Co nowego w Tibbo?

Ethernet dla opornych, ale wymagających

Firma Tibbo Technology Inc. konsekwentnie rozszerza swoją ofertę modułów ethernetowych. Oprócz sprzętu, Tibbo oferuje kompletne środowisko uruchomieniowe, wykorzystujące BASIC jako język programowania. Uwagę przyciąga fakt, iż wszystkie nowe moduły są wyłącznie urządzeniami programowalnymi. Nie posiadają domyślnego firmware'u dostarczającego funkcjonalności mostu RS232 <–> Ethernet. Zacznijmy od nowości software'owych.

Taiko (przypomnijmy) jest zestawem oprogramowania składającym się z trzech elementów. Pierwszym z nich jest (kompilowany) język programowania *Tibbo BASIC*. Drugim *Tibbo Integrated Development Environment* (w skrócie TIDE) – kompletne środowisko uruchomieniowe zawierające edytor kodu, kompilator, linker oraz debugger. Ostatnim elementem jest TIOS (*Tibbo Operating System*) – wbudowany system operacyjny. Dostarczany jest on w postaci pojedynczego pliku binarnego, specyficznego dla platformy. Przed rozpoczęciem pracy z TIDE musi on być umieszczony w wewnętrznej pamięci Flash wybranego modułu.

Zmiany w Tibbo BASIC

Język Tibbo Basic pozwala na stosowanie w programach zmiennych 32–bitowych (w pierwszej wersji maksymalny rozmiar zmiennej wynosił 16 bitów). Programista ma teraz do dyspozycji zmienne typu dword, long, real oraz float. W nomenklaturze Taiko `dword` oznacza 32–bitową zmienną całkowitą bez znaku, (jest odpowiednikiem typu `unsigned int` w C dla procesorów 32–bitowych). `Long` (32 bit) jest typem całkowitym ze znakiem (typ `int` w C dla procesorów 32–bitowych). `Real` – typ zmiennoprzecinkowy ze znakiem. Działania z jego użyciem mogą prowadzić do różnych wyników na różnych platformach sprzętowych. Kalkulacje z użyciem typu `real` mogą też prowadzić do wyników typu nieskończoność lub NaN (*Not a Number*). Należy być ostrożnym w ich stosowaniu. `Float` – typ identyczny do `real`. Wszystkie zmienne zadeklarowane jako `float` są na etapie kompilacji interpretowane jako `real`. Typ ten wprowadzono dla zgodności Tibbo BASIC z innymi implementacjami języka.

W Taiko R2 poprawiono też obsługę tablic. Tablice mogą teraz przechowywać elementy dowolnego typu (poprzednio możliwe było deklarowanie tylko tablic typu `char`). Zmiany dotknęły też sposobu indeksowania tablic. Obecnie nie jest możliwe zaindeksowanie elementu, który nie istnieje. Oznacza to, iż tablica ma percepcję swojego rozmiaru. Jeśli wystąpi sytuacja, w której dojdzie do podania błędnego indeksu w tablicy, wówczas w trybie debugerra zatrzyma on program oraz wygeneruje wyjątek OOR (*out of range*). Jeśli użytkownik mimo to zechce wznowić działanie programu, błędnie zaadresowana tablica zwróci element znajdujący się pod maksymalnym obecnym indeksem. Jeśli taka sytuacja zdarzy się podczas działania programu w normalnym trybie (bez debuggera), wówczas błędnie indeksowana tablica zawsze będzie zwracać swój maksymalny element. Wprowadzenie tego mechanizmu potwierdza deklarowaną wcześniej przez Tibbo chęć upodobnienia BASIC-a do języków wyższych poziomów. Odciąża to nieco programistę i pozwala na szybszą detekcję błędu indeksacji (który często zdarza się podczas programowania w języku C). W najnowszej wersji Taiko możliwa jest też deklaracja tablic wielowymiarowych. Maksymalna liczba wymiarów wynosi 8.

