Test szybkich portów GPIO mikrokontrolerów z serii LPC214x firmy NXP (Philips)

W EP5/2005 zamieściliśmy artykuł „Czy ARM zawsze znaczy szybki? Badanie szybkości pracy portów mikrokontrolera LPC2114 firmy Philips.”. Treść artykułu dotyczyła niedoróbki, jakiej „dopuścili się” inżynierowie firmy Philips Semiconductor (obecnie NXP) projektujący strukturę mikrokontrolerów serii LPC2000, polegającej na powolnej pracy portów I/O. Niniejszy artykuł jest suplementem wspomnianego opracowania.

Po przeobrażeniach kapitałowych firma Philips Semicontuctor przekazała produkcję mikrokontrolerów firmie NXP. Już pod zmienionym szyldem powstał m.in. nowy mikrokontroler ARM oznaczony jako LPC2148, w którym niedoróbki poprzedników zostały poprawione. W artykule zawarto informację o tym, jak można wykorzystać nowe, szybkie porty I/O. Zaprezentowano również praktyczny test szybkich portów I/O wspomnianego wyżej układu.

Powolne porty w szybkim mikrokontrolerze

Czytelnicy zainteresowani tematyką mikrokontrolerów ARM zapewne pamiętają, że powodem powolnej pracy portów pierwszych mikrokontrolerów serii LPC2000 było umieszczenie ich w przestrzeni adresowej standardowych układów peryferyjnych. Rozwiązanie takie skutkowało tym, że dostęp do portu odbywał się przez mostek pomiędzy magistralami AHB i VPB. Powodowało to wydłużenie fazy *execute* tych spośród instrukcji zapisu i odczytu, które odwoływały się do przestrzeni portów GPIO *(General Purpose Input Output)*. Inżynierowie firmy NXP poprawili konstrukcję mikrokontrolerów z rdzeniem ARM produkowanych przez tę firmę i począwszy od serii LPC214x, mikrokontrolery te wolne są od opisanej wady.

Na **rys. 1** przedstawiono schemat blokowy układów serii LPC214x. Dla zachowania kompatybilności z poprzednimi seriami mikrokontrolerów rodziny LPC2000, mikrokontrolery serii LPC214x zachowały zwykłe porty GPIO umieszczone w przestrzeni standardowych urządzeń peryferyjnych i komunikujące się z rdzeniem za pośrednictwem magistrali VPB, mostka AHB/VPB i magistrali AHB. Zachowanie tych portów pozwala konstruktorom stosować mikrokontrolery serii LPC214x tam, gdzie wcześniej używano starszych mikrokontrolerów rodziny LPC2000. Zamiana może być dokonana bez konieczności wprowadzania zmian w układzie sprzętowym bądź oprogramowaniu (dodatkowe sztuczne opóźnienia). Możliwe jest zatem bezproblemowe zachowanie odpowiednich zależności czasowych w opracowanych wcześniej układach, w których szybkie porty GPIO mogłyby być zbyt szybkie dla współpracujących z nimi układów elektronicznych. Jednocześnie, mikrokontrolery serii LPC214x posiadają odrębny blok szybkich portów GPIO (*Fast GPIO*), które komunikują się z rdzeniem ARM7 za pośrednictwem szybkiej magistrali *ARM7 local bus*.

Sposób wykorzystania szybkich portów GPIO mikrokontrolerów serii LPC214x

Jako porty szybkie, pracować mogą porty P0 i P1 mikrokontrolerów serii LPC214x, przy czym, ze względu na zastosowaną obudowę LQFP64, w przypadku mikrokontrolerów tej serii nie są dostępne wyprowadzenia odpowiadające bitom P0.24, P0.26 i P0.27 oraz P1.0... P1.15 (jest to spowodowane po prostu zbyt małą liczbą wyprowadzeń obudowy 64–końcówkowej). Szybkie porty posiadają własny zestaw rejestrów SFR, podobny do zestawów rejestrów związanych ze standardowymi (powolnymi) portami GPIO. Najważniejszymi spośród nich są ($n = \{0,1\}$):

