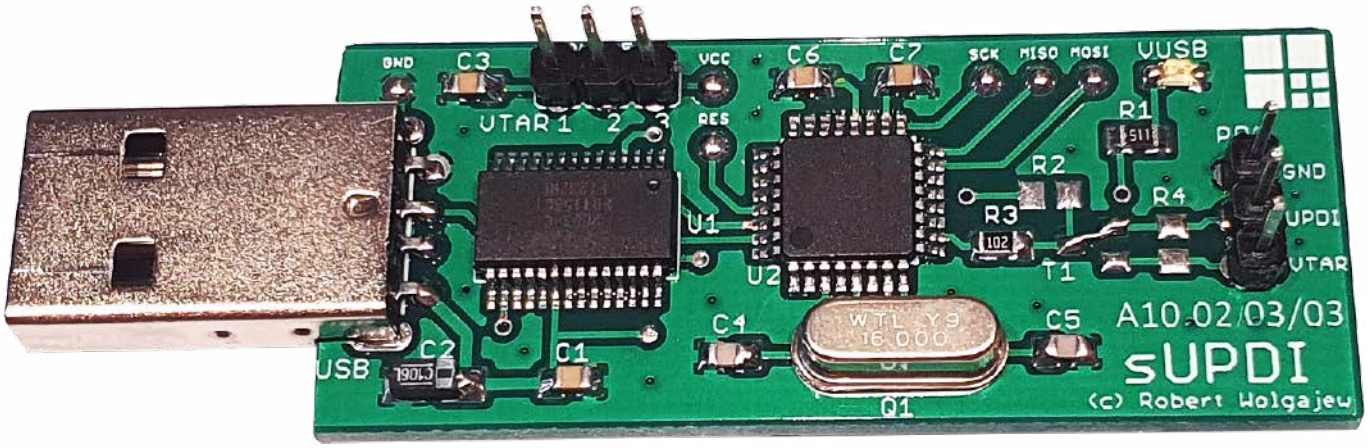


sUPDI – programator UPDI do mikrokontrolerów AVR

Ciągły rozwój mikrokontrolerów powoduje, że całkiem skomplikowane zagadnienia jesteśmy dzisiaj w stanie zrealizować z użyciem coraz prostszych (z punktu widzenia użytkownika) i tańszych układów. Dobrze to widać na przykładzie popularnych mikrokontrolerów z rodziny AVR, które po przejęciu firmy Atmel przez firmę Microchip dostały niejako drugie życie i przechodzą w tej chwili kolejny upgrade typów i możliwości. To całe pozytywne zamieszanie ma jednak także swoją cenę, którą jest wprowadzenie zupełnie nowego interfejsu programowania i debugowania pod postacią sprzęgu UPDI.

Wystarczy wspomnieć o nowych mikrokontrolerach ATtiny serii 0 i serii 1, które integrują w sobie możliwości dotychczas zarezerwowane dla większych braci z serii Xmega, w tym system zdarzeń, przetworniki ADC i DAC oraz peryferia CIP niezależne od rdzenia (*Core Independent Peripherals*). Dodatkowo dysponują sporą ilością pamięci Flash, RAM oraz EEPROM i dostępne są w niewielkich obudowach. Podobnie wygląda sytuacja w przypadku rodziny Xmega i Mega, gdzie wprowadzono szereg udoskonaleń i rozszerzeń funkcjonalności oraz nieznane dotąd peryferia. Mowa o takich układach, jak MegaAVR serii 0 czy AVR-DA.

Nowy interfejs programowania i debugowania UPDI (*Unified Program and Debug Interface*) ma tę unikalną cechę, że angażuje wyłącznie jedno wyprowadzenie mikrokontrolera (UPDI/RESET) w celu przeprowadzenia procesu programowania i debugowania układu. Producent przewidział stosowne oprogramowanie i wsparcie sprzętowe dla nowego standardu. Oprogramowanie Atmel Studio (a co za tym idzie także nowe Microchip Studio) wspiera wszystkie nowe mikrokontrolery i obsługę nowego, niedrogiego programatora pod nazwą MPLAB SNAP, który jest wyposażony w interfejs UPDI.