Począwszy od wersji R2 Taiko wspiera definiowalne przez użytkownika struktury danych. Sposób deklaracji jest zbliżony do znanego z języka C. Podobnie jak i sposób wyłuskiwania kolejnych członków struktury (odbywa się to przy pomocy schematu: *struktura.element*). Struktury w Tibbo BASIC mogą zawierać elementy o typach podstawowych, tablice, jak również inne struktury. Maksymalny poziom zagnieżdżenia struktur w innych strukturach wynosi 8. Zmienna typu predefiniowanej struktury nie ma rozmiaru w pamięci równego sumie wielkości poszczególnych jej członków. Faktyczny rozmiar jest większy ze względu na to, iż (tak jak w przypadku tablic) struktura przechowuje informacje na temat rozmiarów swoich elementów, jak również ich typów (podobnie jak w językach wyższego poziomu).

Nowa wersja Tibbo BASIC wprowadza też jasną i jednoznaczną, automatyczną konwersję typów. Konwersja taka następuje zawsze w przypadku przypisania zmiennej jednego typu wartości zmiennej drugiego typu. Wszystkie możliwe kombinacje przypisań typów i rezultatu autokonwersji przedstawiono w **tab. 1**.

Jeśli na przecięciu dwóch różnych typów znajduje się symbol „OK.", wówczas konwersja typu źródłowego do typu docelowego odbywa się bez straty informacji. Dodatkowo, w przypadku pól „*OK*

Tab. 1. Tabela autokonwersji typów w Tibbo BASIC R2 (za dokumentacją Taiko)

Typ zmiennej źródłowej	Typ zmiennej docelowej								
		byte	word	dword	char	short	long	real	string
	byte	—	OK	OK	Reinterpret	OK	OK	OK	OK (str)
	word	Truncate	—	OK	Reinterpret Truncate	Reinterpret	OK	OK	OK (str)
	dword	Truncate	Truncate	—	Truncate	Truncate	Reinterpret	OK	OK (lstr)
	char	Reinterpret	Reinterpret	Reinterpret	—	OK	OK.	OK	OK (stri)
	short	Reinterpret Truncate	Reinterpret	Reinterpret	Truncate	—	OK.	OK	OK (stri)
	long	Reinterpret Truncate	Reinterpret	Truncate	Truncate	Truncate	—	OK	OK (lstri)
	real	Fraction	Fraction	Fraction	Fraction	Fraction	Fraction	—	OK (fstr)
	string	OK (val)	OK (val)	OK (lval)	OK (val)	OK (val)	OK (lval)	OK (strof)	—

(val)" kompilator przed podstawieniem wartości do zmiennej docelowej automatycznie wywoła funkcję systemową (podaną w nawiasie), która przekonwertuje np. znak zmiennej typu `string` na jego kod ASCII (w przypadku `val`). Funkcja systemowa zostanie wywołana automatycznie tylko w przypadku, kiedy użytkownik nie wywoła jej w sposób jawny.

Jeśli wynik konwersji jest opatrzony wyrażeniem *Truncate* oznacza to, że starsza część (bajt w przypadku zmiennej typu `word` lub słowo w przypadku zmiennej typu `dword` zmiennej źródłowej zostanie odrzucona. Zmienna docelowa odzwierciedli wówczas jedynie młodszą część zmiennej źródłowej. Jeśli wynik konwersji jest oznaczony wyrażeniem *Reinterpret,* oznacza to, że zmienna zostanie przeliczona na wartość nowego typu. W tym przypadku nie występuje problem niewystarczającego rozmiaru zmiennej docelowej, ale różnego sposobu kodowania danych. Zatem z poziomu kompilatora zmienna `byte` o wartości 0xFF jest interpretowana jako 255. Taka sama wartość zapisana binarnie do zmiennej typu `char` (a więc zmiennej 7–bitowej ze znakiem) da na poziomie kompilacji wartość –1. Wynika to ze sposobu kodowania obu zmiennych. Zmienna `byte` jest zapisywana przy użyciu naturalnego kodu binarnego (NKB) na 8 bitach i może przechowywać wartości z zakresu 0 do 255 (dziesiętnie). Zmienna typu `char` wykorzystuje kod U2 i może przechowywać wartości z zakresu –128 do 127. Zatem przy autokonwersji *Reinterpret*, zapis zmiennej w pamięci urządzenia nie ulega zmianie, ale na poziomie kompilatora obie zmienne przechowują inną wartość. W tab. 1 występują też konwersje oznaczone jako *Truncate* i *Reinterpret,* co oczywiście oznacza, iż konwersja po odrzuceniu starszej części zmiennej spowoduje reinterpretację wyniku. W przypadku konwersji oznaczonej jako ~~*Fraction*~~ (w dokumentacji Taiko przekreślenie słowa *Fraction* oznacza, że część ułamkowa zmiennej typu `float` zostaje pominięta) zawartość zmiennej typu `real` po przypisaniu jej do zmiennej innego typu zostanie pozbawiona części ułamkowej, zatem zawsze nastąpi tu zaokrąglenie liczby w dół.

