ST6-Realizer

- niezwykłe narzędzie dla... opornych

Wielu początkujących (i nie tylko!) projektantów po pierwszych kontaktach z mikroprocesorami bardzo szybko się do nich zniechęca. Niby mikroprocesor potrafi wszystko, lecz tak naprawdę trzeba go tego „wszystkiego" nauczyć, pisząc odpowiedni program. Asembler nie jest językiem najbardziej przyjaznym użytkownikowi, a na zakup dobrych kompilatorów C lub Pascala mogą sobie pozwolić tylko bogate firmy. Wyjście z tej trudnej sytuacji znalazła holenderska firma Actum Soultions. Szczegóły w artykule.

Czy zdarzyło Ci się kiedyś stwierdzić, że nie ma rzeczy mniej przyjemnej, niż poprawianie po raz setny jakiegoś uparcie źle funkcjonującego fragmentu programu? Weryfikacja tasiemcowych listingów, próby przypomnienia sobie, co też takiego robi ten fragment programu, kłopoty z przepełniającym się stosem...

Z pewnością wielu Czytelników EP zna doskonale te problemy. Są one jedną z najpoważniejszych przeszkód, jakie stoją przed wszystkimi konstruktorami, którzy chcieliby rozpocząć samodzielne budowanie urządzeń mikroprocesorowych.

Najtańszym i najbardziej popularnym wśród początkujących konstruktorów narzędziem, umożliwiającym pisanie programów, są kompilatory asemblera, czasami umożliwiające linkowanie modułów z wcześniej opracowanymi fragmentami programu. Proces weryfikacji działania programu wspomagają zazwyczaj debugery lub symulatory programowe. Korzystanie z tych programów, pomimo ich ogromnej prostoty koncepcyjnej, wymaga silnej motywacji od użytkownika, co wynika z faktu, że są one zazwyczaj wyposażone w bardzo surowy interfejs, nie spełniający podstawowych zasad „przyjazności".

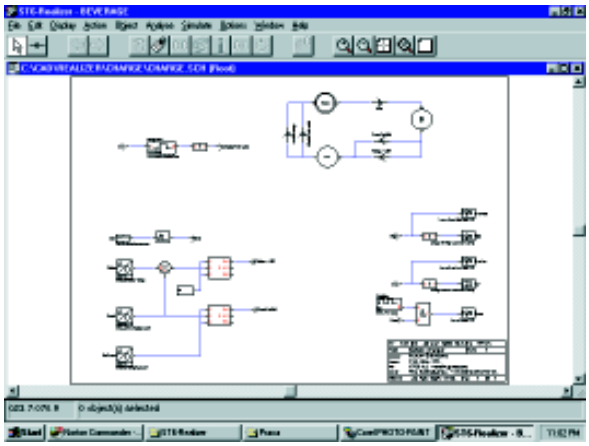
Stosunkowo szybko pojawiły się na rynku znacznie bardziej zaawansowane narzędzia dla projektantów. Początkowo były to kompilatory Pascala, wyparte z czasem przez znacznie bardziej uniwersalne kompilatory języka C (jak chociażby prezentowany przez nas od kilku miesięcy system firmy IAR). Systemy tego typu, oprócz szeregu niezaprze-

Minimalne wymagania sprzętowe pakietu ST6-Realizer

- ✓ komputer PC z procesorem 386DX lub lepszym
- ✓ pamięć RAM 4MB
- ✓ 14 MB wolnego miejsca na dysku twardym

Podstawowe możliwości pakietu ST6-Realizer

- ✓ graficzne definiowanie projektu na dowolny procesor rodziny ST6
- ✓ możliwość korzystania z gotowych bloków i funkcji logicznych
- ✓ możliwość tworzenia własnych modułów funkcjonalnych
- ✓ możliwość symulacji działania tworzonego programu
- ✓ generacja kodu wynikowego w postaci HEX (do bezpośredniego programowania pamięci programu) lub modułów do wykorzystania w programach tworzonych w sposób tradycyjny



Rys. 1.

czalnych zalet, mają także jedną wadę - ich cena przekracza zazwyczaj możliwości finansowe nie tylko amatorów, lecz także niewielkich biur konstrukcyjnych.