Jednak sytuacja się komplikuje w przypadku używania innych środowisk IDE, jak np. Eclipse z pluginem AVR dającym możliwość programowania mikrokontrolerów z użyciem aplikacji AVRdude. Co prawda w przypadku AVRdude i nowego interfejsu UPDI możemy skorzystać z obsługiwanego przezeń programatora Atmel ICE (uprzednio zaktualizowanego), jednak dla wielu konstruktorów hobbystów jest to rozwiązanie bardzo drogie, a co za tym idzie, w zasadzie niedostępne.

Czy w takim razie jesteśmy zdani wyłącznie na oprogramowanie firmy Atmel? Czy musimy rezygnować z wygodnego środowiska programistycznego, jakim bez wątpienia jest Eclipse? Na szczęście nie. Z pomocą przychodzi nam społeczność serwisu GitHub, a dokładnie użytkownik o nicku ElTangas. Przygotował on specjalną wersję programu AVRdude (a w zasadzie nowy plik konfiguracyjny) dostępną pod adresem https://bit.ly/2RCw15P oraz nowy typ obsługiwanego programatora oznaczony jtag2updi. Dzięki temu i przy użyciu prostego programatora

Dodatkowe materiały do pobrania ze strony www.media.avt.pl

W ofercie AVT* AVT5863

Podstawowe parametry:
- współpracuje z nowym interfejsem programowania i debugowania UPDI,
- pozwala programować większość nowych układów AVR z pomocą programu AVRdude i Eclipse,
- łatwa konfiguracja i obsługa

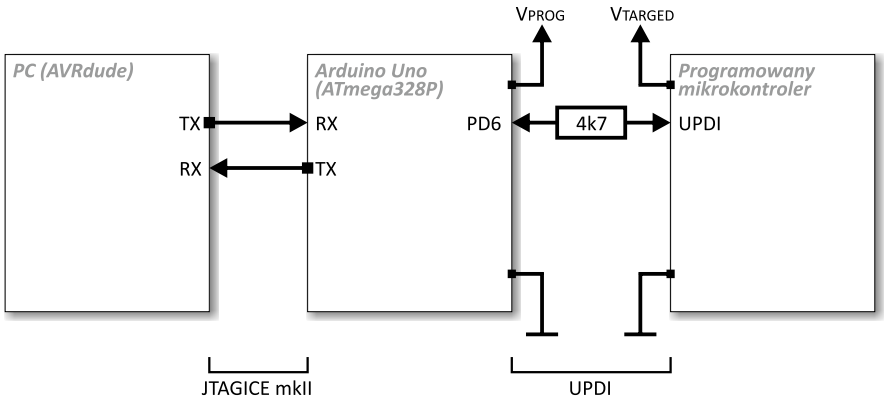
Projekty pokrewne na www.media.avt.pl:

AVT-5848	Minimoduł z mikrokontrolerem LPC845 (EP 3/2021)
AVT-5829	Minimoduł z mikroprocesorem LPC802 (EP 12/2020)
AVT-5726	Rysino – płytka ewaluacyjna z FPGA Intel MAX10 (EP 11/2019)
AVT-5574	Płytka ewaluacyjna dla STM32F2/F4/F7 do celów SDR i nie tylko (EP 2/2017)

*** Uwaga! Elektroniczne zestawy do samodzielnego montażu. Wymagana umiejętność lutowania!**
Podstawową wersją zestawu jest wersja [**B**] nazywana potocznie KIT-em (z ang. zestaw). Zestaw w wersji [**B**] zawiera elementy elektroniczne (w tym [**UK**] – jeśli występuje w projekcie), które należy samodzielnie wlutować w dołączoną płytkę drukowaną (PCB). Wykaz elementów znajduje się w dokumentacji, która jest podlinkowana w opisie kitu.
Mając na uwadze różne potrzeby naszych klientów, oferujemy dodatkowe wersje:
- wersja [**C**] – zmontowany, uruchomiony i przetestowany zestaw [**B**] (elementy wlutowane w płytkę PCB)
- wersja [**A**] – płytka drukowana bez elementów i dokumentacji

