Mikrokontrolery STR91x od podstaw, część 3

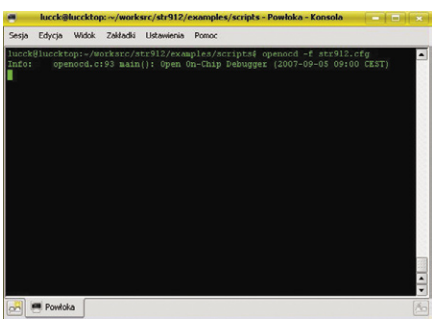
Programowanie Flash, pierwszy projekt w Eclipse

Jest to ostatnia część „teoretyczna" naszego kursu. Za miesiąc pokażemy kilka pierwszych przykładów obsługi bloków peryferyjnych stosowanych w mikrokontrolerach STR91x.

Programowanie pamięci Flash w mikrokontrolerach STR91x

Czasy programowania pamięci Flash mikrokontrolerów za pomocą specjalizowanych programatorów równoległych minęły bezpowrotnie, podobnie jak zewnętrznych pamięci EEPROM. Obecnie mikrokontroler nie mający możliwości programowania w systemie docelowym (IAP) nie ma zupełnie racji bytu. Mikrokontrolery rodziny STR912 mogą być programowane w systemie tylko poprzez złącze JTAG. Istnieje co prawda możliwość napisania własnego loadera, umożliwiającego programowanie w systemie za pomocą na przykład portu szeregowego, niemniej jednak do pierwszego programowania niezbędne jest posiadanie interfejsu JTAG. Aby zaprogramować pamięć Flash mikrokontrolera należy dołączyć interfejs JTAG (zgodny ze standardem OpenOCD) do złącza ZL1 płytki ZL24ARM oraz do złącza USB komputera PC. Następnie należy uruchomić program obsługi interfejsu JTAG wydając polecenie *openocd –f str912.cfg* (w Windows należy użyć polecenia *openocd–ft2xx*). W przypadku Windows należy również w pliku konfiguracyjnym *str912.cfg* odznaczyć linię *ft2232_device_desc „Dual RS232 A"*.

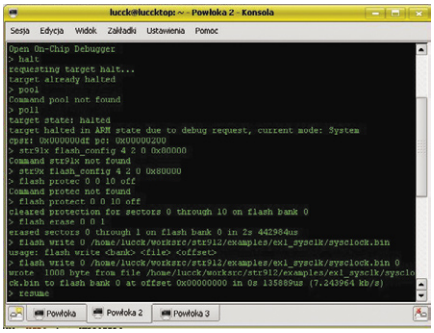
Jeżeli wszystko pójdzie dobrze, w wierszu polecenia powinniśmy



Rys. 8.

ujrzeć komunikat jak na **rys. 8**, co świadczy o prawidłowym uruchomieniu programu sterującego interfejsem JTAG. Teraz aby zaprogramować pamięć Flash mikrokontrolera musimy połączyć się z programem *OpenOCD* wydając w terminalu polecenie *telnet 127.0.0.1 4444*. Teraz żeby zaprogramować pamięć Flash musimy wydać odpowiednią sekwencję komend. Najpierw należy zatrzymać procesor wydając komendę *halt*, a następnie sprawdzić czy procesor został zatrzymany instrukcją *poll*. Jeżeli zobaczymy komunikat *target halted*, możemy przejść do dalszych czynności, do których należy konfiguracja pamięci Flash za pomocą polecenia *str9x Flash_config 4 2 0 0x80000*. Następnie należy odbezpieczyć poszczególne sektory pamięci Flash za pomocą polecenia *flash protect 0 0 1 off*, gdzie ostatnie parametry 0 i 1 oznaczają adres początkowego i końcowego banku. Należy tutaj podać odpowiednią liczbę banków w zależności od rozmiaru pliku.

