# Kurs programowania mikrokontrolerów PIC (11)

## TCPMaker – Łatwiej już się nie da

**Aplikacje, w których sterownik zbudowany w oparciu o mikrokontroler pracuje z interfejsem Ethernet i stosem TCP/IP są coraz bardziej popularne. Zaletą stosowania takiego rozwiązania jest zestandaryzowany interfejs fizyczny o dużej przepustowości oraz standardowe protokoły komunikacyjne. Pozwala to bez większych problemów dołączyć sterownik do lokalnej sieci Ethernet i dalej, do Internetu. Możliwość sterowania z dowolnego miejsca na Ziemi jest nie do przecenienia, ale równie ważna jest możliwość wykorzystania jako interfejsu sterowania i nadzoru dowolnego urządzenia wyposażonego w przeglądarkę internetową.**

Wystarczy wyposażyć nasz sterownik w funkcję serwera HTTP i można go kontrować za pomocą strony internetowej z komputera, laptopa, czy telefonu komórkowego. Gotowe są wszystkie mechanizmy zapewniające identyfikację w sieci i bezbłędne przesyłanie danych. Co ważne, mamy również możliwość łatwego tworzenia interfejsu graficznego dla użytkownika. Prezentujemy drugą część opisu programy TCPMaker ułatwiającego dołączenie mikrokontrolera PIC do sieci Ethernet.

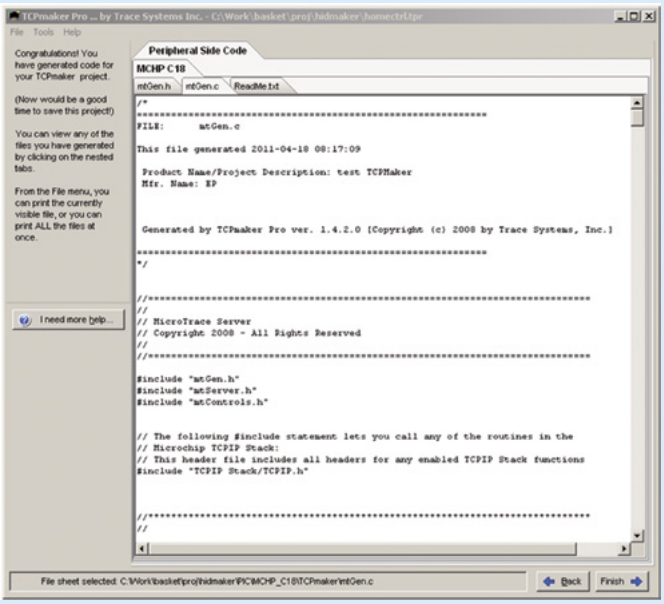
### Program sterownika

Definiowanie listy zmiennych i umieszczenie wszystkich elementów z palety na poszczególnych stronach kończy etap projektowania strony. Wynik tych działań można przetestować w zakładce Save, Test and Exit. Po zapisaniu projektu można przetestować utworzona stronę po kliknięciu na Test. Strona jest otwierana lokalnie w przeglądarce i można sprawdzić jej wygląd oraz działanie nawigacji, wszystkie funkcje oprócz komunikacji ze sterownikiem. Przyznam się, że w trakcie pracy nad projektem wiele razy korzystałem z tej możliwości. W każdej chwili można wrócić do zakładki Lay Out Page(s) i wykonać poprawki wyglądu (rozmieszczenie elementów, ustawienia kolorów itp.). Ostatnim krokiem jest generowanie kodu źródłowego dla sterownika. Tutaj również projektanci TCPMaker'a wykonali dobrą robotę. Po naciśnięciu przycisku w zakładce step 3 przycisku Finally go back and generate code pokazuje się ekran z wygenerowanymi plikami mtGen.c, mtGen.h i ReadMe.txt (**rysunek 21**). Jeżeli w oknie wyboru kompilatora wybralibyśmy więcej niż jeden kompilator, to program wygenerowałby takie pliki również dla pozostałych. W oknie z rysunku 21 pokazano tylko najważniejsze pliki z punktu widzenia modyfikacji komunikacji sterownika z komputerem, ale program generuje kompletny projekt dla MPLAB IDE ze wszystkimi plikami źródłowymi, włączając w to stos TCP/IP.

