe LED do wyprowadzeń PA0, PA1, PA2,
- sprawdzić zachowanie układu, przy czym należy ustawić na zadajniku sygnału analogowego wartości od 0-255,
- dobrać wartości CONSTW tak, aby dioda *zielona* została włączona po przekroczeniu na wejściu POMIAR ACCU wartości 1,4V, dioda *żółta* włączona została przy napięciu od 1,0V do 1,4V, a dioda *czerwona*, gdy napięcie obniży się poniżej 1V. Wartość CONSTW dobieramy korzystając z zależności U / 0.02.



Rys. 11. Wskaźnik napięcia akumulatora

5. WNIOSKI KOŃCOWE

Zaproponowany przez autorów artykułu zestaw obejmuje kilkanaście ćwiczeń, przy czym w artykule przedstawiono tylko wybrane ćwiczenia, które zostaną wykorzystane w laboratorium Techniki Cyfrowej w KAO AM Gdynia. Ćwiczenia pozwolą na praktyczne zapoznanie się studentów z graficznym programowaniem mikrokontrolerów. Jest to całkowicie inne podejście do kwestii programowania niż w przypadku języków programowania typu C lub asembler. Programowanie graficzne w sposób bezbolesny wprowadza elektroników, którzy nie mają czasu na naukę skomplikowanych języków programowania, w świat mikrokontrolerów. Mocną stroną środowiska Realizer jest symulator programowy. Umożliwia on dokładne przetestowanie programu jeszcze przed jego załadowaniem do pamięci mikrokontrolera. Dzięki temu ćwiczenia można przeprowadzić jedynie w oparciu o symulator, bez potrzeby użycia mikrokontrolera, programatora oraz płytki zestawu edukacyjnego AVT-5072 [8]. Zestaw ten umożliwia przeprowadzenie dużej ilości ćwiczeń, bez potrzeby wykonywania płytki prototypowej. Wyposażony jest on między innymi w zasilacz, wyświetlacze LED, diody LED, LCD, przekaźniki, przyciski, złącza, triak, podstawki DIP, rezystory i tranzystory.

Studenci Wydziału Elektrycznego Akademii Morskiej w Gdyni w ramach zajęć z techniki cyfrowej mają możliwość porównania zasad przygotowania projektów i realizacji algorytmów cyfrowego sterowania w różnych technologiach i środowiskach. Obok środowiska Realizer, mogą również sprawdzić działanie układów cyfrowych w środowisku Multisim, które jest doskonałym wirtualnym laboratorium z techniki cyfrowej i mikroprocesorowej, a także z elektroniki, elektrotechniki, cyfrowego przetwarzania sygnałów [9, 10]. Realizują również algorytmy sterowania w środowisku Max Plus Baseline oraz Quartus, które następnie implementują w układach programowalnych CPLD oraz FPGA firmy Altera, sterują modelami urządzeń przemysłowych, modelami mobilnych robotów [9].

7. BIBLIOGRAFIA

1. http://www.actum.com
2. Górski K.: Realizer graficzne programowanie mikrokontrolerów, Wydawnictwo Mikom, Warszawa 2005, ISBN 83-7279-491-X
3. Chimak M.: Realizer graficzna metoda programowania mikroprocesorów, Wydawnictwo MIKOM, Warszawa 2001, ISBN 87-7279-099-X
4. Górski K.: Kurs ST6 - Realizer, cz. 1, Elektronika Praktyczna 2/2001, str.88-90, ISSN1640-7695
5. Górski K.: Kurs ST6 - Realizer, cz. 2, Elektronika Praktyczna 3/2001, str.85-87, ISSN1640-7695
6. Górski K.: Kurs ST6 - Realizer, cz. 4, Elektronika Praktyczna 4/2001, str.87-89, ISSN1640-7695
7. http://www.st.com/mcu/contentid-51.html
8. http://www.avt.com.pl
9. http://www.am.gdynia.pl/ ~jagat
10. Noga K. M., Radwański M.: Multisim. Technika cyfrowa w przykładach, Wydawnictwo BTC, 2009, ISBN 978-83-60233-48-1

GRAPHICAL PROGRAMMING OF MICROCONTROLLERS IN THE ENVIRONMENT REALIZER

Key-words: programming, microcontrollers, digital technique

The article contains a brief description of a software Realizer based on which the authors prepared a set of exercises which can be applied in digital technique laboratories. A number of applications of Realizer in graphical programming of microcontrollers ST62, ST72, ST7 are described.