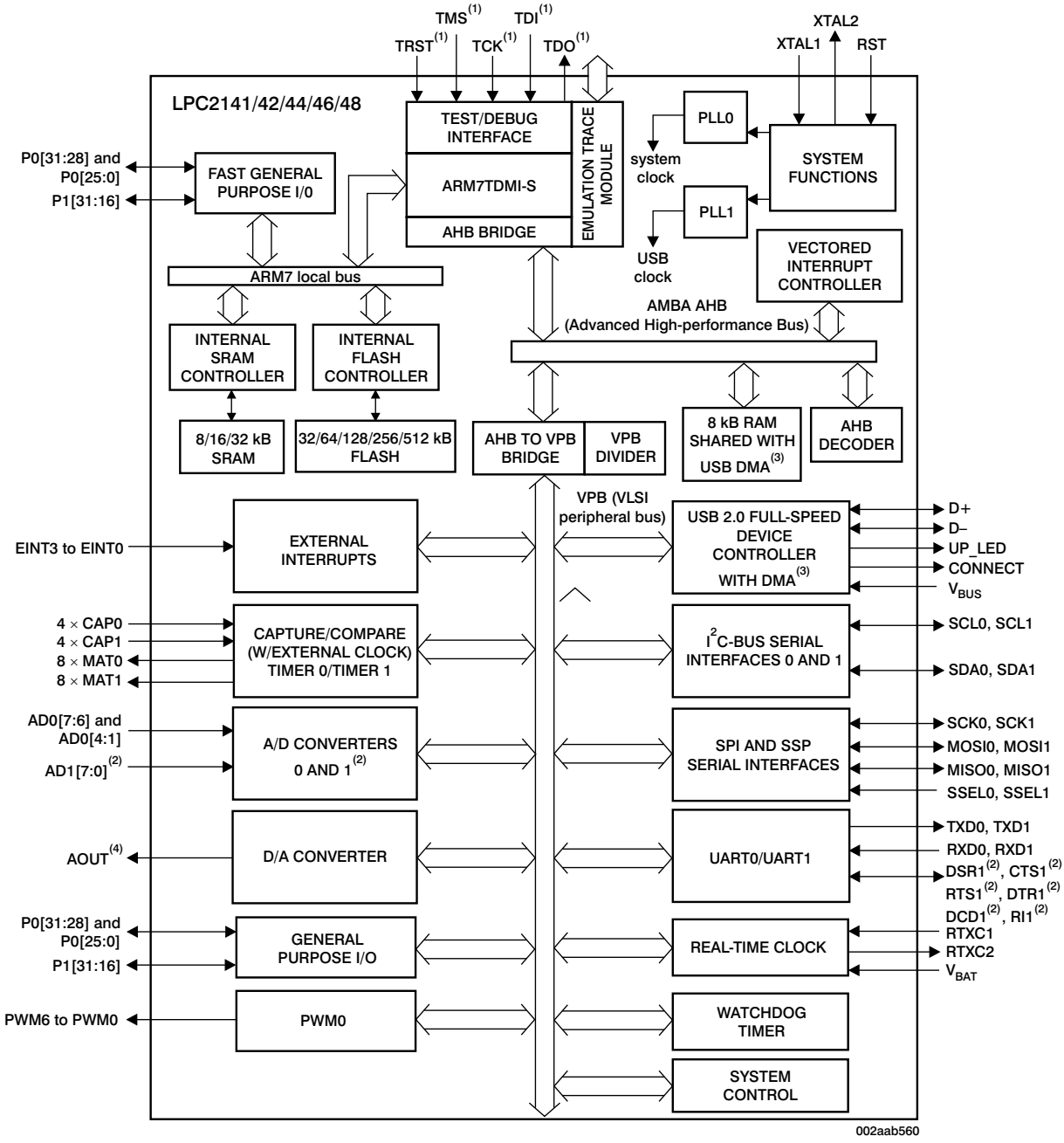
- *FIOnDIR* – rejestr kierunku portu *n*. Nadanie wartości 0 danemu bitowi oznacza, że odpowiadająca mu linia portu pracuje jako wejście. W przeciwnym wypadku – jako wyjście;
- *FIOnPIN* – rejestr portu *n*. Zapis do tego rejestru zmienia stan portu, odczyt pozwala poznać stan portu GPIO;
- *FIOnSET* – zapis wartości 1 do bitów tego rejestru powoduje nadanie odpowiadającym im wyprowadzeniom portu *n* stanu wysokiego. Zapis wartości 0 nie powoduje zmiany stanu portu;
- *FIOnCLR* – zapis wartości 1 do bitów tego rejestru powoduje nadanie odpowiadającym im wyprowadzeniom portu *n* stanu niskiego. Zapis wartości 0 nie powoduje zmiany stanu portu;
- *FIOnMASK* – rejestr maski portu. Pozwala on na określenie, które linie (fizyczne) portu *n* mogą być zmieniane za pośrednictwem rejestrów *FIOnPIN*, *FIOnSET* i *FIOnCLR* lub odczytywane za pośrednictwem rejestru *FIOnPIN*. Zapis i odczyt jest możliwy tylko do/z tych linii portu, które odpowiadają bitom rejestru *FIOnMASK* o wartości zero.