Projekt ma strukturę podobną do programów demonstracyjnych dystrybuowanych przez Microchip'a. Przed rozpoczęciem pracy nad właściwym projektem musimy określić, którego mikrokontrolera chcemy użyć. W tym projekcie wiadomo, że jest to mikrokontroler PIC18F67J60 z własnym PHY i MAC, a do komunikacji musi być skompilowany z driverem ETH97J60.c Pliki źródłowe zawierają szereg komend preprocesora umożliwiających warunkową kompilację dla różnych platform sprzętowych: Explorer16, PICDEMnet2, PIC32 Sterter KIT, Interent Radio Board itp.

Wszystkich ustawień dotyczących sprzętu dokonuje się w pliku HardwareProfile.h. Tak się dobrze składa, że predefiniowana konfiguracja modułu radia internetowego „pasuje" do naszego projektu. Na samym początku tego pliku wystarczy usunąć komentarz z definicji INTERENT RADIO i wszystkie definicje dotyczące konfiguracji sygnału zegarowego i wewnętrznego modułu ethernetowego są gotowe (**listing 1**).

Producent TCPMakera zachęca, aby pierwsze próby przeprowadzać z firmowymi modułami Microchip'a. Dla bardziej zaawansowanych jest przygotowana opcja




**Rysunek 21. Okno generowania plików źródłowych**

**Listing 1. Wybór sprzętu**
```
/* Choose which hardware profile to compile for here. See
   the hardware profiles below for meaning of various options. */
//#define PICDEMNET2
#define INTERNET_RADIO
//#define PIC18_EXPLORER
//#define HPC_EXPLORER
//#define PIC24FJ64GA004_PIM
/* Explorer 16, but with the PIC24FJ64GA004 PIM module, which has significantly different pin mappings */
//#define EXPLORER_16
/* PIC24FJ128GA010, PIC24HJ256GP610, dsPIC33FJ256GP710, PIC32MX360F512L, PIC32MX460F512L, PIC32MX795F512L,
etc. PIMs */
//#define DSPICDEM11
/* PIC32MX360F512L General Purpose Starter Kit (for purposes of TCP/IP, defining this macro is the same as
defining PIC32_USB_DM320003_1 or PIC32_USB_SK_DM320003_2) */
//#define PIC32_GP_SK_DM320001
/* PIC32MX460F512L USB Starter Board (for purposes of TCP/IP, defining this macro is the same as defining
PIC32_GP_SK_DM320001 or PIC32_USB_SK_DM320003_2) */
//#define PIC32_USB_DM320003_1
/* PIC32MX795F512L USB Starter Kit II (for purposes of TCP/IP, defining this macro is the same as defining
PIC32_GP_SK_DM320001 or PIC32_USB_DM320003_1) */
//#define PIC32_USB_SK_DM320003_2
/* PIC32MX795F512L Ethernet Starter Kit board with embedded Ethernet controller */
//#define PIC32_ENET_SK_DM320004
// PIC24FJ256DA210 Development Board (Graphics)
//#define PIC24FJ256DA210_DEV_BOARD
//#define YOUR_BOARD
```

YOUR BOARD. Po jej wyborze trzeba odszukać w HardwareProfile.h miejsce przeznaczone do własnych definicji i zdefiniować konfigurację sprzętową według potrzeb. Po zdefiniowaniu sprzętu głównie będziemy się zajmować plikiem mtGen.c. Na początku są w nim umieszczone deklaracje zmiennych i tablic zdeklarowanych na liście zmiennych (**listing 2**).

Deklaracja części zmiennych jest zdefiniowana jako komentarz. Jeżeli są potrzebne to trzeba je ponownie zdefiniować według instrukcji zawartej w pliku obok deklaracji. Do zmiennych, które mogą być przesyłane do komputera automatycznie są deklarowane flagi TxFlag używane do inicjowania przesyłania zwartości zmiennych do komputera.

