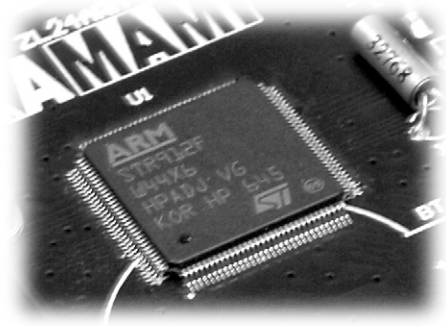
STR91x – alternatywny flasher



Mikrokontrolery z rodziny STR91x, produkowane przez firmę STMicroelectronics, to jedne z najpotężniejszych „jednoukładowców" dostępnych na (nie tylko polskim) rynku. Cechuje je nowoczesny rdzeń ARM966E–S, duża pojemność pamięci Flash (do 512 kB) i SRAM (do 96 kB) oraz bogaty zestaw peryferiów – między innymi kontroler MAC Ethernet. W artykule przedstawię alternatywne narzędzia do programowania w systemie tych mikrokontrolerów – JTAG–owy flasher str91isp oraz prosty bootloader obsługiwany przez port szeregowy.

Częstym problemem, przed jakim staje konstruktor korzystający w swoich projektach z nowych mikrokontrolerów, jest niewygodne w użyciu oprogramowanie do zapisywania pamięci Flash dostarczane przez producentów układów. Opracowany przez STMicroelectronics program CAPS ma bardzo ładny okienkowy interfejs przez co z jego obsługą bez wątpienia poradzi sobie każdy, nawet bardzo początkujący elektronik. Niestety po kilku użyciach przełączanie okienek i „przeklikiwanie się" przez kreatory staje się dokuczliwe. Nie wspomnę już o braku wersji programu dla systemów operacyjnych innych niż Jedyny Słuszny. A jeżeli chcemy coś lepszego – trzeba kupić (i zwykle słono zapłacić)...

Alternatywny flasher

Sfrustrowany takim stanem rzeczy autor niniejszego artykułu napisał własny prosty flasher, którego możliwości wymieniono w ramce. Dzięki temu, że program obsługuje się z linii poleceń, można go uruchamiać w trybie wsadowym np. z pliku *Makefile*. W środowiskach programistycznych w rodzaju WinARM+Eclipse pozwala to na programowanie układu przez naciśnięcie jednego przycisku.

Zanim przejdziemy do opisu programu – kilka słów na temat organizacji pamięci Flash w mikrokontrolerach STR91x. Składa się ona z dwóch niezależnych bloków: Flasha głównego (*main Flash*) o pojemności 256 lub 512 kB oraz Flasha pomocniczego (*secondary Flash*) o rozmiarze 32 kB, przeznaczonego dla programu bootloadera. Bloki pamięci Flash mogą zaczynać się od dowolnych adresów z zakresu `0x0 – 0x3f00000` ustawianych w rejestrach `BBADR` i `NBBADR`. Możliwe jest również ustawienie rozmiaru obydwu bloków widzianego przez procesor (rejestry `BBSR`, `NBBSR`) oraz ich włączanie i wyłączanie (rejestr `FMI_CR`) Po wyzerowaniu mikrokontrolera układ bloków pamięci w przestrzeni adresowej zależy od stanu bitu `Csx_Mapping` w sektorze konfiguracyjnym. Jeśli `Csx_Mapping = 1`, pod adresem `0x0` znajduje się początek głównego (*main*) bloku pamięci Flash, w innym wypadku – początek bloku pomocniczego (*secondary*). Umożliwia to na wybór bloku pamięci, z którego procesor rozpoczyna wykonywanie programu. Ilustruje to **rys. 1.**

Obsługa programu

Program *str91isp* obsługuje się podobnie do innych tego typu programów – należy podać nazwę pliku i co z tym plikiem zrobić:

`str91isp polecenie nazwa_pliku dodatkowe_opcje`

gdzie `poleceniem` może być:

`-p plik.bin` – programowanie wybranego obszaru pamięci Flash (*main* lub *secondary*) danymi z pliku `plik.bin`,

`-pv plik.bin` – programowanie i weryfikacja wybranego obszaru flasha,

`-r plik.bin` – odczyt wybranego bloku pamięci Flash i zachowanie go do pliku `plik.bin`

`-e` – kasowanie całego układu (łącznie z sektorami konfiguracyjnymi i lockbitami)