Ze względu na to, że zarówno standardowe, jak i szybkie porty GPIO dzielą między siebie te same wyprowadzenia mikrokontrolera, konieczne jest określenie, czy dany port pracuje jako zwykły, czy jako szybki port GPIO. Służy do tego rejestr *SCS* (*System Control and Status flags register*). Bit numer 0 określa tryb pracy portu P0, zaś bit numer 1 – tryb pracy portu P1. Nadanie danemu bitowi (numer 0 lub 1) wartości 0 spra-

wia, że odpowiadający mu port GPIO pracuje jako port standardowy. Nadanie wartości 1 powoduje wejście portu w tryb *Fast GPIO*.

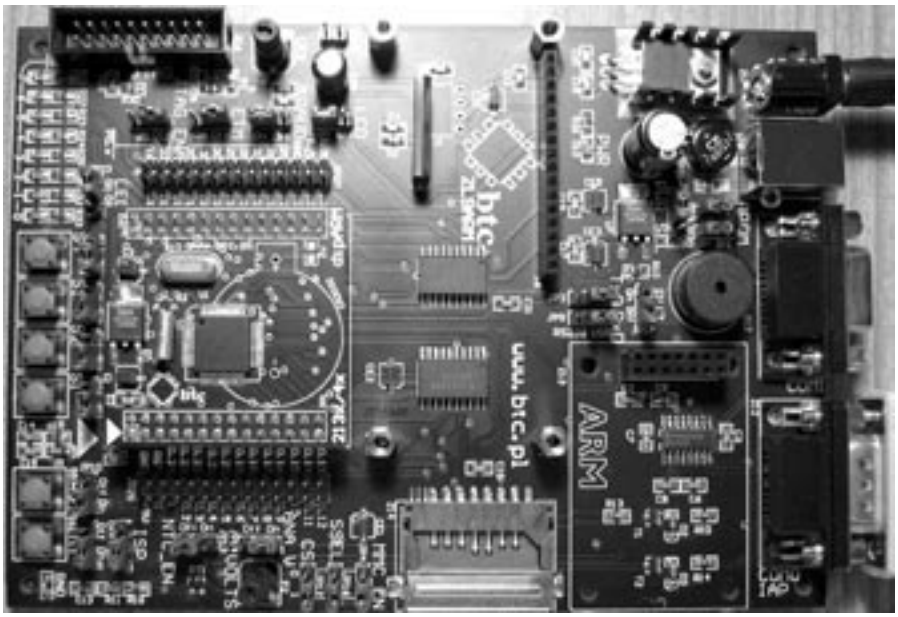
Przykład użycia szybkiego portu GPIO1 mikrokontrolera LPC2148. Pomiar czasu wykonania instrukcji zapisu i odczytu w przestrzeni szybkich portów GPIO

Do testów wykorzystano połączone zestawy ewaluacyjne ZL9ARM i ZL10ARM firmy BTC, które pozwalają na szybkie zapoznanie się z mikrokontrolerami serii LPC214x. Na **fot. 2** przedstawiono ich widok podczas dokonywania pomiarów. Podobnie jak w przypadku mikrokontrolera LPC2114 (artykuł *„Czy ARM zawsze znaczy szybki? Badanie szybkości pracy portów mikrokontrolera LPC2114 firmy Philips”* zamieszczony w EP5/2005) zmierzono czasy wykonania instrukcji zapisu i odczytu do/z portu GPIO, oczywiście z tą różnicą, że tym razem wykorzystano szybki port I/O – FGPIO1. Wykorzystano tę samą, co wcześniej, metodę pomiaru czasu trwania instrukcji. Instrukcje, których czas wykonania mierzono, dodano do pętli, w której naprzemiennie zmieniany jest stan portu. Dzięki temu okres tych zmian wydłuża się o czas ich wykonania. Stworzono prosty program hybrydowy, w którym główna część (odpowiedzialna za konfigurację procesora) została napisana w języku C (kompilator ARM–GCC 4.0.1), zaś właściwa pętla pomiarowa została



(1) Piny współdzielone z GPIO
(2) Tylko LPCC2144/6/8
(3) Kontroler USB DMA z 8 kB RAM-u dostępną jako pamięć RAM ogólnego zastosowania i/lub DMA dostępne tylko w LPC2146/8.
(4) Tylko LPC2142/4/6/8