Po odbezpieczeniu odpowiednich banków pamięci Flash należy je skasować za pomocą polecenia *Flash erase 0 0 1*, gdzie dwa ostatnie parametry określają pierwszy i ostatni sektor pamięci Flash do skasowania. Gdy już mamy skasowaną pamięć, możemy teraz przystąpić do jej zaprogramowania za pomocą polecenia *Flash write 0 nazwa_pliku.bin 0*, gdzie dwa ostatnie parametry określają nazwę pliku do zaprogramowania oraz adres początkowy pamięci, gdzie plik będzie zapisany (w tym przypadku jest to początek pamięci Flash).



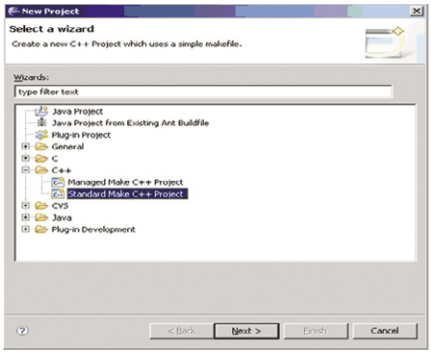
Rys. 9. Przebieg sesji programowania pamięci Flash

Gdy mamy już zaprogramowaną pamięć należy rozpocząć ponowne wykonanie programu przez STR91x wydając polecenie resume 0. Na **rys. 9** przedstawiono przebieg sesji programowania pamięci Flash mikrokontrolera.

Programowanie pamięci Flash poprzez wpisywanie rozkazów dla OpenOCD jest uciążliwe, dlatego dobrze jest zautomatyzować ten proces. Można to zrobić za pomocą skryptów dla OpenOCD: wystarczy na końcu pliku *str912.cfg* dopisać linię *target_script 0 reset pgm.script* oraz stworzyć plik *pgm.script* zawierający listę komend do wykonania, na przykład:

```
halt
poll
str9x Flash_config
4 2 0 0x80000
Flash protect 0 0 1 off
Flash erase 0 0 1
Flash write 0 /home/lucck/
worksrc/str912/examples/ex1_
sysclk/sysclock.bin 0
resume
shutdown
```

Teraz wystarczy jedynie wywołać polecenie *openocd –f str912s.cfg* i pa-



Rys. 10. Okno wyboru typu projektu programu Eclipse

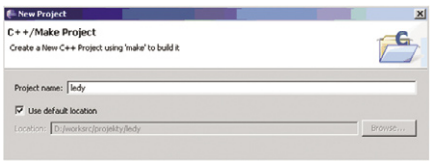
mięć mikrokontrolera zostanie zaprogramowana automatycznie. Niestety rozwiązanie to nie jest pozbawione wad, do których należą głównie „sztywne" przypisanie wszystkich parametrów w pliku skryptowym oraz konieczność włączania i wyłączania programu OpenOCD.

Praca w środowisku Eclipse – pierwszy projekt

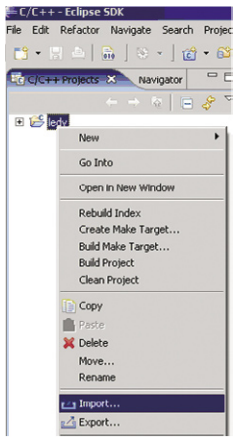
Autor zdaje sobie sprawę, że większość Czytelników przyzwyczajona jest do korzystania z IDE, dlatego w dalszej części cyklu będziemy się posługiwać środowiskiem Eclipse. Nic nie stoi na przeszkodzie, aby zaprezentowane przykłady kompilować w wierszu polecenia, a do pisania kodów źródłowych wykorzystać dowolny inny edytor.

W celu wstępnego zapoznania Czytelników ze środowiskiem Eclipse zaznajomimy się ze sposobem uruchomienia programu przykładowego, zapalającego kolejno diody LED umieszczone na płytce zestawu ZL25ARM (*kamami.pl*), tworząc efekt węża świetlnego.

