Realizacja projektów na 8051 przy pomocy oprogramowania firmy IAR SYSTEMS

IAR TINY-51 - najszybsza droga do aplikacji wielozadaniowych

Na zakończenie prezentacji części teoretycznej pakietu oprogramowania firmy IAR-Systems przybliżymy dodatkowe narzędzie, umożliwiające tworzenie wielozadaniowych aplikacji stosowanych w systemach wykorzystujących 8-bitowe mikrokomputery z rodziny MCS-51.

TINY-51, to dodatkowe, doskonałe narzędzie, które umożliwia szybkie tworzenie wielozadaniowych aplikacji z wykorzystaniem kontrolerów jednoukładowych z rodziny MCS-51. Tak, jak pozostałe elementy systemu Embedded Workbench, ten wielozadaniowy rdzeń wraz z biblioteką wchodzi w skład prezentowanego pakietu firmy IAR. Dzięki temu narzędziu proces tworzenia aplikacji zostaje skrócony do minimum.

„Wielozadaniowość" (wielowątkowość) tego rozwiązania polega na oddzielnym tworzeniu poszczególnych wątków oraz opracowywaniu zależności i sposobów współpracy pomiędzy wykonywanymi jednocześnie w systemie zadaniami. Cudzysłów przy słowie wielozadaniowość jest nieprzypadkowy, bowiem fizycznie każde zadanie podczas pracy procesora zajmuje jego określony czas, wiadomo wszakże że procesory serii MCS-51 to układy jednopotokowe.

Charakterystyka ogólna

Rdzeń TINY-51 może być wykorzystywany także w aplikacjach, w których mikrokontroler pracuje tylko z wykorzystaniem wewnętrznej pamięci RAM (128B). Sytuacja taka mam miejsce np. przy wykorzystaniu układów 87/89C51/2051 i podobnych, o strukturze takiej, jak w przypadku podstawowego modelu procesora 8051. W takim jednak przypadku jest możliwe wprowadzenie maksymalnie kilku zadań, i to z pewnymi ograniczeniami, ze względu na niewystarczającą ilość pamięci danych przeznaczoną na stos systemowy. Dlatego dla pełnego wykorzystania możliwości modułu TINY-51 zaleca się stosowanie procesora z 256B RAM (np. 8052) oraz dołączoną zewnętrzną pamięcią danych (XDATA - max. 64kB).

W praktyce programista w prosty sposób może zmieniać przydział czasów dla poszczególnych zadań, modyfikując parametry stałe w zbiorach źródłowych tiny51.h i tiny51.i, które są dostarczane w pakiecie Embedded Workbench.

Do przełączania zadań w programie wielozadaniowym wykorzystuje się Timer 0 procesora oraz bank 3 rejestrów R0...R7. W przypadku timera programista ma możliwość implementacji innego licznika, np. T1 lub T2, chcąc wykorzystać Timer 0 dla innych potrzeb.

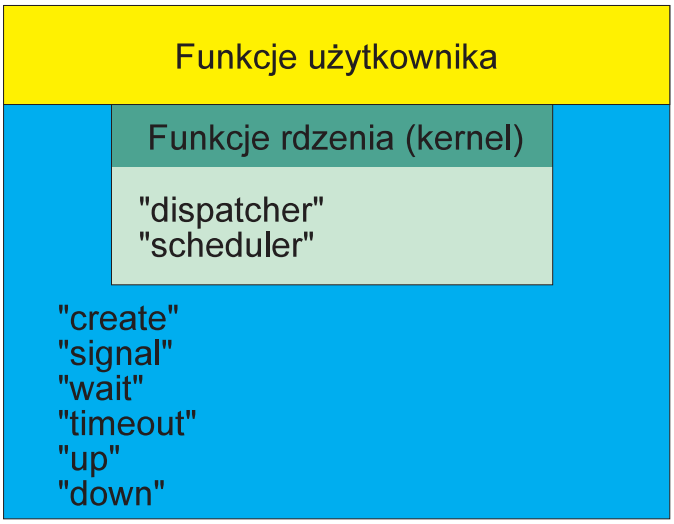
Terminologia - dowód na prostą doskonałość

Aby zaprezentować czytelnikom ideę działania rdzenia TINY-51, zostanie przedstawionych kilka definicji pojęć używanych przy tworzeniu programu wielozadaniowego.

Task (zadanie) - program, który fizycznie może być wykonany niezależnie od pozostałych zadań. Różne zadania mogą wymieniać między sobą dane (komunikować się) i synchronizować (uzależniać) swoje wykonywanie w zależności od pozostałych. W przypadku układów z jednym kontrolerem, każde zadanie zabiera określony czas procesora, a ich kolejność i alokacja określona jest w specjalnej liście tzw. *scheduler* (ang. plan, lista).

Idle-task - specjalny rodzaj zadania (programu), który może zostać wykonany, gdy żadne z pozostałych zadań nie oczekuje na wykonanie. W praktyce jest to stan jałowy procesora (idle-state).