**Listing 2. Fragment pliku mtGen.c z deklaracją zmiennych**
```
// Variables that travel either into or out of the device are declared
// Data transfer variable as created in Visual Page Designer
int Impower;
// Transmit flag for this variable: set True to send
unsigned char ImpowerTxFlag;
// Data transfer variable as created in Visual Page Designer
int Imled;
// Transmit flag for this variable: set True to send
unsigned char ImledTxFlag;
// Data transfer variable as created in Visual Page Designer
int Ntemp;
// Transmit flag for this variable: set True to send
unsigned char NtempTxFlag;
// Data transfer variable as created in Visual Page Designer
int Ntempo;
// Transmit flag for this variable: set True to send
unsigned char NtempoTxFlag;
// Data transfer variable as created in Visual Page Designer
int Ilight;
// Transmit flag for this variable: set True to send
unsigned char IlightTxFlag;
// Data transfer variable as created in Visual Page Designer
int Ilightled;
// Transmit flag for this variable: set True to send
unsigned char IlightledTxFlag;
// Data transfer variable as created in Visual Page Designer
int Isec;
// Transmit flag for this variable: set True to send
unsigned char IsecTxFlag;
// Data transfer variable as created in Visual Page Designer
int Isecled;
// Transmit flag for this variable: set True to send
unsigned char IsecledTxFlag;
// Data transfer variable as created in Visual Page Designer
int Nbat;
// Transmit flag for this variable: set True to send
unsigned char NbatTxFlag;
// Data transfer variable as created in Visual Page Designer
int Isetlight;
// Transmit flag for this variable: set True to send
//unsigned char IsetlightTxFlag;
// Data transfer variable as created in Visual Page Designer
int Nlight;
/* Un-comment the following declaration, and other lines marked Nlight_TX elsewhere in mtGen.h and mtGen.c,
to be able to transmit this variable to the PC browser. Also, increase MtTxListLength by 1 in mtGen.h if you
enable Nlight_TX lines */
// Nlight_TX:
// Transmit flag for this variable: set True to send
//unsigned char NlightTxFlag;
// Data transfer variable as created in Visual Page Designer
int Ialert;
// Transmit flag for this variable: set True to send
unsigned char IalertTxFlag;
// Data transfer variable as created in Visual Page Designer
char Mt2[Mt2_Length];
// Transmit flag for this variable: set True to send
unsigned char Mt2TxFlag;
// Data transfer variable as created in Visual Page Designer
char Mt1[Mt1_Length];
// Transmit flag for this variable: set True to send
unsigned char Mt1TxFlag;
```

**Listing 3. Funkcja obsługująca naciśnięcie przycisku Power ON**

```
// Event handler for variable:  Impower - main power on/off
void eventImpower(void)
{
/* Variable has just arrived, so add your code to DO something with it here:  */
  if(Impower==1)
  {
    Imled=1;
    POWER=1;  //załącz przekaźnik
    LCDPutStr(" ON  " , 25, 88, LARGE, WHITE,RED);
    ImledTxFlag=1;
  }
    else
  {
    Imled=0;
    POWER=0;  //wyłącz przekaźnik
    LCDPutStr(" OFF " , 25, 88, LARGE, BLACK,GREEN);
    ImledTxFlag=1;
  }
}
```

**Listing 4. Fragment funkcji mtEndOfMessage**

```
Ntemp=ReadDS();  //odczytanie wewnętrznej  temperatury z DS18B20
DispTemp(Ntemp,85,64,BLACK, YELLOW);  //wyświetlenie temperatury
NtempTxFlag = 1;  //wysłanie do komputera
if(Ntemp<=TL&&thi==1)//porównanie z zadanymi progami i symulacja włączenia grzejnika
{
  tlo=1; thi=0;
  strcpypgm2ram(Mt2,"HEATER ON ");//przesłanie napisu do Text Indicator
  Mt2TxFlag=1;
}
if(Ntemp>=TH&&tlo==1)
{
  tlo=0; thi=1;
  strcpypgm2ram(Mt2,"HEATER OFF");//przesłanie napisu do Text Indicator
  Mt2TxFlag=1;
}
Ntempo=ReadDS1();//odczytanie zewnętrznej temperatury z DS18B20
DispTemp(Ntempo,65,64,WHITE,BLUE);//wyświetlenie temperatury
NtempoTxFlag = 1;//wysłanie do komputera
```