Kity w których występuje układ scalony wymagający zaprogramowania, mają następujące dodatkowe wersje:
- wersja [**A+**] – płytka drukowana [**A**] + zaprogramowany układ [**UK**] i dokumentacja
- wersja [**UK**] – zaprogramowany układ

Nie każdy zestaw AVT występuje we wszystkich wersjach! Każda wersja ma załączony ten sam plik pdf! Podczas składania zamówienia upewnij się, którą wersję zamawiasz!
http://sklep.avt.pl. W przypadku braku dostępności na *http://sklep.avt.pl*, osoby zainteresowane zakupem płytek drukowanych (PCB) prosimy o kontakt via e-mail: kity@avt.pl.



Rysunek 1. Schemat blokowy programatora UPDI z użyciem płytki Arduino Uno

Wykaz elementów:

Rezystory: (SMD0805)
R1: 510 Ω
R2: 4,7 kΩ

Kondensatory:
C1, C3, C6, C7: 100 nF (SMD0805)
C2: tantalowy 10 µF/10 V (SMD A)
C4, C5: 22 pF (SMD0805)

Półprzewodniki:
U1: FT232RL (SSOP28)
U2: ATmega88 (TQFP32)
VUSB: dioda LED czerwona (SMD0805)

Pozostałe:
PRG, VTAR: goldpin 3×1
Q1: rezonator kwarcowy 16 MHz niski
USB: gniazdo męskie USB-A

Ustawienia Fusebitów:

```
CKSEL3...0: 1111
SUT1...0: 11
CKDIV8: 1
CKOUT: 1
DWEN: 1
EESAVE: 0
```

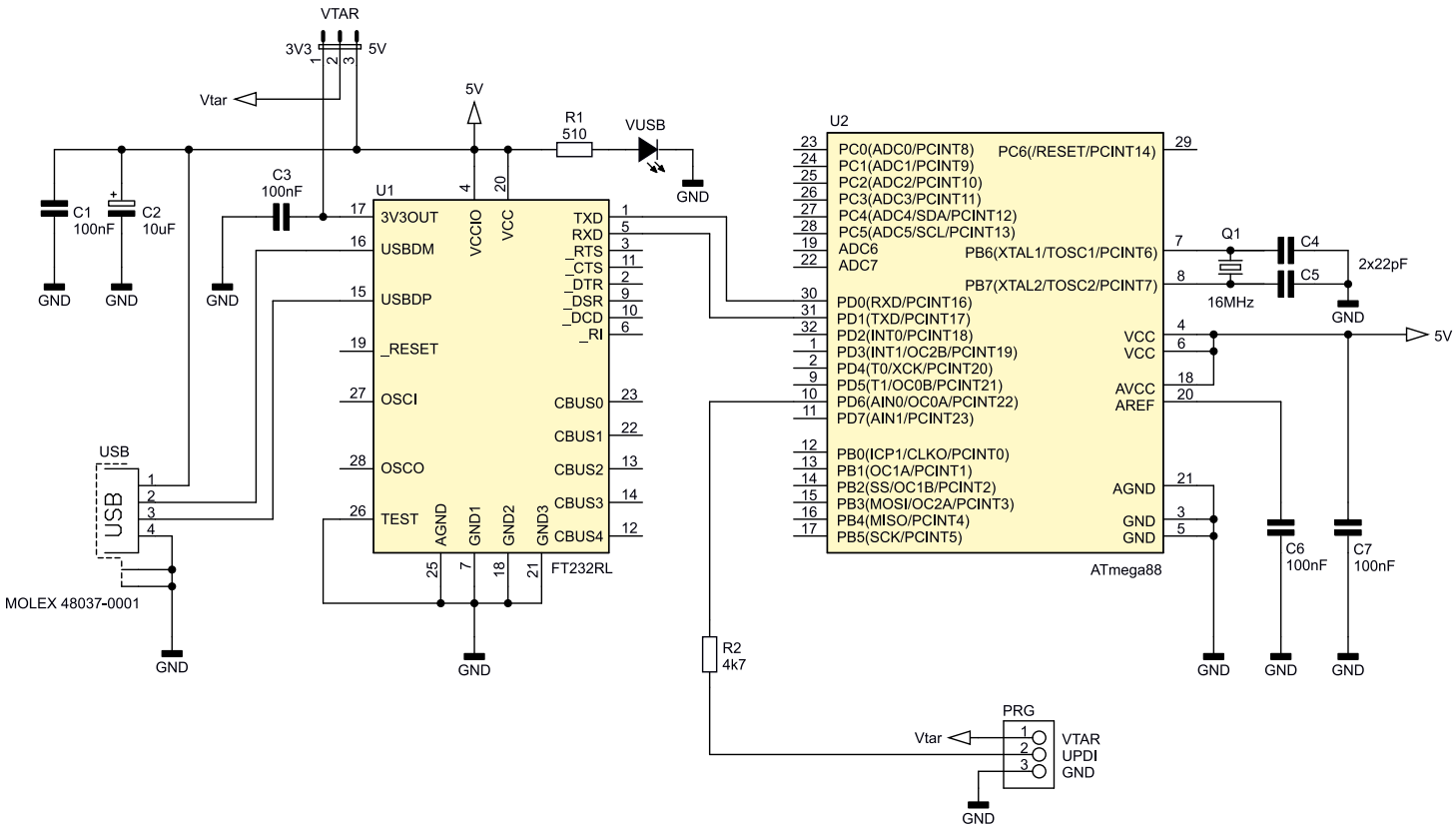
jesteśmy w stanie programować większość nowych układów AVR wyposażonych w interfejs UPDI za pomocą programu AVRdude.