Tak tragiczna sytuacja (jest w tym stwierdzeniu oczywiście nieco przesady) trwała zaskakująco długo, co można wytłumaczyć tylko faktem, że środowisko konstruktorskie jest bardzo konserwatywne. Nabyte niegdyś przyzwyczajenia są cenione często znacznie bardziej, niż wygoda pracy.

Jednym z pierwszych kroków w kierunku wypracowania nowego standardu projektowania oprogramowania był AppBuilder firmy Intel. Nie ma w tym zdaniu pomyłki. Programiści tworzący AppBuildera założyli, że oprogramowanie będzie się projektować, a nie pisać! Idea ta została rozwinięta i doprowadzona do perfekcji przez holenderską firmę Actum Soultions, która opracowała niezwykłe narzędzie dla procesorów rodziny ST62 firmy SGS-Thomson. System ten nazywa się ST6-Realizer i został opracowany z myślą o maksymalnym ograniczeniu wymagań stawianych projektantowi, który chce wykorzystać w swoich opracowaniach mikrokontroler, lecz nie może sobie pozwolić na uczenie się jego architektury i języka programowania niskiego poziomu.

Sposób definiowania projektu

Oprogramowanie ST6-Realizer pracuje „pod skrzydłami" Windows (3.1/3.11 lub '95), dzięki czemu obsługa programu nie odbiega od znanych już standardów. Stosunkowo małe wymagania programu wobec komputera, na którym pracuje, znacznie ograniczają koszty realizowanego projektu.

W skład pakietu nazwanego ST6-Realizer wchodzą cztery programy:

- *Realizer*, który jest zintegrowanym edytorem i zarządcą projektu, umożliwia uruchamianie innych programów pakietu z poziomu menu;
- *Simulator*, program umożliwiający symulację programową tworzonego projektu;
- *Analyzer*, program analizujący tworzony projekt; generuje on raport z wykazem błędów (można je podglądać bezpośrednio w edytorze schematów!) oraz program wynikowy w postaci wybranej przez użytkownika;
- *Sch2Lib Converter*, jest to program odpowiadający za konwersję schematów do postaci modułów bibliotecznych, które można wykorzystywać w kolejnych projektach.

Stworzenie projektu wymaga określenia jego dwóch zasadniczych elementów:

- *algorytmu działania*, czyli opisania krok po kroku reakcji procesora na zdarzenia zewnętrzne i wewnętrzne. Mogą to być zmiany stanów logicznych na wejściach cyfrowych, zmiany poziomu napięcia na wejściach analogowych (większość procesorów serii ST62 ma wbudowany 8-bitowy przetwornik A/C), odmierzanie zadanego odcinka czasu, itp.
Algorytm zapisuje się jako graf przejść pomiędzy poszczególnymi stanami charakterystycznymi dla pracy procesora w tworzonej aplikacji. Przejścia pomiędzy stanami można zdefiniować jako warunki uzależnione od pojedynczych zjawisk lub ich logicznej kombinacji (dostępne funkcje przedstawimy w dalszej części artykułu). Po opracowaniu algorytmu działania należy przenieść go na planszę w edytorze programu ST6-Realizer.
- *zależności logicznych pomiędzy zdarzeniami*. Na tym etapie tworzymy schemat elektryczny, przedstawiający fizyczne połączenia pomiędzy wejściami i wyjściami procesora. W większości typowych wypadków definiuje się tutaj sygnały umożliwiające realizację algorytmów z przejściami warunkowymi.

Po zdefiniowaniu tych dwóch podstawowych elementów tworzonego projektu należy określić jaki typ procesora zamierzamy zastosować. Kolejnym krokiem jest przypisanie wejść i wyjść konkretnym wyprowadzeniom układu, przy czym doskonale działa system podpowiedzi wbudowany w program - na bieżąco jest wyświetlana lista wyprowadzeń dostępnych dla danego wejścia lub wyjścia. Na **rys.1** przedstawiono kompletną definicję przykładowego projektu.