Rys. 1. Moduł EM1000

Zmiany w TIDE

W Taiko R2 zmiany dotknęły też środowiska uruchomieniowego. Następstwem wprowadzenia struktur była zmiana w narzędziu *Watch,* które teraz pozwala na dowolną weryfikację danych w instancjach zdefiniowanych struktur. Niejako przy okazji narzędzie to wzbogacono o możliwość podglądu zawartości tablic.

Do IDE dodano też nowe narzędzie nazwane *Image Editor*, które pozwala na prostą edycję plików graficznych dodawanych do projektu jako elementy typu *resource*.

Zmiany w działaniu platform

W Takio R2 poprawiono działanie niektórych obiektów platform. Kosmetyczne zmiany dotknęły obiekty i metody *pat.play* oraz *beep.play*. Polegają one na dodaniu wyższych częstotliwości odgrywania wzorców przy pomocy diod LED oraz sygnału dźwiękowego. Bardzo przydatne jest natomiast wprowadzenie własności obiektu sys: *sys.onsystimerperiod*. Własność ta konfiguruje okres generowania zdarzenia *on_sys_timer*, który dotychczas był stały i wynosił 0,5 s (co było nieco uciążliwe). Obecnie możliwe jest jego konfigurowanie z ziarnem 10 ms w zakresie od 10 ms do 2550 ms. Niestety własność ta nie jest dostępna na wszystkich platformach. Taiko R2 wnosi też coś do poprawy bezpieczeństwa sieciowego modułów. Wprowadzono bowiem własność obiektu sock: *sock.inconenabledmaster*. Powoduje ona globalne (dla wszystkich gniazd niezależnie od ich konfiguracji) zablokowanie akceptacji połączeń przychodzących do modułu. Co istotne, nie zmienia ona konfiguracji poszczególnych gniazd, zatem po ponownym odblokowaniu połączeń przychodzących nie ma konieczności ich ponownej inicjalizacji. Taiko R2 umożliwia też większą kontrolę nad wysyłanymi i odbieranymi pakietami TCP. Gdy własność *sock.splittcppackets* jest ustawiona na jeden, otrzymujemy możliwość percepcji rozmiaru każdego przychodzącego pakietu TCP. Z kolei po stronie nadawczej możemy zdefiniować, jak długie mają być pakiety wysyłane z naszego urządzenia. Własność ta musi być jednak stosowana z rozwagą. Zdefiniowanie zbyt długich pakietów spowoduje, iż moduł będzie oczekiwał na ich gromadzenie się w buforze nadawczym, co może niekorzystnie wpłynąć na całościową prędkość transmisji. Drugim niebezpieczeństwem jest to, iż strona odbiorcza może nie posiadać dostatecznie dużych buforów odbiorczych, które pozwalałyby na obsługę długich pakietów. Sytuacja taka będzie powodowała to, iż żaden z pakietów nie dotrze do miejsca przeznaczenia, gdyż urządzenie odbiorcze będzie miało stale przepełniony bufor.