Stos TCP/IP działa pod kontrolą czegoś w rodzaju systemu RTOS. Komunikacja pomiędzy sterownikiem a komputerem również działa na tej zasadzie. TCPMaker dla zmiennych, które mogą być przesyłane automatycznie tworzy funkcje obsługujące zdarzenia. Po wygenerowaniu plików funkcje obsługujące zdarzenia są puste i trzeba samemu dopisać obsługę.

Na **listingu 3** pokazano przykład napisanej przeze mnie funkcji obsługi załączania zasilania wykorzystującej dane przesłane z elementu Pushbutton. Po przyciśnięciu przycisku na stronie komputer zapisuje do zmiennej Impower wartość 0x01. Sterownik po jej odebraniu ze stosu TCP/IP przekazuje sterownie do funkcji eventImpower(). W tej funkcji do zmiennej Imled związanej z zapaleniem diody LED umieszczonej pod przyciskiem Power ON wpisywane jest 0x01, a potem jest załączany przekaźnik (ustawienie linii POWER), na wyświetlaczu LCD w linijce MAIN POWER jest wyświetlany napis ON. Na końcu działania tego fragmentu ustawiana jest flaga ImledTxFlag. Powoduje to, ze zawartość Imled jest przesyłana do komputera i dioda LED się zapala na ekranie. Każde kolejne przyciśnięcie przycisku ON będzie powodowało sekwencyjne wpisywanie do Impower 0x01 i 0x00 i w wyniku załączanie i wyłączanie przekaźnika i sygnalizację stanu diodą LED.

Mechanizm nieskomplikowany i co ważne skuteczny i niezawodny.

Dane przesyłane są do komputera po każdym wpisaniu do flagi wartości 0x01 w funkcji obsługi zdarzenia (event) wywoływanej po odebraniu danej z komputera. Ale gdy chcemy na przykład wysyłać zmierzona temperaturę, taki sposób może być użyty tylko do pomiaru na żądanie (na przykład po naciśnięciu Pushbutton). Spontaniczne wysyłanie danej przez sterownik można wykonać w funkcji mtEndOfMessage() zamieszczonej na **listingu 4**. Nie zaleca się przesyłania zbyt wielu danych jednocześnie. W tym przypadku przesłanie dwóch temperatur i ewentualnie komunikatu tekstowego nie stanowi problemu, bo wykonuje się stosunkowo szybko.

Bufor Mt2 nie jest definiowany w liście zmiennych i TCPMaker sam je sobie zdefiniował na podstawie informacji o umieszczeniu na stronie elementu Text Indicator. Fragment z **listingu 5** jest wywoływany co około 0.5 sekundy.

Zadbano również o możliwość inicjalizacji zmiennych po włączeniu zasilania (zerowaniu) sterownika. Po każdym zerowaniu wywoływana jest funkcja mtUserInit nadająca początkowe wartości wszystkim zmiennym i funkcja mtConnect, w której można do komputera przesłać wartości początkowe potrzebnych zmiennych. Inicjalizacja jest pokazana na listingu 5.

Obsługę utraty połączenia z komputerem zapewnia funkcja mtDisconnect. Można tam na przykład ustawić znacznik braku łączności. Testowanie tego znacznika umożliwia pełniejszą obsługę połączenia z komputerem i ewentualne przejście na pełny tryb lokalny. Dla celów testowych wyświetlam w tej funkcji napis „PC DISCONNECT" (**listing 6**).