Na początek opiszemy, w jaki sposób stworzyć projekt w *Eclipse*, a następnie zaimportujemy do niego wszystkie pliki z przykładowego projektu błyskającego diodami, skompilujemy go oraz zaprogramujemy pamięć Flash mikrokontrolera obserwując efekt działania programu. Przed rozpoczęciem pracy należy rozpakować wszystkie przykłady do katalogu roboczego



Rys. 11. Okno wyboru nazwy projektu w programie Eclipse



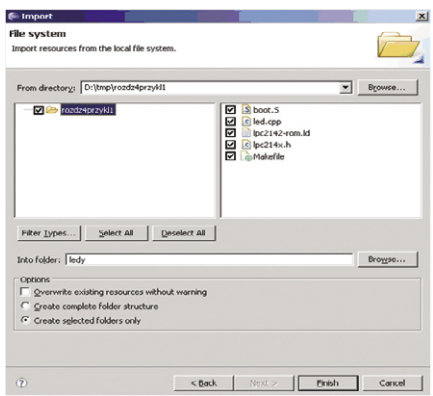
Rys. 12. Importowanie istniejącego projektu do Eclipse

projektów Eclipse. Aby stworzyć projekt uruchamiamy *Eclipse*, a następnie wybieramy z menu polecenie *File>New>Project*. Pokaże się wówczas okno dialogowe (**rys. 10**), z którego wybieramy opcję: *Standard Make C Project*.

Następnie klikamy przycisk *Next*, pojawi się wówczas kolejne okno dialogowe służące do wpisania nazwy projektu (**rys. 11**).

W polu tekstowym *Project Name* wpisujemy nazwę projektu *ex1_sysclk* i wciskamy *Finish*, co spowoduje utworzenie projektu. Ponieważ katalog o tej nazwie już istnieje i zawiera wszystkie pliki zawarte w przykładzie, zostaną one automatycznie dołączone do Eclipse i nie musimy już nic więcej robić. Natomiast, gdy tworzymy nowy projekt i chcemy zaimportować pliki *makefile* i startowe z wzorcowego projektu, możemy to zrobić klikając prawym przyciskiem myszy na otwartym projekcie (tak jak przedstawiono na **rys. 12**), a następnie wybrać polecenie *import*.

Pojawi się wówczas okno dialogowe, z którego wybieramy opcję

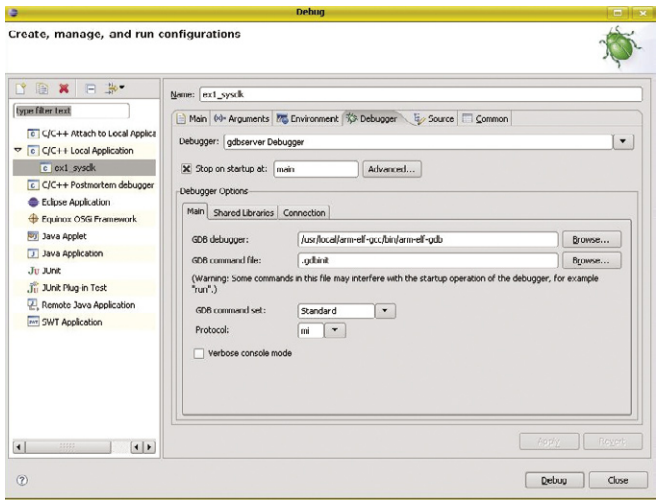


Rys. 13. Import przykładowego projektu do środowiska Eclipse

File System. Zobaczymy wówczas kolejne okno, w którym klikamy przycisk *Browse* i wybieramy katalog z przykładowym projektem. Po prawej stronie pokaże się lista plików znajdujących się w archiwum (**rys. 13**).

Możemy tutaj selektywnie wybierać pliki, które będą potrzebne w naszym nowym projekcie. Gdy mamy już zaimportowany projekt możemy przystąpić do kompilacji, co możemy zrobić wydając z menu polecenie *Project>Build All*. Jeżeli cały proces kompilacji przebiegł pomyślnie, wówczas powstanie plik wynikowy *sysclock.bin*, natomiast w przypadku wystąpienia błędu podczas kompilacji na dole w zakładce *Problems* pojawi się opis zawierający przyczynę błędu. Po kliknięciu na wybranym komunikacie zostaniemy przeniesieni do linii, w której wystąpił błąd. Bardzo ciekawą i przydatną funkcją Eclipse jest system inteligentnych podpowiedzi, które informują programistę, jakie metody i pola znajdują się w danej klasie/strukturze. Po prawej stronie jest umieszczona zakładka *Outline*, w której w sposób graficzny przedstawione są poszczególne klasy, zmienne, funkcje definicje itp., co jest również bardzo pomocne na etapie pisania oprogramowania. Gdy mamy już gotowy plik wynikowy przystępujemy do programowania pamięci Flash mikrokontrolera, za pomocą programu *OpenOCD*. W tym celu przechodzimy do katalogu projektu *ex1_sysclk* i wydajemy polecenie *make program*. Po kilku sekundach potrzebnych do zaprogramowania pamięci na diodach LED (D4... D11) zestawu ZL25ARM powinien ukazać się efekt węża świetlnego.