Po dokładnym opisaniu algorytmu i powiązań logicznych, należy poddać projekt analizie poprawności opisu. Zadanie to realizuje program *Analyzer*, który można wywołać z poziomu *Realizera*. Na tym etapie są tworzone wszystkie pliki niezbędne do późniejszego testowania programu, kod wynikowy oraz plik raportu. Bardzo dużą pomocą dla projektanta jest wskazywanie bezpośrednio na planszy ze schematem przez program *Realizer* miejsca, w których wykryto błędy.

Po poprawnym przejściu tego etapu projektowania, można już zaprogramować procesor lub wykorzystać we własnym programie pliki ze źródłową wersją programu. Znacznie lepszym rozwiązaniem będzie jednak zweryfikowanie przy pomocy programu *Simulator* prawidłowości działania programu.

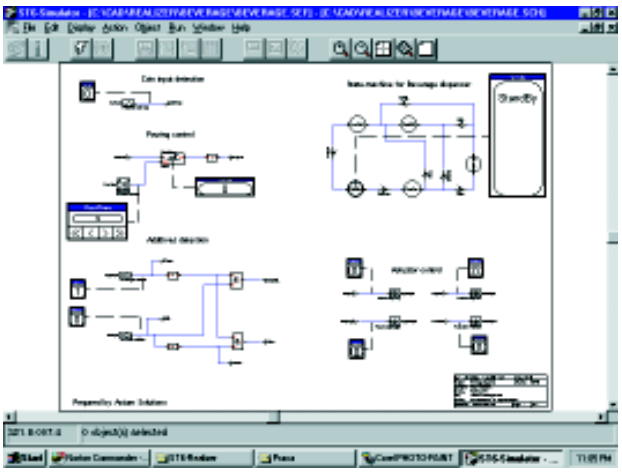
Na **rys.2** przedstawiono widok ekranu monitora z działającym programem *Analyzer*. Przy pomocy tego programu jest możliwa bardzo precyzyjna analiza sposobu działania procesora. Licznik taktów zegarowych ułatwia orientację w szybkości reakcji układu na pobudzenia zewnętrzne i określenie przybliżonego czasu

trwania procedur obsługujących poszczególne zdarzenia.

Co oferuje ST6-Realizer użytkownikom?

Prezentowane oprogramowanie umożliwia użytkownikowi korzystanie z szeregu predefiniowanych funkcji logicznych i modułów funkcjonalnych. Wśród nich są dostępne:

- wszystkie funktory logiczne (AND, NAND, OR, NOR, INV, ExOR); liczba dostępnych wejść to 2..8 (w zależności od funkcji);
- wyjście cyfrowe, wejścia analogowe i cyfrowe;
- tablice przekodowań, co upraszcza np. zmianę sposobu kodowania liczb, generowanie przebiegów o dowolnym kształcie, itp;
- licznik dwukierunkowy z możliwością ustalenia wartości początkowej;
- multipleksery;
- detektory zmian stanów logicznych;
- detektory pojawienia się zbocza;
- rejestr przesuwny;
- przetwornik A/C;
- układy arytmetyczne: sumator, subtraktor, moduł dzielący i mnożący;
- przerzutnik D;
- stałe;
- detektory przekroczenia zadanego poziomu;
- komparator;
- generator opóźnienia;
- generator przebiegu prostokątnego;
- układ czasowy (timer);
- moduł pamięci EEPROM.



Rys. 2.

Jest to więc, jak widać, bardzo bogaty zestaw modułów, z których praktycznie da się zrobić wszystko.

Operacje matematyczne można wykonywać na kilku typach liczb. Podstawowym jest *UBYTE* (1 bajt: 0..225), wersja ze znakiem *SBYTE* (1 bajt: -128..+127), *UINT* (2 bajty: 0..65536), SINT (2 bajty: -32768..+32767), oraz *LONG* (4 bajty: od -2147483648.. -2147483647).

Ponieważ jest możliwe tworzenie własnych bloków funkcjonalnych, połączenie szeregu gotowych modułów umożliwia rozszerzenie zawartości bibliotek.

Jak działa ST6-Realizer?

Każdy moduł funkcjonalny jest opisany w asemblerze przy pomocy makrofunkcji. Nie są one widoczne dla użytkownika - komfort pracy polega na operowaniu symbolami graficznymi z przyporządkowanymi w modułach bibliotecznych odpowiednimi procedurami.