Rys. 1. Schemat blokowy mikrokontrolera z serii LPC214x



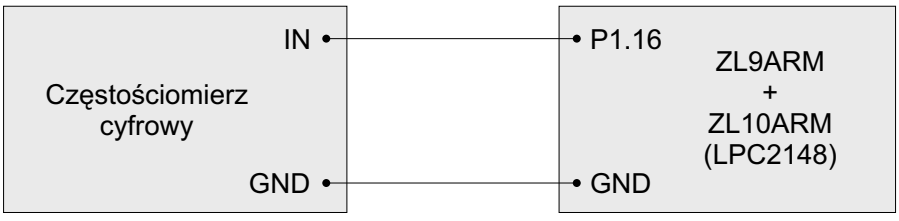
Fot. 2. Połączone zestawy ZL9ARM i ZL10ARM podczas testów

zaimplementowana w asemblerze ARM7. Do edycji programu wykorzystano środowisko uVision3 firmy Keil. Program uruchomiono przy częstotliwości taktowania rdzenia równej 60 MHz i całkowicie włączonym module MAM (*Memory Acceleration Module*), co zapewniło najszybszy z możliwych odczyt instrukcji z pamięci Flash.

List. 1. Ustawienie portu 1 w tryb Fast GPIO

```
//////////////////////////////////////////////////////////
// Function:     LPC214x fast GPIO test
// Target:       LPC2148
// Board:     ZL9ARM + ZL10ARM
// Quartz:       12MHz
// Compiler: ARM-GCC 4.0.1
// Author:       Arkadiusz Antoniak @ 2006, Poland
// Copyright:    (c) Arkadiusz Antoniak, 2006
//////////////////////////////////////////////////////////
#include <LPC214x.H>
//toggle assembly function
extern void toggle(void);
//*******************************
// MAIN
//*******************************
int main(void)
{
    //Peripherials' frequency pclk = cclk/4 = 15MHz
    VPBDIV&=0xFFFFFFFC;
    VPBDIV|=0x00;

    //Enable PORT1 as fast GPIO
    SCS|=(1<<1);

    //Fast IO mask - enable access to P1.16 ... P1.23 pins
    FIO1MASK = ~(0xFF << 16);

    //Set PORT1 direction
    FIO1DIR = 0x00FF0000;

    //Toggle P1.16 ... P1.23
    toggle();

    while(1);
}
```

Na **list. 1** przedstawiono część napisaną w języku C. Jest to krótki program, w którym najpierw jest odpowiednio konfigurowana szybkość taktowania peryferiów (rejestr VPBDIV), a następnie port numer 1 konfigurowany jest do pracy w trybie szybkim. Nadanie rejestrowi *FIO1MASK* wartości 0xFF00FFFF sprawia, że możliwy jest dostęp do linii P1.16...P1.23 traktowanych jako szybki port I/O. Tuż przed wywołaniem funkcji *toggle()* linie P1.16...P1.23 ustawiane są jako wyjścia. Funkcja *toggle()* została zadeklarowana z dyrektywą *extern* po to, aby jej ciało można było umieścić w innym pliku źródłowym (w tym przypadku jest to plik toggle.s). Kompilatora nie interesuje w jakim języku została ona zaimplementowana – wystarczy, aby linker miał informację o tym, gdzie jej szukać. Na **list. 2** przedstawiono kod funkcji *toggle()*. Dyrektywa *.global toggle* pozwala wywołać ją z innych plików projektu, zaś zgodność nazw funkcji w plikach *.c i *.s daje linkerowi jednoznaczną informację o tym, gdzie się ona znajduje.



Rys. 3. Schemat układu testowego

Pierwszą część funkcji *toggle()* stanowi inicjalizacja rejestru R3, który przechowuje adres rejestru szybkiego portu numer 1 – *FIO1PIN*. Adres ten jest równy 0x3FFFC034. Rejestry R1 i R2 przechowują wartości, które będą wpisywane do *FIO1PIN* powodując odpowiednio wygaszenie lub zapalenie linii P1.23...P1.16. To, że adres rejestru *FIO1PIN* nie jest bezpośrednio wpisywany do rejestru R3 wynika stąd, że asembler akceptuje jedynie takie stałe 32–bitowe, z których da się otrzymać stałe 8–bitowe po rotacji o parzystą liczbę miejsc (0, 2, 4, ..., 30). Na przykład – nie da się przesłać do rejestru stałej 0x1234. Da się natomiast wpisać tam stałą 0xC000003F, gdyż po rotacji o 2 miejsca w lewo otrzymujemy z niej 8–bitową liczbę 0xFF. W niniejszym przypadku wartość 0x3FFFC034 jest otrzymywana poprzez odjęcie od wartości 0x40000000 liczb 0x3F00 i 0xCC (każda z wymienionych liczb spełnia podany wyżej warunek).