Wystarczy zamknąć stronę sterującą i pojawia się ten komunikat. Po ponownym nawiązaniu łączności funkcja mtConnect z list. 5 wyświetla napis PC CONNECT.

**Listing 5. Funkcja mtConnect.**

```
void mtConnect(void)
{
  Nbat=125;  //napięcie baterii
  NbatTxFlag=1;
  Ntemp=ReadDS();  //temperatura wewnętrzna
  NtempTxFlag = 1;
  Ntempo=ReadDS1();  //temperatura zewnętrzna
  NtempoTxFlag = 1;
  LCDPutStr("PC CONNECT   ",0,8, LARGE,BLACK ,WHITE);
}  // mtConnect()
```

**Listing 6. Funkcja mtDisconnect**

```
void mtDisconnect(void)
{
  LCDPutStr("PC DISCONNECT " ,0, 8, LARGE,RED ,YELLOW);
}
```

```
Listing 7. Fragmentu pliku podglądu transmisji
//iv1=1  włacz Main Power - kierunek PC->sterownik
<-PC at 8169 msec: <LF>iv1  1<LF>
<-PC at 8254 msec: <LF>
<-PC at 8336 msec: <LF>
<-PC at 8419 msec: <LF>
<-PC at 8504 msec: <LF>
<-PC at 8586 msec: <LF>
<-PC at 8669 msec: <LF>
<-PC at 8754 msec: <LF>
<-PC at 8836 msec: <LF>
<-PC at 8919 msec: <LF>
<-PC at 9004 msec: <LF>
<-PC at 9086 msec: <LF>
<-PC at 9169 msec: <LF>
<-PC at 9254 msec: <LF>
<-PC at 9336 msec: <LF>
<-PC at 9419 msec: <LF>
<-PC at 9504 msec: <LF>
<-PC at 9586 msec: <LF>
//przesłanie temperatur nv0 i nv1 - kierunek sterownik
->PC
->PC at 9656 msec: <mt><nv0 i="44"/><nv1 i="49"/></mt>
<-PC at 9669 msec: <LF>
//przesłanie stanu diody LED Main Power  - kierunek
sterownik ->PC
->PC at 9686 msec: <mt><iv2 i="1"/></mt>
```

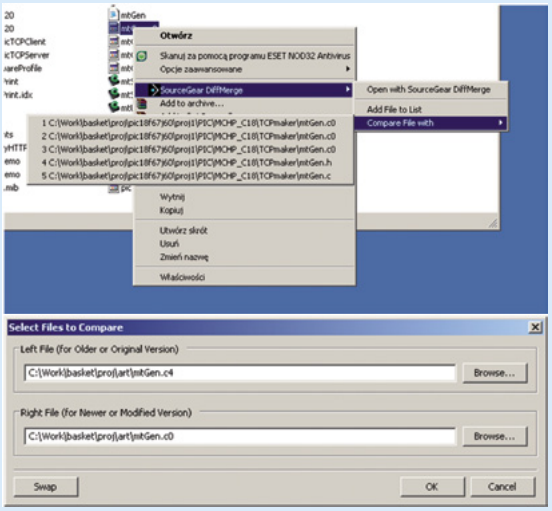
Plik mtGen.c zawiera jeszcze inne przydatne funkcje na przykład mtFastTick, czy mtStartOfMessage. Kazda z tych funkcji jest wyczerpująco opisana komentarzami. Warto tez zajrzeć do pliku nagłówkowego mtGen.h i zapoznać się ze zmiennymi mtMinConnectSeconds i mtMinIdleSeconds. Sterownik nie może się jednocześnie łączyć z więcej niż jedną stroną. Każdemu użytkownikowi po połączeniu przypisywany jest czas 30 minut aktywności i 5 minut stanu bezczynności. Po przekroczeniu tych czasów próba połączenia się nowego użytkownika powoduje odłączenie poprzedniego. Program sterownika modyfikuje pliki HardwareProfile.h, mtGen.c oraz MainDemo.c. Dodatkowo są dołączone pliki źródłowe obsługi wyświetlacza, magistrali 1-wire i termometry DS18B20.

