Mikrokontrolery Precision32 (1)

Rozpoczęcie pracy

W artykule pokazujemy krok po kroku jak rozpocząć pracę z opartymi na rdzeniu ARM Cortex-M3 mikrokontrolerami z rodziny Precision32 firmy Silicon Labs.

Utworzenie oprogramowania dla mikrokontrolera to jeden z głównych elementów procesu projektowania i budowania systemu wbudowanego. Nim jednak schematy blokowe, opis, notacja UML bądź inne formy określające funkcjonalność oprogramowania staną się kodem wynikowym wykonywanym przez mikrokontroler, programistę czeka wiele zadań, które musi wykonać. Należą do nich:

- Wykonanie szablonu projektu w środowisku programistycznym.
- Wykonanie kodu konfiguracyjnego włączającego zasoby mikrokontrolera i ustawiającego parametry ich pracy.
- Wykorzystanie zasobów mikrokontrolera do zaimplementowania algorytmów i mechanizmów realizujących ustaloną funkcjonalność.

W kolejnych rozdziałach niniejszego artykułu szczegółowo opisujemy jak przy wykorzystaniu dostępnych narzędzi zrealizować wymienione wyżej zadania na przykładzie wykonania od początku kompletnej aplikacji.

Oprogramowanie narzędziowe

W celu opisania wyżej wymienionych zagadnień użyta zostanie pewna grupa narzędzi sprzętowych i programistycznych. Pierwszym narzędziem sprzętowym jest programator/debuger *USB Debug Adapter* firmy Silicon Labs. Został on opisany w artykule „Narzędzia dla Precision32 (1)” w EP 12/2012. Drugim narzędziem sprzętowym jest płytka uruchomieniowa z mikrokontrolerem z rodziny Precision32. Jej projekt został przedstawiony w artykule „Płytka uruchomieniowa z mikrokontrolerem Precision32” w bieżącym numerze Elektroniki Praktycznej.

Rysunek 1. Elementy składowe pakietu Precision32 Development Suite

Pierwszym z narzędzi programistycznych jest program komputerowy *Silicon Labs Precision32 AppBuilder*. Za jego pomocą, przy wykorzystaniu graficznego interfejsu użytkownika, programista może wygenerować kod konfiguracyjny dla mikrokontrolera. Drugim narzędziem programistycznym jest środowisko *Keil MDK-ARM* służące do tworzenia oprogramowania. Oba narzędzia zostały opisane w artykule „Narzędzia dla Precision32 (2)” w EP 2/2013. Trzecim z narzędzi programistycznych jest oprogramowanie w postaci kodu źródłowego dla Precision32, które nosi nazwę *Precision32 SDK* (*Software Development Kit*). Przedstawiono je w artykule „Narzędzia dla Precision32 (3)” w EP 6/2013.

Instalacja narzędzi programistycznych

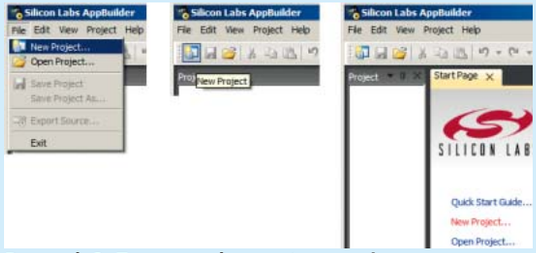
Program *Precision32 AppBuilder* i oprogramowanie *Precision32 SDK* są częścią pakietu *Precision32 Development Suite*. Dlatego, aby móc z nich korzystać, jest konieczne zainstalowanie całego pakietu. W pakiecie *Precision32 Development Suite*, oprócz *Precision32 AppBuilder* i *Precision32 SDK* znajdują się jeszcze następujące narzędzia: Precision32 IDE, Precision32 Utilities, SiLabs 32-bit SDK oraz USB to UART Bridge Driver (opcjonalny) (**rysunek 1**). Plik instalacyjny pakietu można pobrać ze strony internetowej firmy Silicon Labs o adresie *http://goo.gl/fC4EGm*.