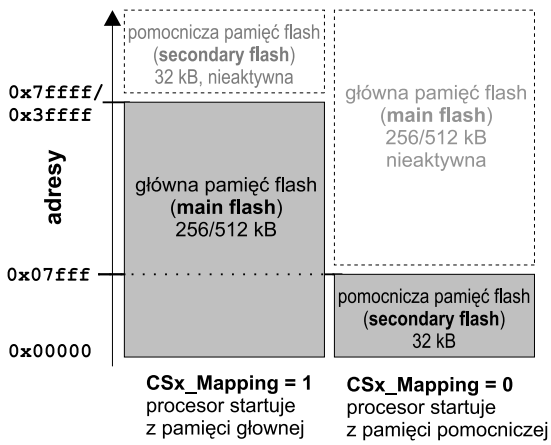
`-s` – skanowanie łańcucha JTAG – program wypisuje listę kodów identyfikacyjnych (IDCODE) kontrolerów TAP wykrytych w łańcuchu JTAG.

`-b [main/secondary]` – wybór bloku pamięci Flash, z którego procesor będzie bootował (patrz rys. 1).

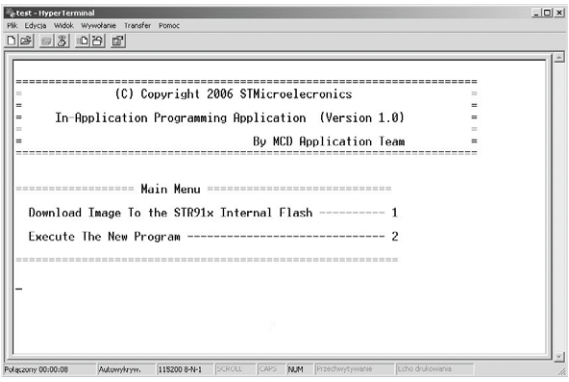
`dodatkowymi_opcjami` mogą być:

`-c [flashlink/wiggler]` – wybór rodzaju kabla LPT–JTAG, którym łączymy mikrokontroler z komputerem. Domyślną wartością jest `flashlink` (programator ST FlashLink lub jego odpowiednik – ZL18PRG firmy *Kamami.pl*). Program *str91isp* współpracuje również z klonami Wigglera.

`-d nazwa_urządzenia_LPT` – wybór urządzenia portu równoległego (domyślnie LPT1 lub /dev/parport0).



Rys. 1. Organizacja pamięci Flash po wyzerowaniu mikrokontrolera



Rys. 2. Ekran powitalny bootloadera w programie HyperTerminal

`-f [main/secondary]` – wybór bloku pamięci Flash, na którym program będzie wykonywał operację odczytu lub zapisu (rys. 1). Domyślnie *main*.

`-v opóźnienie` – spowolnienie transmisję danych przez port równoległy komputera proporcjonalnie do wartości `opóźnienia`. Opcja może być przydatna gdy łączymy programator z urządzeniem długim przewodem.

Na przykład, by zaprogramować i zweryfikować główny blok Flasha plikiem `test.bin` korzystając z kabla Wiggler podłączonego do portu LPT2 trzeba wykonać polecenie:

```
str91isp -pv test.bin
-c wiggler -d LPT2
```

Program *str91isp* wykorzystuje szybki tryb programowania mikrokontrolera (*turbo mode*), który omija w łańcuchu JTAG powolny kontroler TAP obsługujący sprzętowy debugger ARM9 umożliwiając szybkie programowanie układów. Na większości komputerów z systemem Windows, *str91isp* potrafi także automatycznie zainstalować sterowniki portu równoległego (*giveio*) jeśli plik *giveio.sys* znajduje się w tym samym katalogu co plik *str91isp.exe*. Funkcja ta może nie działać na wszystkich instalacjach Windows.

Bootloader

Tańszą alternatywą dla JTAG–a jest programowanie pamięci Flash z wykorzystaniem portu szeregowego i odpowiedniego bootloadera. Układy STR91x nie mają fabrycznie wbudowango bootloadera, ale zastosowana w nich organizacja pamięci Flash (dwa niezależne bloki) pozwala na bardzo wygodną samodzielną implementację takiego programu. Na stronie internetowej ST można znaleźć przykładowy bootloader razem z dość szczegółowym opisem (nota aplikacyjna AN2475). Jest on przeznaczony dla zestawu ewaluacyjnego firmy STMicroelectronics, lecz można go bez trudu dostosować dla dowolnego systemu uruchomieniowego. Port przedstawiony w artykule działa na zestawie ZL25ARM firmy *Kamami.pl*.