Po fazie inicjalizacji następuje właściwa pętla pomiarowa *toggle_loop*. Stanowią ją dwie instrukcje *str* wpisujące do rejestru *FIO1PIN* zawartość rejestrów R1 i R2. Po nich występuje instrukcja mierzona, a następnie skok zamykający pętlę. Należy zauważyć, że jedną z instrukcji mierzonych może być instrukcja zapisu do portu (*str R2,[R3]*). Ze względu na to, że wpisuje ona tam to samo co poprzednia instrukcja *str R2,[R3]*, wpływa ona na okres przebiegu wyjściowego jedynie czasem swojego wykonywania.

Schemat układu testowego przedstawiono na **rys. 3**. Pomiary polegały na odkomentowaniu wybranej instrukcji i zmierzeniu częstotliwości na linii P1.16. Wyniki zamieszczono w **tab. 1**. Jak widać, w przeciwieństwie do starszych układów rodziny LPC2000, w tym przypadku czas wykonywania instrukcji przesłań do portu i do pamięci RAM nie różni się (przypomnijmy – zapis i odczyt do/z pamięci RAM zajmuje odpowiednio 2 i 3 cykle, czyli tyle samo co dostęp do szybkich portów GPIO). W starszych wersjach mikrokontro-

Tab. 1. Wyniki pomiarów (cclk=60 MHz)

Instrukcja mierzona	Częstotliwość [kHz]	Okres T [ns]	Okres T [cykle]	ΔT=T-116,7 [ns]	Liczba cykli wykonywania fazy Execute
- brak instrukcji	8572	116,(6)	7	0	-
str R2,[R3] rejestr -> port	6666	150	9	33,(3)	2
ldr R4,[R3] rejestr <- port	6000	166,(6)	10	50	3

Fot. 4. Maksymalna częstotliwość przełączania

List. 2. Funkcja toggle()

```
        .global toggle
toggle:
        mov     R3,#0x40000000
        sub     R3,R3,#0x3F00
        sub     R3,R3,#0xCC          @ R3 = 0x3FFFC034 (FIO1PIN)
        mov     R1,#0
        mov     R2,#0x00FF0000

toggle_loop:
        str     R1,[R3]              @ P1.16 ... P1.23 = 00000000b
        str     R2,[R3]              @ P1.16 ... P1.23 = 11111111b

        @ -- testing instructions execution time --
        @ldr R4,[R3]                 @ read P1 (FIO1PIN)
        str  R2,[R3]                 @ write P1 (FIO1PIN)

        btoggle_loop
```

lerów serii LPC2000 zapis i odczyt do/z portu trwają odpowiednio 7 i 8 cykli. Należy odnotować, że odczyt z portu jest o 1 cykl wolniejszy niż zapis.

Na **fot. 4** zamieszczono fotografię częstościomierza podczas pomiaru maksymalnej częstotliwości przełączania portu (przy usuniętych obu instrukcjach mierzonych). Częstotliwość ta wynosi 1/7 częstotliwości taktowania rdzenia *cclk*. Skok jest wykonywany w ciągu trzech cykli, zaś każdy z zapisów do portu zajmuje 2 cykle. Stąd też maksymalna możliwa do uzyskania częstotliwość przełączania portu wynosi 60 [MHz] /4=15 [MHz]. Oczywiście, przebieg o takiej częstotliwości pojawi się na wyjściu portu, gdy fragment programu generujący przebieg wyjściowy będzie miał postać szeregu instrukcji postaci *str R2,[R3]* nie ujętych w pętlę (skok wprowadza dodatkowe opóźnienie).

Podsumowanie

Wyposażenie mikrokontrolerów serii LPC214x w szybkie porty GPIO jest przykładem, jak *feedback* ze strony konstruktorów i użytkowników elementów elektronicznych może skłonić producenta do poprawienia ich funkcjonalności, a zatem konkurencyjności na dynamicznym rynku mikrokontrolerów z rdzeniem ARM. W mikrokontrolerach wyposażonych w ten rdzeń szybkie porty nie są w prawdzie najistotniejszym czynnikiem decydującym o ich użyteczności (lub jej braku), nie mniej jednak istnieje szeroki zakres zastosowań tych układów, gdzie szybkość pracy portów GPIO jest bardzo istotna. Tym bardziej cieszy, że historia drobnego, acz dokuczliwego w wielu przypadkach błędu, została zamknięta.

Arkadiusz Antoniak, EP
arkadiusz.antoniak@ep.com.pl