Oprócz oprogramowania niezbędnego po stronie PC (AVRdude) potrzebny jest prosty układ, który protokół JTAGICE mkII stosowany po stronie PC transkoduje do nowego protokołu interfejsu UPDI. Autor oprogramowania postanowił wykonać konstrukcję na bazie popularnej płytki Arduino Uno/Nano i mikrokontrolera ATmega328P, jednak dla osób chcących poeksperymentować z innymi typami układów, przy udziale których można skonstruować wspomniany programator, autor udostępnił także całe oprogramowanie źródłowe w postaci repozytorium serwisu GitHub.

Budowa i działanie

Schemat blokowy i funkcjonalny całego toru pomiędzy oprogramowaniem na PC, programatorem zawierającym mikrokontroler ATmega328P i programowanym układem z interfejsem UPDI został pokazany na **rysunku 1**. Bazując na tym prostym, ale bardzo ciekawym schemacie blokowym, zaprojektowałem prosty programator o nazwie sUPDI, którego schemat został pokazany na **rysunku 2**. Powstał bardzo prosty system mikroprocesorowy zgodny w założeniach z projektem użytkownika ElTangas z pewnym rozszerzeniem funkcjonalności. Rozszerzenie, o którym mowa, to prosty sprzęg USB-Serial w postaci dobrze znanego układu FT232RL, który wraz ze standardowym sterownikiem VCP stanowi w systemach operacyjnych PC wirtualny port szeregowy.

Ponadto, podobnie jak w oryginalnym rozwiązaniu, zaimplementowano prosty translator napięć z użyciem rezystora R2, który pozwala na programowanie mikrokontrolera zasilanego innym napięciem, niż układ ATmega88, stanowiący element programatora. Ten ostatni, z uwagi na dość wysoką częstotliwość taktowania równą 16 MHz, zasilono napięciem 5 V dostępnym z gniazda USB. Dodatkowo, zastosowano JUMPER, za pomocą którego można ustawić poziom napięcia zasilania VTAR (3,3 V lub 5 V) na wyjściu interfejsu UPDI programatora (złącze PRG). Oczywiście sam interfejs składa się wyłącznie z jednego wyprowadzenia UPDI (i masy), jednak dodatkowy pin zasilania daje możliwość zasilania programowanego układu z naszego programatora. Należy jedynie pamiętać o małej wydajności prądowej tego źródła napięcia, wynoszącej jedynie



Rysunek 2. Schemat ideowy programatora sUPDI

50 mA w przypadku napięcia 3,3 V i 100 mA w przypadku napięcia 5 V.

