Realizacja projektów na 8051 przy pomocy oprogramowania firmy IAR SYSTEMS

Kontynuując opis pakietu kompilatora C firmy IAR, prezentujemy jego możliwości konfiguracyjne oraz ich znaczenie w odniesieniu do realizacji projektów na procesorach rodziny MCS-51.

Typowa droga tworzenia projektu w skład którego wchodzi oprogramowanie na wybrany mikroprocesor, przedstawiona jest na **rys.1**.

Każdy programista systemów opartych na mikrokontrolerach jednoukładowych doskonale wie, że każdy projekt jest odmienny, patrząc z punktu konfiguracji podstawowych elementów takich jak np. podział pamięci, sposób rozdziału i wykorzystania portów I/O. Różne są także wymagania co do wielkości stosu systemowego.

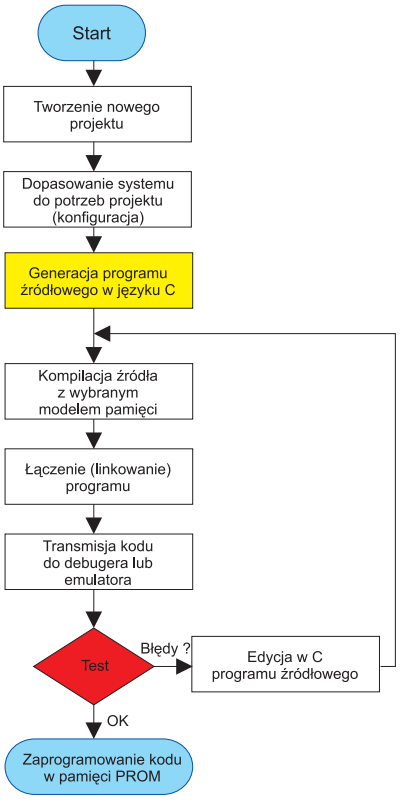
Wielu, najczęściej mniej doświadczonych, programistów tworzy oprogramowanie w sposób indywidualny, tzn. bez korzystania z utworzonych wcześniej bibliotek procedur standardowych. Prowadzi to do wydłużenia czasu tworzenia projektu i utrudnia analizę kodu źródłowego, po dłuższym czasie, np. kiedy zajdzie potrzeba modyfikacji programu przy tworzeniu kolejnej jego wersji.

Działanie takie jest w zasadzie do przyjęcia przy tworzeniu mało złożonych aplikacji, często pozwala ono początkującemu programiście na bardziej swobodne traktowanie każdego projektu, zgoła indywidualnie. Sprawa komplikuje się przy pisaniu programów przeznaczonych do bardziej zaawansowanych zastosowań. Ma to miejsce szczególnie w przypadku tworzenia systemów wieloprocesorowych, bądź wymagających obliczeń arytmetyki stało i zmiennoprzecinkowej. W takim przypadku niezbędne i uzasadnione jest jak najszybsze przejście i oswojenie się ze wspomnianą hierarchiczną strukturą tworzenia programu z wykorzystaniem standardowych elementów, tak często powtarzanych w wielu projektach.

Zintegrowane środowisko do tworzenia aplikacji jest najlepszą metodą na oduczenie się złych nawyków i efektywniejsze tworzenie programów, bądź w wersji asemblerowej, bądź na poziomie języka wyższego poziomu.

Pierwszym krokiem podczas tworzenia nowego projektu przy użyciu prezentowanego pakietu jest zdecydowanie się na odpowiednią konfigurację systemu. Najogólniej mówiąc wymaga to wybrania modelu pamięci czyli sposobu adresowania i wykorzystania segmentów programu, zawartych w nich stałych i zmiennych. W przypadku pamięci ROM użytkownik kompilatora definiuje w zależności od potrzeb:

- tradycyjny (bezpodziałowy) sposób wykorzystania pamięci programu (non-bankable);
- z podziałem na banki (bankable);
- segment stałych;
- segment zmiennych - inicjujących (tzn. takich stałych które są wartościami początkowymi zmiennych używanych w programie, a fizycznie umieszczonych w pamięci RAM).