Kolejnym potrzebnym narzędziem programistycznym jest środowisko do tworzenia oprogramowania. Mogłoby nim być *Precision32 IDE*, które zostało zainstalowane jako jeden z elementów pakietu *Precision32 Development Suite*, jednak ze względu na fakt, iż większą funkcjonalność oferują IDE komercyjne wybrano pakiet *ARM-MDK* firmy Keil. Pakiet ten można pobrać ze strony internetowej firmy Keil o adresie *http://goo.gl/H0xFJe*.

Korzystanie z programu Precision32 AppBuilder

W pierwszej kolejności wykorzystany zostanie program AppBuilder. Dzięki niemu stworzony zostanie projekt konfiguracji mikrokontrolera. Na bazie tego projektu zostanie stworzony projekt programistyczny, który zawierał będzie pliki z kodem źródłowym realizującym wybraną konfigurację. Kod źródłowy wykorzystuje gotowe funkcje z bibliotek si32Hal będących częścią oprogramowania Precision32 SDK (*Software Development Kit*).

Po włączeniu programu *AppBuilder* należy utworzyć nowy projekt. Można to zrobić na trzy sposoby (**rysunek 2**):



Rysunek 2. Trzy sposoby na stworzenie nowego projektu w programie AppBuilder

Rysunek 3. Okno konfiguracyjne nowego projektu w programie AppBuilder

- Wybierając z menu opcji *File*, a następnie *New Project*.
- Poprzez naciśnięcie ikony nowego projektu, która jest umieszczona pod menu.
- Poprzez naciśnięcie *New Project* w panelu *Start Page*.

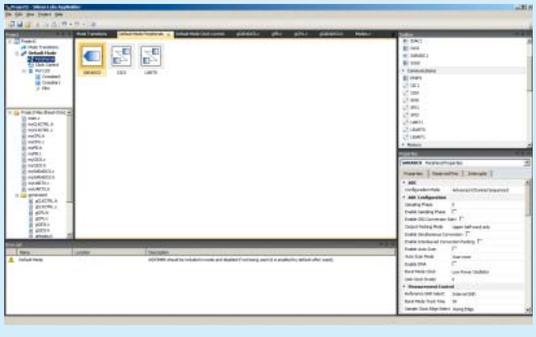
Wyświetlone zostanie nowe okno konfiguracyjne projektu, w którym należy uzupełnić pola dotyczące: grupy mikrokontrolerów, modelu w danej grupie, rewizji modelu, wersji używanego oprogramowania *Precision32 SDK*, nazwy projektu, miejsca zapisania projektu na dysku twardym komputera, nazwy środowiska do tworzenia oprogramowania dla którego zostanie wygenerowany projekt programistyczny, rodzaju kompilatora oraz ścieżki na dysku, gdzie znajduje się oprogramowanie Precision32 SDK. W przypadku naszego projektu pola te zostały wypełnione w sposób, jaki pokazano na **rysunku 3**.

Po zaakceptowaniu ustawień projekt zostanie stworzony, a następnie AppBuilder otworzy go do edycji. Do nawigacji po projekcie służy drzewo projektu, które znajduje się w lewym górnym oknie programu *AppBuilder*. Za jego pomocą należy przejść do zakładki *Peripherals* (**rysunek 4**). Składają się na nią trzy okna. W prawym, górnym oknie programu *AppBuilder* o nazwie *Toolbox* jest dostępna lista peryferiów mikrokontrolera. Dwukrotne kliknięcie na dowolny zasób spowoduje przeniesienie go do znajdującej się w środkowym oknie *Default Mode Peripherals* listy peryferiów, które zostaną włączone. Pojedyncze kliknięcie na wybrane peryferium z tej listy spowoduje wyświetlenie w prawym dolnym oknie *Properties* dostępnych dla niego parametrów pracy, które użytkownik może ustawić.