Prezentowane środowisko oprócz możliwości wygodnego pisania programu umożliwia również debugowanie programu. Aby było to możliwe, musimy w pliku *makefile* włączyć opcję DEBUG=y oraz wyłączyć optymalizację kompilatora OPT=0, a następnie skompilować program i zaprogramować pamięć Flash procesora. Następnie należy z katalogu *scripts* do katalogu projektu skopiować plik *.gdbinit* zawierający polecenia inicjalizacyjne dla debugera *gdb*, a następnie w środowisku Eclipse z menu wybrać opcję *Run>Debug*. Na ekranie pojawi się wówczas okno konfiguracyjne (**rys. 14**), w którym należy wejść

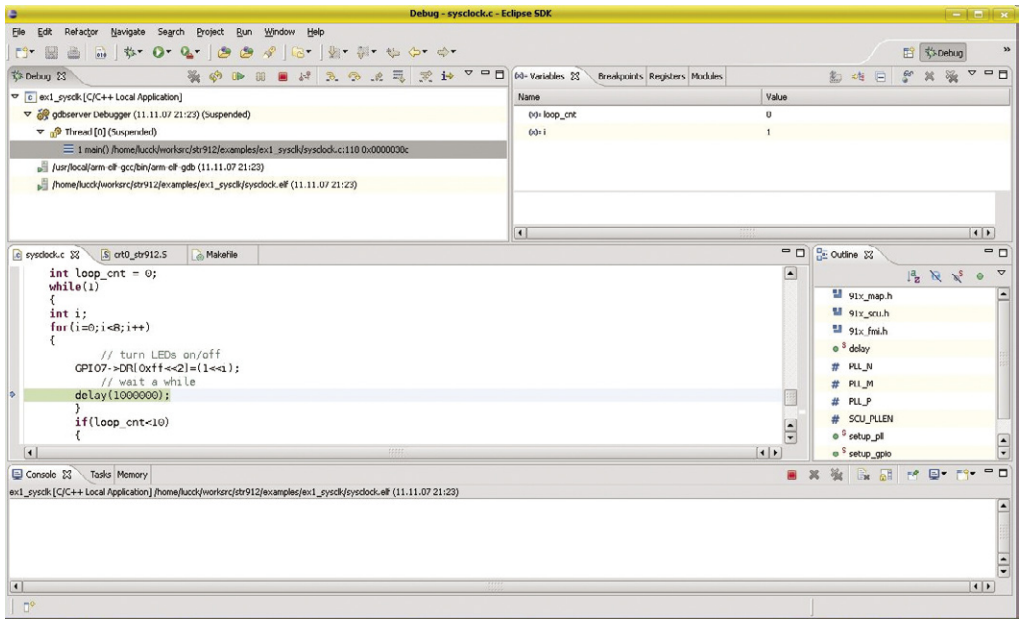


Rys. 14. Okno konfiguracyjne debugera

na zakładkę *Debug* i w polu *Debuger* ustawić zdalny debuger *gdb ServerDebuger* oraz podać ścieżkę do programu *arm–elf–gdb*. Następnie należy przejść do pod zakładki *Connection* i wybrać opcję umożliwiające połączenie się z programem OpenOCD.