Tasks states (stany zadania) - każde z zadań może znajdować się w jednym z kilku stanów:
- *running* - zadanie znajduje się w trakcie wykonywania;
- *ready* - zadanie oczekuje na wykonanie w kolejce;



Rys. 1.

- *waiting* - zadanie oczekuje na komunikację z innym zadaniem (zadaniami);
- *stopped* - zadanie nie zostaje wykonane (zdjęte z kolejki);
- *new* - zadanie zostało przydzielone do kolejki (nowe zadanie);

Dispatcher - fragment wielozadaniowego programu, który odpowiada za przerwanie aktualnie wykonywanego zadania i rozpoczęcie wykonywania nowego. Kolejność wykonywanych zadań znajduje się w liście zadań.

Scheduler - fragment programu decydujący, które z zadań ma zostać wykonane jako następne, czyli określa kolejność wykonywania poszczególnych zadań. Istnieje wiele metod i algorytmów, na podstawie których odbywa się porządkowanie poszczególnych zadań, dlatego też efektywność działania całego programu uzależniona jest od wyboru odpowiedniego z nich.

Round-robin - fragment programu zawierający algorytm, który określa kiedy dane zadanie, będące w kolejce, zostanie wykonane. Ilość czasu przeznaczona na każde zadanie jest limitowana. Kiedy czas wykonywania określonego zadania kończy się, zostaje ono zawieszone, a następnie wstawione na koniec kolejki zadań.

Preemptive multitasking - rodzaj pracy wielozadaniowej w której dispatcher sprawuje całkowita kontrolę nad przebiegiem zadań - czyli zgodnie z jego funkcją w odpowiednim momencie przerywa wykonywanie jednego zadania i aktywuje inne.

Non-preemptive multitasking - inaczej niż w poprzednim przypadku, ten rodzaj wielozadaniowości polega na tym, że każde z wykonywanych zadań niejako odpowiada samo za siebie, czyli samo odpowiada za swoje wykonanie. Zakończenie zadania następuje w momencie zakończenia wykonania przez procesor instrukcji tego zadania.

Signals (sygnały) - dzięki tej części wielozadaniowego programu możliwa jest synchronizacja pomiędzy poszczególnymi zadaniami.

Semaphore (semafory) - semafory są używane w krytycznych częściach programu, gdzie może dojść do sytuacji, w której w jednym momencie dwa lub kilka zadań jednocześnie będzie żądało obsługi przez procesor (np. przy aplikacjach wykorzystujących rozbudowane systemy przerwań).

Struktura zadania

TINY-51 zawiera dwie części: funkcje rdzenia (ang. *kernel functions*) i funkcje użytkownika (ang. *user functions*). Rdzeń jest napisany w języku asemblera, w jego skład wchodzą dwie funkcje opisywane wcześniej: *dispatcher i scheduler*.

Funkcje użytkownika są zdefiniowane w języku C. Funkcje te umożliwiają użytkownikowi wykonywanie prostych, wielozadaniowych czynności, np. wstawienie zadania do listy (kolejki) oraz pozwalają na komunikację między wieloma zadaniami.

Każde zadanie jest strukturą opisaną w sposób przedstawiony poniżej. Opis ten wykorzystuje jednocześnie rdzeń oraz funkcje użytkownika.

```
struct TASK {
struct TASK *nextptr
byte pid;
byte wait_signals;
byte rec_signals;
byte timeout;
byte state;
byte *sp
void (*pushfunc)();
void (*popfunc)();
};
```

nextptr - wskaźnik kolejnego zadania;

pid - numer zadania; wartość 0 jest zarezerwowana dla specjalnego zadania związanego z przeterminowaniem, natomiast wartość 255 jest wykorzystywana wewnętrznie przez rdzeń. Użytkownik ma do dyspozycji numery: 1...254;

wait_signals - zmienna (maska bitowa) określająca, na który z sygnałów oczekuje dane zadanie;

rec_signals - maska bitowa określająca, który z odebranych sygnałów ma za zadanie aktywację tego zadania;

timeout - wewnętrzny licznik przeterminowania każdego z zadań; licznik ten jest dekrementowany przez funkcję TimeoutTask; kiedy timeout osiąga wartość 0, funkcja wysyła sygnał do danego zadania;

state -zmienna do wewnętrznego użytku przez rdzeń (kernel); w praktyce używana przez scheduler przy wyborze następnego zadania z listy;

sp - zmienna używana do przechowania aktualnego wskaźnika stosu w przypadku przełączania zadań;

pushfunc - funkcja zachowuje zmienne lokalne danego zadania przy przełączaniu na inne zadanie przez dispatchera;

popfunct - funkcja jest wywoływana po zawieszeniu wykonywania poprzedniego zadania (przerwanego przez dispatchera).

W jednym z kolejnych numerów EP przedstawimy Czytelnikom przykład wykorzystania systemu Embedded Workbench pracującego z emulatorem sprzętowym procesorów MCS-51 do utworzenia przykładowego projektu konkretnego urządzenia.

Sławomir Surowiński, AVT