W wyniku kompilacji projektu powstaje łańcuch makrofunkcji, wykonywanych w kolejności zależnej od wymagań narzuconych przez projektanta. Istnieją dwie możliwości skompilowania programu:

1. ST6-Realizer kompilując projekt może stworzyć minisystem operacyjny (ang. Realizer Operating System), który odpowiada za wstępne skonfigurowanie procesora po włączeniu zasilania (inicjuje porty I/O, watchdoga, timery, układ przerwań, itp.), oraz rozpoczyna wykonywanie zadanego programu. Tempo jego realizacji można zmieniać podczas kompilacji dobierając czas trwania pojedynczego taktu wzorcowego systemu operacyjnego. Realizacja programu odbywa się krok po kroku zgodnie z algorytmem zapisanym przy pomocy grafu. *ROS* nadzoruje poprawność wykonywania programu. W sytuacjach awaryjnych następuje restart procesora wywołany sprzętowym watchdogiem. Wynikiem takiej kompilacji jest program w postaci pliku HEX, który można wykorzystać do programowania pamięci procesora.
2. Skompilowanie programu do postaci relokowalnego modułu dla linkera, co umożliwia dołączenie opracowanej przez nas procedury do innych programów. Ponieważ w tej opcji system operacyjny jest pomijany (generowany jest tylko ciąg procedur), wszystkie bloki procesora należy inicjować oddzielnie.

ST6-Realizer - za i przeciw

Kilkutygodniowe testy pakietu ST6-Realizer wykazały, że jest to narzędzie o dużych walorach użytkowych. Dzięki opracowaniu przez producenta oprogramowania wszystkich niezbędnych w praktycznych zastosowaniach modułów funkcjonalnych i możliwości samodzielnego rozszerzania istniejących bibliotek, każdy użytkownik może bez trudu dopasować ich zawartość do własnych wymagań.

Ogromnymi atutami pakietu są łatwość budowania projektu i możliwość kompleksowego przetestowania działania układu przed zaprogramowaniem proce-

sora. Zastosowanie popularnego interfejsu graficznego czyni z *Realizera* narzędzie prawdziwie przyjazne, zwłaszcza dla mało wprawnych użytkowników. Całość uzupełnia doskonała dokumentacja, której wadą (zaletą?) jest fakt, że występuje tylko w języku angielskim.

Najistotniejszą wadą, jaką udało się nam wychwycić podczas tworzenia projektów testowych, to „zachłanność" kompilatora na pamięć programu. Jest to wynik ogromnej uniwersalności makrofunkcji stanowiących rdzeń programu. Ograniczenie to jest o tyle łatwe do ominięcia, że w rodzinie ST6 procesory z pamięcią 4 lub 8kB są łatwe dostępne i stosunkowo tanie.

Pamięciożerność *Realizera* maleje wraz ze wzrostem ogólnych rozmiarów programu - świadczy to o tym, że stosunkowo dużo miejsca w pamięci zajmuje system operacyjny *ROS*.

Prezentowane narzędzie można śmiało polecić zarówno początkującym elektronikom, jak i posiadającym doświadczenie, którzy chcą rozszerzyć swoje możliwości bez konieczności żmudnego uczenia się nowego języka programowania. Doświadczeni twórcy programów na procesory ST6 potraktują *Realizera* raczej z przymrużeniem oka, gdyż (opinia autora) rzeczą bardzo cenioną jest możliwość absolutnego panowania nad strukturą tworzonego programu. Tego ze względów zupełnie oczywistych *Realizer* nie może zaoferować.

Warto jednak zapoznać się z tym oprogramowaniem, gdyż jest to jeden z najdoskonalszych przykładów nowego (i chyba przyszłościowego) podejścia do tworzenia aplikacji w oparciu o mikrokontrolery. Atrakcyjna cena zachęca do podjęcia ryzyka, gdyż jest ona porównywalna do ceny kompilatora asemblera z prostym linkerem.

Tak czy inaczej, zanosi się na to, że już w niedalekiej przyszłości pisanie programów wyjdzie z mody. Modne będzie ich rysowanie.

Piotr Zbysiński, AVT

System ST6-Realizer udostępniła redakcji firma Eltron.