Gdy już mamy wszystko skonfigurowane, należy uruchomić program *OpenOCD (OpenOCD –f str912.cfg)*, a następnie wcisnąć przycisk *DEBUG*. Po chwili Eclipse powinno przejść w tryb debugowania, dzięki czemu możemy śledzić pracę programu, podobnie jak w większości innych zintegrowanych środowisk programistycznych. Mamy możliwość ustawiania pułapek w określonych miejscach programu, podglądania zawartość zmiennych, itp.

Z uwagi na to, że debuger wykorzystuje pułapki sprzętowe należy uważać, aby w danym momencie były aktywne co najwyżej dwie pułapki. Na **rys. 15** przedstawiono wygląd okna Eclipse podczas sesji debugowania przykładowego programu.



Rys. 15. Okno Eclipse podczas sesji debugowania programu

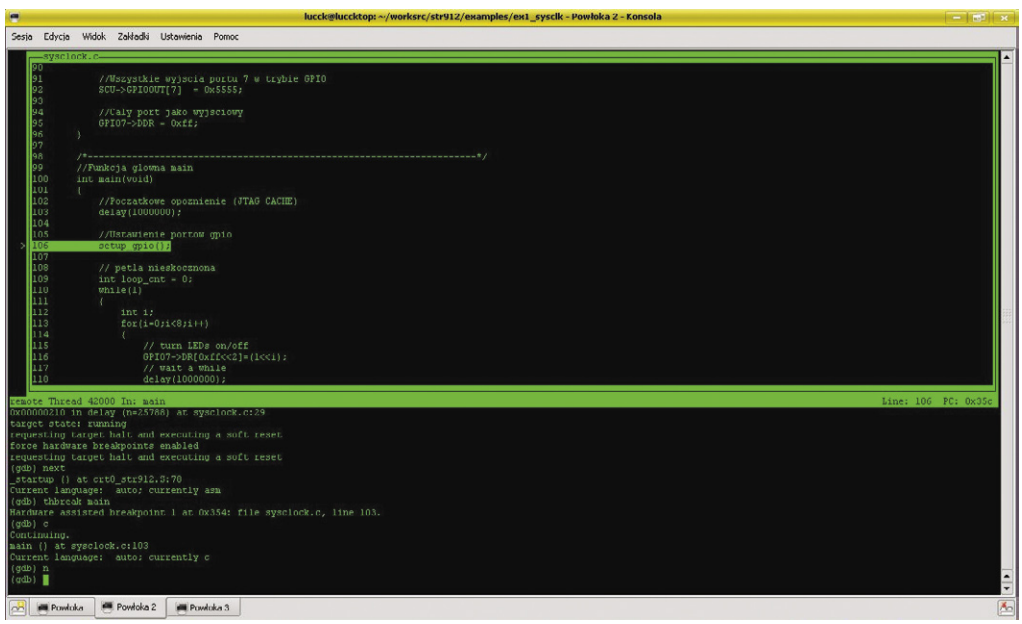
Dla osób, które są – podobnie jak autor – „uczulone” na narzędzia graficzne oraz system Windows, istnieje możliwość debugowania programów bezpośrednio za pomocą wygodnego konsolowego debugera *arm–elf–gdbtui*. W celu debugowania przykładowego programu uruchamiamy debuger w terminalu za pomocą polecenia *arm–elf–gdbtui sysclock.elf*, następnie zakładamy tymczasową pułapkę sprzętową *thbreak main* oraz kontynuujemy wykonanie programu wpisując *continue*. Gdy program wejdzie do funkcji *main*, wykonanie jego zostaje zatrzymane i możemy debugować program, wydając odpowiednie polecenia tekstowe. Do najważniejszych poleceń należą:

– *thbreak* ustawiającą tymczasową pułapkę sprzętową,

– *step* umożliwiającą wykonanie jednego kroku programu z wejściem do wnętrza funkcji,

– *next* umożliwiającą wykonanie programu bez wchodzenia do wnętrza funkcji.

Na **rys. 16** przedstawiono program *arm–elf–gdbtui* podczas śledzenia przykładowego projektu.



Rys. 16. Okno programu *arm–elf–gdbtui* podczas śledzenia przykładowego projektu

Lucjan Bryndza, EP
lucjan.bryndza@ep.com.pl