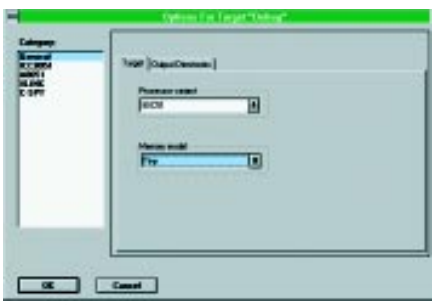
Rys. 1.

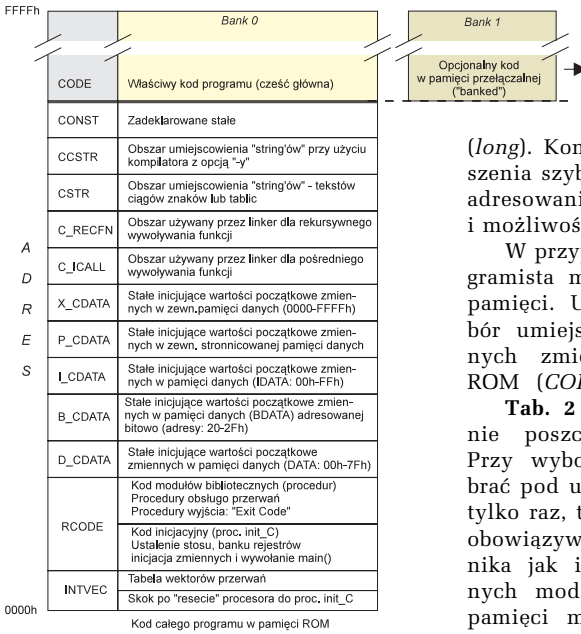
Tabela 1.

Pozycja "pakietu" konfiguracyjnego	Konfigurowany element (program)
model pamięci	kompilator, linker
alokacja pamięci	linker
pamięć typu NV-RAM	kompilator-słowo kluczowe, linker
rozmiar stosu (stack size)	linker
podstawowe funkcje I/O	
putchar i *getchar*	moduły biblioteczne
funkcje *printf/scanf*	linker
rozmiar sterty (heap size)	moduł biblioteczny opisujący stertę
sposób inicjalizacji sprzętu i pamięci	moduł *CSTARTUP*

Przy wyborze modelu pamięci RAM mamy do dyspozycji:

- adresowaną wprost pamięć wewnętrzna procesora (internal RAM);
- pośrednio adresowaną pamięć RAM;
- pracę z zewnętrzna pamięcią danych (external memory);
- oraz zewnętrzną pamięć typu NV-RAM - RAM z podtrzymaniem bateryjnym.


Rys. 2.



Rys. 3.

W praktyce użytkownik deklaruje używane w projekcie typy pamięci poprzez nadanie im odpowiednich - rozpoznawalnych przez kompilator i linker nazw (np. *RCODE, CODE, DATA, D_IDATA, XDATA i NO_INIT*). Znaczenie poszczególnych nazw opiszemy w dalszej części artykułu.

Rozmiar i położenie (adres) danego segmentu oraz jego funkcja są określane w linii parametrów wywołania lub w specjalnie utworzonych zbiorach parametrów kompilatora C 8051 oraz linkera. W skład takiego „pakietu" parametrów wchodzą elementy przedstawione w **tabeli 1**.

Poniżej opiszemy poszczególne pozycje, tak aby wyjaśnić Czytelnikom ich znaczenie przy tworzeniu przykładowego programu.

XLINK Command file - zbiór konfiguracyjny linkera

Pakiet IAR zawiera trzy zbiory parametrów linkera dla trzech sposobów wykorzystania pamięci ROM

	nazwa zbioru z komendą wywołania
bez podziału na banki (non-banked)	*lnk8051.xcl*
typ 80751	*lnk8051a.xcl*
z podziałem na banki (bank-switched)	*lnk8051b.xcl*

Modele Pamięci

Wszystkie procesory rodziny 8051 pozwalają na dwa rodzaje adresowania: bliski (*short*) i daleki (*long*). Kompilator używa ich do zwiększenia szybkości wykonywania procedur adresowania w zależności od potrzeb i możliwości systemu.