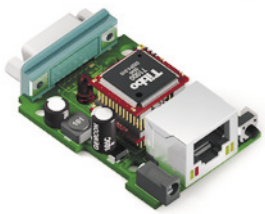
Poprawiono też działanie mechanizmu dynamicznego generowania stron WWW. Problem objawiał się w przypadku jednoczesnego żądania strony WWW przez dwie różne przeglądarki. Jeśli w kodzie strony (html) umieszczono procedurę BASIC odpowiedzialną za dynamiczne generowanie jej zawartości, wówczas tylko pierwszy z klientów (pierwsza przeglądarka) otrzymał poprawną stronę. W przypadku drugiego klienta, zwracana była strona, jednak bez części zawartości generowanej dynamicznie. Zachowanie to było błędne i wynikało z natury Tibbo BASICA, który nie pozwala na rekursywne wywoływanie procedur. Dzieje się tak dlatego, iż system TiOS nie używa dynamicznej alokacji pamięci (co powodowałoby jego wolniejsze działanie). Problem dynamicznego generowania kodu html dla dwóch klientów jednocześnie został zatem rozwiązany w inny prosty sposób. Przy jednoczesnym żądaniu strony, drugi klient będzie musiał poczekać aż procedura BASIC zostanie dokończona po wywołaniu przez pierwszego klienta. Po tym czasie zostanie ona wywołana ponownie na potrzeby drugiego klienta.

A co nowego w sprzęcie?

Tibbo nie próżnuje też w dziedzinie platform sprzętowych, (bez których nie powstałby Tibbo BASIC). Wszystkie nowe platformy są wyłącznie urządzeniami programowalnymi, nie posiadającymi domyślnego firmware'u. Pierwszą z platform trzeciej generacji (bo tak Tibbo określa nowe moduły), jest EM1000 (**rys. 1**). Urządzenie ma formę modułu PCB z wyprowadzonymi pinami w rastrze 2,54 mm, co z pewnością ułatwi wykonanie projektu płytki drukowanej. Moduł jest wyposażony w dedykowa-

ny mikroprocesor taktowany zegarem 88 MHz i legitymujące się mocą obliczeniową 50 MIPS'ów. Wyposażony jest w jeden port Ethernetowy 100BaseT oraz 4 szybkie porty szeregowe. Każdy z nich może być skonfigurowany do pracy w charakterze UART–u, jak również interfejsu Wiegand oraz interfejsu zbliżonego do SPI. Maksymalna prędkość transmisji wynosi 1,8 Mbit/s. Nie zabrakło też miejsca dla zegara czasu rzeczywistego z opcjonalnym kondensatorem podtrzymującym jego pracę (wersja modułu „–S"). Moduł posiada 49 linii I/O ogólnego przeznaczenia oraz 4 linie dedykowane do obsługi diod statusowych (np. linku ethernetowego). Moduł posiada pamięć typu Flash 512 k lub 1024 k oraz pamięć RAM (20 k). Ponadto, do dyspozycji użytkownika pozostaje też pamięć typu EEPROM o rozmiarze 2048 bajtów.

Rys. 2. Moduł EM1202



Rys. 3. Płytka ewaluacyjna z modułem EM1202

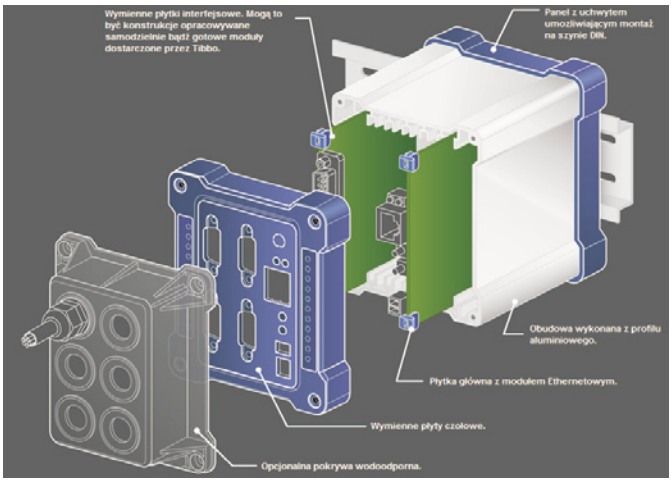
Rys. 4. Złącze RJ45 wraz z wbudowanym transformatorem separującym