Bootloader umożliwia wyłącznie programowanie głównego bloku pamięci Flash. Nie jest do tego potrzebne żadne specjalistyczne oprogramowanie – wystarczy dowolny program terminalowy, np. HyperTerminal (Windows) lub *minicom* (systemy uniksowe). Program komunikuje się z komputerem za pomocą interfejsu RS232 o parametrach: 115200 b/s, 8 bitów danych, 1 bit stopu, brak bitu parzystości i kontroli przepływu danych. Od strony mikrokontrolera RS232 powinien być dołączony do linii P3.0 (RXD) i P3.1 (TXD).

Bootloader musi być zainstalowany w pomocniczym 32–kilobajtowym bloku pamięci Flash. W tym celu konieczne jest użycie programatora JTAG, na szczęście tylko raz. Jeśli korzystamy z *str91isp*, polecenie programowania jest następujące:

```
str91isp -pv str91_
bootloader.bin -b
secondary -f secondary.
```

Można też wykorzystać plik *Makefile* i zaprogramować procesor poleceniem make program.

Korzystanie z bootloadera jest bardzo proste:

1. Otwieramy program terminalowy i ustawiamy podane wyżej parametry transmisji.
2. Zerujemy mikroprocesor przy ustawionym stanie niskim na porcie P3.4 – w zestawie ZL25ARM należy przy wciśniętym przycisku SW0 nacisnąć przycisk RESET. Spowoduje to uruchomienie aplikacji bootloadera zamiast skoku do głównego programu.
3. Powinniśmy zobaczyć w terminalu ekran powitalny taki jak na **rys. 2.**
4. Wciskamy klawisz 1 i wysyłamy plik z naszym programem **(***Menu Transfer–>Wyślij plik*). Należy wybrać protokół *Ymodem*.
5. Jeśli wszystko przebiegnie prawidłowo pojawi się komunikat "*Programming Completed Successfully*" oraz nazwa i rozmiar wysyłanego pliku. Aby uruchomić załadowany program wystarczy wcisnąć klawisz 2 lub wyzerować mikrokontroler (przy ustawionym stanie wysokim na P3.4).

Cechy bootloadera:
– połączenie z komputerem przez RS232,
– nie wymaga żadnego specjalnego oprogramowania,
– umożliwia wyłącznie programowanie pamięci Flash.

Ponieważ bootloader dostępny na stronie ST opracowano z wykorzystaniem narzędzi komercyjnych, opracowałem port dla kompilatora GCC. Program prawidłowo kompiluje się między innymi w środowisku WinARM. Dostępna jest również prosty programik migający diodami LED w zestawie ZL25ARM, za pomocą której można sprawdzić czy bootloader działa prawidłowo.

Na koniec jeszcze jedna ważna sprawa: aplikacja, która będzie ładowana z użyciem bootloadera nie powinna zmieniać konfiguracji pamięci Flash (zapis do rejestrów `NBBSR`, `BBSR`, `NBADR`, `BBADR`, `FMI_CR`). Inicjalizacja kontrolera Flash jest zwykle wykonywana w kodzie startowym kompilatora C (pliki *startup.S*, *startup_STR91x.S*). Linie odpowiedzialne za nią należy usunąć, ponieważ za właściwe ustawienia kontrolera Flash odpowiada bootloader. Aplikacja, która wykonuje taką inicjalizację i działa poprawnie załadowana do głównego bloku Flasha przez JTAG **nie będzie** działać po wgraniu z wykorzystaniem bootloadera.

Program *str91isp*, bootloader oraz przykładowy program wraz z kodami źródłowymi będzie można znaleźć na płycie CD–EP6/2007B dołączonej do EPoL, publikujemy je także na stronach *http://wlostowski.ep.com.pl* oraz *http://arm.ep.com.pl*.

Tomasz Włostowski, EP
tomasz.wlostowski@ep.com.pl

Cechy programu *str91isp*:
– połączenie z mikrokontrolerem za pomocą portu równoległego i kabla ST FlashLink, ZL18PRG lub Wiggler,
– sterowanie z linii poleceń,
– szybkie programowanie, odczyt i weryfikacja pamięci Flash,
– kasowanie zawartości całej pamięci,
– nie wymaga ładowania sterowników portu LPT typu *giveio*, *porttalk*,
– pracuje pod kontrolą Windows (2000/XP) lub Linuksa,
– kod źródłowy dostępny na licencji GPL.