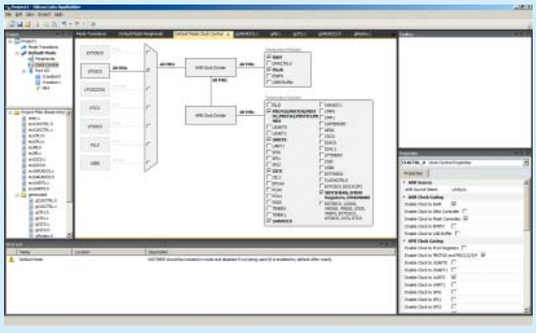
Następnie za pomocą drzewa projektu należy przejść do zakładki *Clock Control* (**rysunek 5**). Główne okno tej zakładki (*Default Mode Clock Control*) przedstawia schemat dystrybucji sygnału zegarowego w mikrokontrolerze. Schemat ten składa się z bloków reprezentujących źródła sygnału zegarowego, dzielniki sygnału zegarowego i listę sygnałów zegarowych taktujących peryferia. Przy wykorzystaniu dostępnych na schemacie opcji oraz opcji dodatkowych, w prawym dolnym oknie *Properties*, użytkownik ma w jednym miejscu dostęp do wszystkich ustawień dotyczących sygnałów zegarowych mikrokontrolera.

Po zakończeniu konfigurowania sygnałów zegarowych można przejść do konfigurowania wyprowadzeń mikrokontrolera. Do tego celu służą zakładki drzewa projektu o nazwie *Crossbar0*, *Crossbar1* i *PB4* (**rysunek 6**). Główne okno każdej z tych zakładek udostępnia listę wyprowadzeń mikrokontrolera oraz ich graficzne rozmieszczenie w obudowie układu. Użytkownik wybierając dane wyprowadzenie ma możliwość wybrania dla niego (w prawym dolnym oknie *Properties*) trybu pracy. Pozwala to na elastyczne przyporządkowanie według potrzeb wyprowadzeń do wyjść i wejść peryferiów.

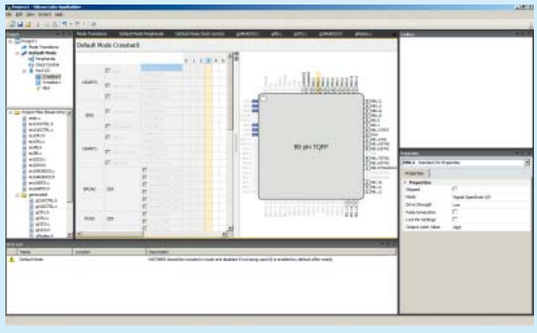
Warto zauważyć, że niezależnie od tego, która zakładka jest w danej chwili aktywna (*Peripherals*, *Clock Control*, *Crossbar0*, *Crossbar1*, *PB4*), to zawsze pozostają widoczne dwa okna programu. Pierwsze z nich, usytuowane z lewej strony, przedstawia listę plików, które zostaną wygenerowane wskutek działania programu



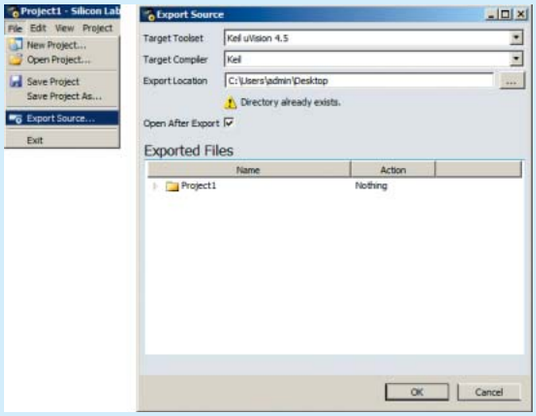
Rysunek 4. Zakładka „Peripherals" programu AppBuilder służąca do skonfigurowania i włączenia peryferiów mikrokontrolera