## Modyfikowanie projektu

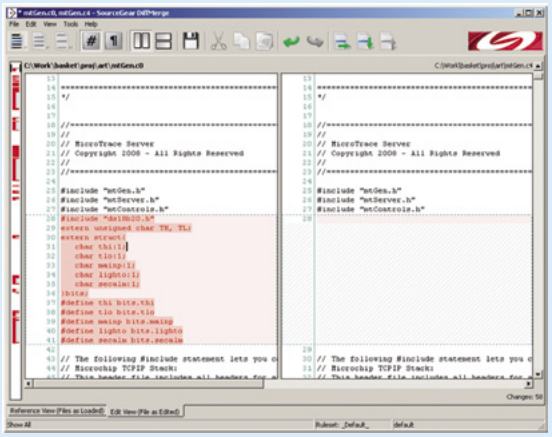
W trakcie pracy nad projektem często zachodzi potrzeba zamian i poprawek. Trzeba wtedy wrócić do zakładek z listą zmiennych (step1) i projektu strony (step2). TCPMaker nie jest w stanie określić jakie zmiany wykonał programista w funkcjach obsługi zdarzeń mtGen.c w momencie generowania nowych plików (step3). Może jednak poprzednie pliki zapisać z innymi nazwami, tak by nie niszczyć włożonej wcześniej pracy. Poprzednie pliki są zapisywane na żądanie pod kolejnymi nazwami mtGen.c0, mtGen.c1, mtGen.c2 itd. Za każdym razem po poprawkach i generowaniu nowych plikach poleceniem Finally go back and generate code trzeba na nowo uzupełnić zawartość funkcji obsługujących zdarzenia, ale tez innych zmian w pliku mtGen.c. Przy mniej rozbudowanym projekcie można to robić ręcznie. Ja początkowo poradziłem sobie tworząc dodatkowy plik rob.c, w którym wklejałem każdą zmodyfikowana funkcję z pliku mtGen.c i po wygenerowaniu nowego ponownie wklejałem zwartość funkcji do nowego pliku. Jednak przy stopniowym rozbudowywaniu projektu staję się to coraz bardziej uciążliwe.

Producent zaleca do wykrywania zmian i nanoszenia poprawek wykorzystanie programu Beyond Compare3. Dokładna instrukcja postępowania znajduje się w pliku pomocy. Ja wykorzystałem inny program SourceGear DiffMerge, który dla niekomercyjnego wykorzystania jest bezpłatny.

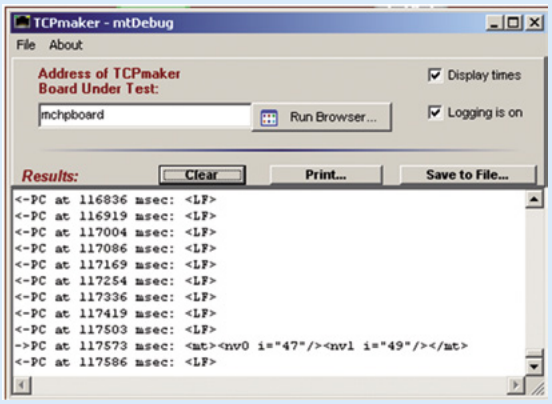
Pliki do porównania (AddFile to List) zaznacza się po kliknięciu prawym klawiszem na plik i wybraniu SourceGear DiffMerge. Następny wybrany plik można porównywa z dowolnym wcześniej wybranym z listy (**rysunek 22**). Plik „wzorcowy” (w lewym oknie), to plik wygenerowany




**Rysunek 22. Wybór plików do porównania**



**Rysunek 23. Okno programu z otwartymi plikami**




**Rysunek 24. Okno programu debugowania**

wcześniej i zawierający poprawki, czyli na przykład mtGen.c0. Plik w prawym oknie (do modyfikacji) to plik ostatnio wygenerowany przez TCPMakera z nowymi definicjami zmiennych i pustymi funkcjami obsługi zdarzeń.