Wspomniany wcześniej rezystor R2 pełni w naszym układzie jeszcze jedną funkcję, a mianowicie służy zabezpieczeniu programatora, gdyby w tym samym czasie nastąpiła próba nadawania transmisji przez oba układy (nasz mikrokontroler ATmega88 oraz programowany układ z interfejsem UPDI). Już tylko dla porządku dodam, że wbudowana dioda LED VUSB służy do sygnalizacji napięcia interfejsu USB.

Montaż i uruchomienie

Schemat montażowy urządzenia sUPDI został pokazany na **rysunku 3**. Zaprojektowano bardzo zgrabną płytkę drukowaną ze zdecydowaną przewagą elementów SMD montowanych wyłącznie po stronie TOP obwodu drukowanego. Aplikację urządzenia rozpoczynamy od przylutowania układów scalonych. Proces ten najłatwiej wykonać przy użyciu stacji lutowniczej na gorące powietrze (Hot Air) i odpowiednich stopów lutowniczych.

Jeśli jednak nie dysponujemy tego rodzaju sprzętem, można również zastosować metodę z użyciem typowej stacji lutowniczej czy nawet zwykłej lutownicy. Najprostszym sposobem montażu elementów o tak dużym zagęszczeniu wyprowadzeń, niewymagającym jednocześnie stosowania specjalistycznego sprzętu, jest użycie stacji lutowniczej, dobrej jakości cyny z odpowiednią ilością topnika oraz dość cienkiej plecionki rozlutowniczej. Plecionka umożliwi usunięcie nadmiaru cyny spomiędzy wyprowadzeń układów. Należy przy tym uważać, by nie uszkodzić termicznie montowanych elementów.

W następnej kolejności lutujemy pozostałe elementy półprzewodnikowe, potem rezystory i kondensatory a na samym końcu rezonator kwarcowy, listwy goldpinów VTAR (wyposażone w jumper) i PRG oraz gniazdo USB. Z uwagi na zagęszczenie wyprowadzeń układów scalonych przed pierwszym podłączeniem układu do zasilania należy jeszcze raz sprawdzić jakość wykonanych połączeń, aby nie dopuścić do ewentualnych zwarć. Wspomniana kontrola będzie znacznie łatwiejsza, jeśli zmontowaną płytkę przemyjemy alkoholem izopropylowym w celu wypłukania nadmiaru kalafonii lutowniczej.

Na **fotografii tytułowej zostało** pokazane zmontowane urządzenie (od strony warstwy TOP) w wersji prototypowej różniącej się nieznacznie od projektu docelowego.

Obsługa

Na koniec kilka słów na temat obsługi urządzenia. Pierwszym krokiem, jaki musimy wykonać, jest podłączenie programatora sUPDI do dowolnego portu USB komputera PC. Pierwszemu podłączeniu powinno towarzyszyć zainstalowanie stosownego sterownika wirtualnego portu szeregowego VCP pod nazwą COMx (gdzie x jest liczbą) dostępną, dla przykładu, z panelu sterowania systemu Windows. Warto zapamiętać nazwę naszego wirtualnego portu szeregowego, gdyż będzie niezbędna do wykonania dalszych czynności.

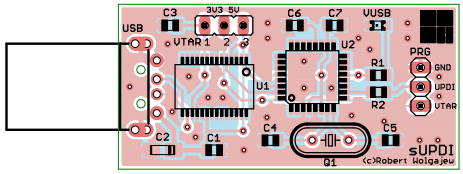
W przypadku, gdy używamy środowiska Eclipse i aplikacji AVRdude, wykorzystujemy:

- zaktualizowaną wersję aplikacji AVRdude (w tej chwili 6.3) oraz jej pliku konfiguracyjnego (*avrdude.conf*) obsługującą nowy typ programatora (jtag2updi) oraz definicje nowych typów mikrokontrolerów AVR,
- nową wersję toolchain'a z pakietu Atmel z obsługą nowych typów mikrokontrolerów, którą wskażemy w środowisku Eclipse.