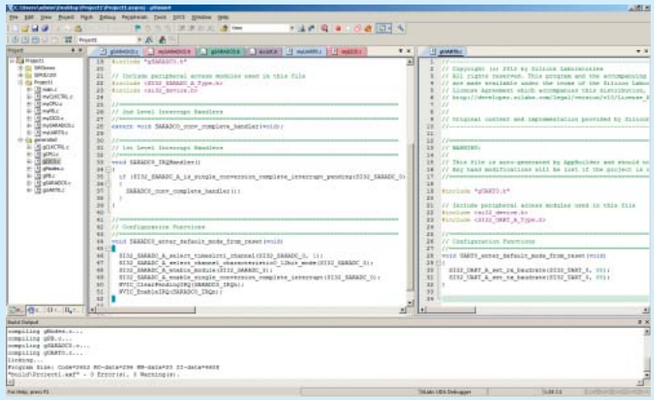
Rysunek 5. Zakładka „Clock Control" programu AppBuilder służąca do skonfigurowania sygnałów zegarowych mikrokontrolera



Rysunek 6. Zakładka „Crossbar0" programu AppBuilder służąca do skonfigurowania trybu pracy wyprowadzeń mikrokontrolera



Rysunek 7. Okno „Export Source" programu AppBuilder służąca do wygenerowania plików z kodem źródłowym



Rysunek 8. Wygenerowany na bazie projektu programu AppBilder projekt programistyczny w środowisku Keil MDK-ARM

AppBilder. Pliki te są aktualizowane na bieżąco podczas edycji projektu w programie *AppBuilder*, dzięki czemu użytkownik w każdej chwili może otworzyć wybrany plik i mieć wgląd w jego zawartość, która odzwierciedla aktualną konfigurację. Drugie okno o nazwie *Error list* jest umieszczone na samym dole *AppBuilder*. Zgodnie z nazwą to okno wyświetla informacje dotyczące błędów i ostrzeżeń dotyczących konfiguracji. Przykładowo, gdy użytkownik wybierze przetwornik A/C jako peryferium, które ma zostać włączone, program *AppBuilder* poprzez okno *Error list* zwróci uwagę na potrzebę włączenia sygnału zegarowego dla tego peryferium.

Gdy praca nad projektem w programie *AppBuilder* zostanie zakończona (peryferia zostaną włączone i skonfigurowane, sygnały zegarowe odpowiednio ustawione oraz każde z używanych wyprowadzeń uzyska tryb i parametry pracy), można przystąpić do wygenerowania kodu źródłowego dla mikrokontrolera. Aby to uczynić, należy wybrać z menu opcji *File*, a następnie *Export Source*. *Appbuilder* otworzy okno *Export Source* (**rysunek 7**), w którym należy wybrać środowisko programistyczne, kompilator oraz ścieżkę docelową, pod którą zapisane zostaną pliki z kodem źródłowym. Po zaakceptowaniu (przycisk *OK*) pliki zostaną wygenerowane.

Jeśli w oknie *Export Source* zaznaczono opcję *Open After Export*, projekt programistyczny zostanie automatycznie otworzony w środowisku programistycznym (**rysunek 8**).

Teraz programista może przystąpić do zaimplementowania algorytmów i mechanizmów realizujących ustaloną funkcjonalność aplikacji. Najwygodniejszym sposobem na zrealizowanie tego zadania jest wykorzystanie służących do sterowania peryferiami funkcji z dołączonych do projektu bibliotek *si32Hal* (są to te same biblioteki, które zostały wykorzystane do stworzenia kodu konfiguracyjnego w programie *AppBuilder*).

Gdy kod źródłowy zostanie już uzupełniony o powyższe fragmenty, należy go skompilować. Następnie można już przystąpić do wgrania kodu wynikowego do pamięci mikrokontrolera w celu jego wykonania lub debugowania.

Szymon Panecki
Wydział Elektroniki, Politechnika Wrocławska
szymon.panecki@pwr.wroc.pl

Literatura

[1] www.silabs.com AN670: Getting started with the Silicon Labs Precision32 AppBuilder
[2] www.silabs.com AN719: Precision32 IDE and AppBuilder detailed tutorial and walkthrough