Po wybraniu plików pojawia się okno porównania (**rysunek 23**). Różnice pomiędzy oboma plikami podświetlane są na czerwono. Kliknięcie na ikonę Apply change from left (zielona strzałka w prawo) spowoduje, że różnice pliku z lewej strony zostaną przepisane do prawego pliku i zaznaczone pogrubioną czcionką z podkreśleniem. Następną różnicę identyfikuje się kliknięciem na ikonę Jump to next change (zielona strzałka w dół). Tak postępując można szybko i bezbłędnie przepisać wszystkie poprawki wykonane przed ponownym wygenerowaniem plików przez TCPMakera.

TCP Maker ma wbudowaną jeszcze jedną bardzo przydatną funkcję podglądania transmisji danych w obu kierunkach w czasie rzeczywistym w trakcie działania serwera. Aplikacja debugowania instaluje się w trakcie instalowania pakietu TCPMaker i jest uruchamiana po kliknięciu na ikonę trace mtDebug. Okno działającego programu pokazano na **rysunku 24**. Rozpoczęcie podglądu transmisji następuje po kliknięciu na Run Browser z wpisanym w okienku adresem mchpboard. Po automatycznym otwarciu się strony o tym adresie w okienku programu wypisują się informacje o przesyłanych danych. W praktyce bardzo użyteczna jest możliwość zapisania wypisów do pliku tekstowego, bo informacje przesuwają się w okienku dość szybko. Fragment zarejestrowanego pliku z dodanymi moimi komentarzami pokazano na **listingu 7**. Podgląd można przerwać w dowolnym momencie odznaczając pole Logging is on.

To dodatkowe narzędzie znakomicie ułatwia znalezienie błędów w projekcie. Ja na początku pracy z TCPMaker'em wykorzystywałem go by zobaczyć jak w praktyce działają poszczególne elementy dodawane do projektu. Ta wiedza znacznie ułatwiła mi zrozumienie idei działania całości.

## Podsumowanie

Ograniczona objętość artykułu nie pozwala na opisanie wszystkich możliwości programu. W przedstawionym tutaj przykładzie starałem się pokazać najważniejsze cechy ukazujące potencjał programu. Według mojej opinii narzędzie TCPMaker można nazwać przełomowym, ponieważ ogromnie upraszcza i przyspiesza proces projektowania aplikacji sterowania sieciowego, dając jednocześnie możliwość zaprojektowania wydajnego i efektownego interfejsu w postaci strony otwieranej w przeglądarce internetowej. Trzeba oczywiście poświęcić trochę czasu, żeby poznać zasadę możliwości i działania, ale tak jest w przypadku każdego bardziej rozbudowanego narzędzia. Jednak po opanowaniu przynajmniej najważniejszych czynności projektowanie przebiega nadzwyczaj sprawnie. Również bardzo istotne jest to, że moja aplikacja działała bardzo stabilnie. Nie zauważyłem zawieszania się i konieczności zerowania mikrokontrolera, lub zamykania i otwierania strony. Być może jest to wynik w miarę ścisłego stosowania się do wskazówek zawartych w pliku pomocy i tych zapisanych w komentarzach plikach mtGen.c i mtGen.h.

Działanie TCPmaker'a jest ograniczone do mikrokontrolerów produkowanych przez Microchip'a. Projektanci wykorzystujący inne mikrokontrolery mogą się poczuć zawiedzeni, ale oferta Microchipa jest na tyle szeroka od pokazanego tu 8 bitowego PIC18F67J60 poprzez 16 bitowe PIC24 i 32 bitowe PIC32, że trudno nazwać to istotnym ograniczeniem.

TCPMaker jest (jak już wspomniałem) programem płatnym. W MicrochipDirect kosztuje około 380$. Jest to wydatek akceptowalny dla bardziej zamożnych amatorów, a na pewno dla firm zajmujących się projektowaniem interfejsów sieciowych.

**Tomasz Jabłoński, EP**

REKLAMA