Najprostszym sposobem pozyskania obu niezbędnych pakietów będzie ściągnięcie darmowej, najnowszej wersji aplikacji Arduino oraz oprogramowania Atmel Studio (lub Microchip Studio) w jego najnowszej kompilacji. Pierwsze ze środowisk da nam najnowszą wersję aplikacji AVRdude, zaś drugie, najnowszą wersję toolchaina, niezbędną do korzystania ze środowiska Eclipse. Mając wspomniane oprogramowanie, wystarczy wskazać stosowne ścieżki dostępu w środowisku Eclipse i już możemy cieszyć się możliwością tworzenia aplikacji na najnowsze mikrokontrolery AVR oraz ich programowania, za pomocą aplikacji AVRdude.

Warto zauważyć, że dostępna lista typów mikrokontrolerów, na jakie możemy przygotowywać oprogramowanie przy użyciu środowiska Eclipse, nie wyczerpuje wszystkich możliwości, jakie daje nam nowy toolchain i sam program AVRdude. Może to wynikać z nie najnowszej, i niestety nierozwijanej już, wersji pluginu AVR. No cóż, na tę chwilę nic więcej z tym nie zrobimy. Wróćmy zatem do zagadnienia programowania nowych mikrokontrolerów przy użyciu naszego programatora będącego najtańszą opcją sprzętową.

Samo programowanie przy udziale aplikacji AVRdude możemy wykonać z poziomu środowiska Eclipse, konfigurując stosowny programator (jtag2updi). Niemniej jednak lepiej wykonać tę operację ręcznie, w trybie konsoli, co szczególnie polecam. Nie powinno to stanowić jakiegokolwiek problemu, gdyż stosowne polecenia są bardzo proste i bez problemu możemy zapamiętać tych kilka szczegółów. Zacznijmy więc od przetestowania połączenia AVRdude z programowanym mikrokontrolerem przy użyciu programatora sUPDI. Stosowne polecenie (dla



Rysunek 3. Schemat płytki PCB z rozmieszczeniem elementów

przykładowego procesora ATtiny1614) wygląda następująco:

```
avrdude -c jtag2updi -P comx -p
t1614
```

gdzie *comx* to nazwa wirtualnego portu szeregowego naszego programatora sUPDI.

Jeśli wszystko przebiegnie poprawnie, powinniśmy otrzymać następujący (lub zbliżony) komunikat:

```
avrdude: AVR device initialized
and ready to accept instructions

Reading | ####################
############################## |
100% 0.03s

avrdude: Device signature =
0x1e9422 (probably t1614)

avrdude done.  Thank you.
```

Następnie wpisujemy polecenie pozwalające na zapis pliku *plik.hex* do pamięci Flash mikrokontrolera (w tym wypadku ATtiny1614), które wygląda następująco:

```
avrdude -c jtag2updi -P comx -p
t1614 -U flash:w:plik.hex
```

W tym miejscu pora na polecenie pozwalające na zapis fusebitu fuse6 mikrokontrolera (w tym wypadku ATtiny1614) na wartość 0x00:

```
avrdude.exe -c jtag2updi -P comx
-p t1614 -U fuse6:w:0x00:m
```

I podobne polecenie, ale służące do odczytu tego fusebitu i zapisanie jego wartości do pliku HEX:

```
avrdude.exe -c jtag2updi -P comx
-p t1614 -U fuse6:r:"fuse6.hex":i
```

Jest to pewna różnica w porównaniu do wcześniejszych mikrokontrolerów AVR, gdyż najnowsze modele konfigurację sprzętową przechowują w większej liczbie fusebitów, niż to miało miejsce w przypadku starszych konstrukcji.

Tyle w kwestii podstawowych poleceń. Po szczegóły odsyłam do aplikacji AVRdude, której opis dostarcza szczegółowych informacji na temat wymaganej składni poleceń. Polecam również śledzenie repozytorium GitHub pod adresem https://bit.ly/2RCw15P, gdyż autor tej implementacji może umieszczać pod wskazanym adresem najnowsze oprogramowanie wspierające nowe funkcjonalności.

Robert Wołgajew, EP