W przypadku kompilatora C IAR programista ma dostęp do sześciu modeli pamięci. Umożliwia to optymalny wybór umiejscowienia lokalnych i globalnych zmiennych w obszarze pamięci ROM (*CODE*) i RAM (*DATA*).

Tab. 2 wyjaśnia praktyczne znaczenie poszczególnych modeli pamięci. Przy wyborze modelu pamięci należy brać pod uwagę fakt, iż można to zrobić tylko raz, tzn. że wybrany model będzie obowiązywał tak w modułach użytkownika jak i we wszystkich, wykorzystanych modułach bibliotecznych. Model pamięci musi być określony zarówno dla kompilatora jak i linkera. Ustalenie wybranego typu przy korzystaniu ze środowiska „Embedded Worbench" odbywa się w oknie opcji (**rys.2**).

Poniżej opisane zostanie znaczenie poszczególnych typów segmentów pamięci oraz ich powiązanie ze sprzętem opartym na procesorze 8051.

CODE memory

Pamięć typu CODE rozciąga się w zakresie adresowania procesora 8051: 0h...0FFFFh. W zależności od konkretnego typu mikroprocesora, część tej pamięci w zakresie 0...16kB może znajdować wewnątrz samego układu (internal ROM). Zewnętrzna pamięć programu rozpoczyna się począwszy od „końca" wewnętrznej pamięci i może być dostępna poprzez taki sam sposób adresowania.

Jeżeli użytkownik decyduje się na częściowe wykorzystanie pełnego obszaru adresowego procesora - 64kB, to segment ten może być umieszczony pod dowolnym adresem z zakresu 0h...0FFFFh. Zwykle adres początkowy segmentu kodu ma wartość 0000h, ze względu na skok pod ten adres procesora, po jego restarcie.

Rys. 3 ilustruje sposób wykorzystania przez kompilator segmentu CODE , w którym zawarto: tablice wektorów przerwań, instrukcje inicjujące, moduły biblioteczne (procedury), stałe inicjujące zmienne, stałe programowe oraz kod programu użytkownika.

W górnej części rysunku widać opcjonalny bank pamięci programu. Zastosowanie takiego podziału możliwe jest dzięki kompilatorowi ICC8051 przy wyborze modelu pamięci przełączalnej (banked-memory). Program umożliwia wykorzystanie maksymalnie 256 banków. W tym trybie pamięć programu adresowana jest poprzez 16-bitowy adres oraz numer 8-bitowy banku. Ten ostatni wystawiany jest przez mikroprocesor poprzez jeden z jego wolnych portów (np. P1 dla układu 8051). Istotny jest fakt, iż w obrębie wyższych banków (powyżej banku 0) znajduje się tylko kod główny programu, nie mogą znajdować się tam np. procedury biblioteczne, obsługi przerwań sprzętowych, stałe inicjujące zmienne, czy stałe programowe. Problem korzystania z przełączanych banków pamięci opiszemy w kolejnej części artykułu.

Sławomir Surowiński, AVT

Tabela 2.

Model	tiny	small	compact	medium	large	banked
opcja wywołania	-mt	-ms	-mc	-mm	-ml	-mb
program	DATA	IDATA	XDATA	XDATA	XDATA	XDATA
zm.lokalne	DATA	IDATA	DATA	IDATA	XDATA	XDATA
pamięć zewn. RAM	nie	nie	tak	tak	tak	tak
max. rozmiar kodu programu	64kB	64kB	64kB	64kB	64kB	>1MB
typowy procesor	8051	8052	8031	8032	8032	8032
biblioteka C	cl8051t	cl8051s	cl8051c	cl8051m	cl8051l	cl8051b
biblioteka C dla typu 80751	cl805ta	cl8051sa	nie dotyczy	nie dotyczy	nie dotyczy	nie dotyczy