Rys. 5. Kontroler DS1000

Zbliżonym pod względem funkcjonalnym do EM1000 jest moduł EM1202 (**rys. 2**). Jedyna różnica w stosunku do EM1000 to ograniczona liczba linii I/O. EM1202 jest za to dużo mniejszy. Zbudowany jest w formie stosu. Specjalnie z myślą o nim Tibbo opracowało kompaktowe gniazdo RJ45 (**rys. 3**) wraz z wbudowanym transformatorem separującym. Wraz z modułem EM1202 zajmują na PCB przestrzeń podobną do znanego czytelnikom modułu EM202. Oczywiście DS1202 może być stosowane z innymi modułami (np. EM1000). Dla EM1202 została też opracowana mała platforma uruchomieniowa: moduł EM1202–EV (**rys. 4**). Na płytce drukowanej modułu znajdziemy EM1202 oraz złącze RJ1202. Pozostałe komponenty różnią się w zależności od wariantu urządzenia (obecnie dostępne są trzy: –RS, –TM, –TS).

Moduł EM1202 oprócz pracy jako platforma rozwojowa, może służyć też do szybkiego upgradu istniejących urządzeń. Poszczególne jego warianty mogą służyć do innych modyfikacji. I tak, moduł –RS (RS232) może pracować jako samodzielne urządzenie (w najprostszej postaci jako konwerter Ethernet <–> RS232). Wersja –TM (TTL Master) nie jest wyposażona w konwerter poziomów RS232 i służy do zabudowy wewnątrz istniejącego urządzenia. Może za to dostarczyć stabilizowanego napięcia zasilania 3,3 V (max. 100 mA). Wersja –TS (TTL Slave) musi natomiast być zasilana z zewnętrznego stabilizowanego źródła napięcia 3,3 V.

Tibbo przedstawiło też kompletne rozwiązanie przemysłowego kontrolera (ideowo zbliżonego do PLC) z interfejsem Ethernet – DS1000 (**rys. 5**). Został on opracowany w oparciu o moduł EM1000 wraz z niezbędnymi do jego działania peryferiami. Urządzenie posiada obudowę wykonaną z profilu aluminiowego i jest wyposażone w uchwyt montażowy na szynę DIN. Obudowa zawiera wewnętrzne sloty, w które można wsuwać płytki drukowane stanowiące moduły rozszerzeń. Mogą ty być elementy produkowane prze Tibbo, jak i własne konstrukcje. Zapowiedziane są już moduły komunikacji bezprzewodowej. Z przodu kontroler posiada wymienny panel z różnymi rodzajami złącz. Konkretna konfiguracja/wygląd zależy od zastosowanych płyt rozszerzeń. Dodatkowo urządzenie można wyposażyć w panel wodoodporny, w którym znajdują się otwory wyprowadzające przewody. Na **rys. 6** przedstawiono poglądowy szkic budowy kontrolera DS1000.



Rys. 6. Budowa wewnętrzna kontrolera DS1000

Podsumowanie

Rozszerzanie oferty przez Tibbo o wyłącznie programowalne moduły stanowi sygnał, iż pomysł z wprowadzeniem własnego języka programowania był dobrym posunięciem. Zapowiedzi firmy na temat wprowadzania modułów komunikacji bezprzewodowej (jak na razie nie wiadomo, jaki to będzie standard transmisji) są interesujące o tyle, iż oprócz Ethernetu i RS232 dostaniemy do dyspozycji zupełnie nowy kanał transmisji danych. To z kolei pozwoli na coraz efektywniejsze aplikacje i konstrukcje. Z drugiej jednak strony pozostawia mniej miejsca dla konstruktorów w zakresie projektowania sprzętu, a to przecież lubimy robić najbardziej.

Marcin Chruściel, EP
marcin.chrusciel@ep.com.pl

Dodatkowe informacje

Soyter Sp. z o.o., Klaudyn, ul. Ekologiczna 14/16, 05-080 Izabelin, tel. 022 752 82 55, fax. 022 722 05 50, handlowy@soyter.pl, www.soyter.pl