Mikrokontrolery z rdzeniem ARM, część 16

System przerwań c.d.

Przykładowe programy obsługi przerwań

Wszystkie programy przedstawione w bieżącym odcinku reagować będą na wciśnięcie klawisza S5, który jest podłączony do linii P0.14 mikrokontrolera. W programie przedstawionym na **list. 5** (*ep6b.zip*, na CD–EP3/2007B) w momencie wciśnięcia klawisza S5 diody będą na przemian załączane i wyłączane po każdym wciśnięciu klawisza.

Funkcja obsługi przerwania została zadeklarowana z atrybutem FIQ, informującym kompilator że będzie ona wywołana w reakcji na przerwanie FIQ. Plik startowy *boot.s* musi być skonfigurowany tak, aby w miejscu wektora przerwania FIQ umieszczona była funkcja *FIQIntHandler*, co realizuje poniższa linijka kodu:

```
FIQ_Addr:
.word       FiqIntHandler
```

Należy również pamiętać o zapewnieniu kilkudziesięciu bajtów stosu dla trybu FIQ, co realizuje poniższa linijka kodu:

```
.equ        FIQ_Stack_Size, 0x00000020
```

Program rozpoczyna się od wykonania funkcji *main()*, w której najpierw inicjalizowane są linie sterujące diodami mikrokontrolera w kierunku wyjścia. Następnie ustawiany jest rejestr PINSEL0 tak, aby linia P0.14 portu pełniła rolę wejścia EINT1. Kolejną rzeczą, jaką musimy zrobić to ustawienie przerwań zewnętrznych tak, aby > 17) były one wyzwalane opadającym zboczem, czego dokonujemy poprzez ustawienie pierwszego bitu rejestru EXTMODE oraz wyzerowanie pierwszego bitu rejestru EXTPOLAR. Na zakończenie konfiguracji przerwań zewnętrznych kasujemy znacznik zgłoszenia przerwania, który mógł zostać przypadkiem zmieniony podczas konfigurowania przerwań. Pozostało nam jeszcze skonfigurowanie kontrolera przerwań VIC: Przerwanie EINT1 kwalifikowane jest jako przerwanie FIQ oraz odblokowywane jest zezwolenie na przerwanie. Na zakończenie włączane są przerwania w jednostce centralnej, po czym program kończy działanie. W momencie wciśnięcia klawisza S5 zostanie zgłoszone przerwanie FIQ w wyniku, czego zostanie wywołana funkcja jego obsługi *FiqIntHandler*. W funkcji tej zmieniany jest stan portu diod LED na przeciwny, a na koniec obsługi przerwania kasowana jest flaga zgłoszenia przerwania EINT1 poprzez wpisanie do rejestru EXTINT jedynki na pierwszym bicie.

Wiemy już jak skonfigurować system przerwań, aby wybrane przerwanie było zgłaszane jako FIQ, teraz pokażemy, co trzeba zrobić, aby to samo przerwanie zewnętrzne zostało zakwalifikowane jako wektoryzowane przerwanie IRQ. Na **list. 6** (*ep6c.zip*, publikujemy na CD–EP3/2007B) przedstawiono program, który zlicza liczbę wystąpień przerwania EINT1 (klawisz S5), a następnie za pomocą klasy obsługującej wyświetlacz LCD zaprezentowanej w poprzednim odcinku kursu wyświetla tę wartość.

Przerwanie *IrqInt1Handler* zadeklarowano z atrybutem IRQ, informującym, że będzie to procedura obsługi przerwania IRQ. Tym razem nie musimy modyfikować pliku startowego *boot.s*, ponieważ adres procedury obsługi przerwa-

List. 5. Program powodujący po wciśnięciu klawisza S5 przemienne włączane i wyłączane LED–ów

```
#include "lpc213x.h"
#include "armint.h"

//Definicja LEDOW
#define LEDS (0xFF<<16)
#define LEDDIR IO1DIR
#define LEDSET IO1SET
#define LEDCLR IO1CLR
#define LEDPIN IO1PIN

#define EINT1_SEL (2<<28)
#define P014_SEL_MASK (3<<28)
#define EINT1_VIC (1<<15)

//Przerwanie szybkie (FIQ)
extern "C" void FiqIntHandler(void) __attribute__ ((interrupt("FIQ")));

void FiqIntHandler(void)
{
  //Zmien stan LEDOW na przeciwny
  IO1PIN ^= LEDS;
  //Kasuj zrodlo przerwania
  EXTINT = EXTINT_EINT1;
}


/* Funkcja glowna main */
int main(void)
{
  //Diody LED jako wyjscie
  LEDDIR |= LEDS;
  //Wylacz LEDY
  LEDCLR = LEDS;
  //Uruchomienie na P0.14 funkcji alternatywnej INT1
  PINSEL0 &= ~P014_SEL_MASK;
  PINSEL0 |= EINT1_SEL;
  //Przerwanie zboczem
  EXTMODE |= EXTINT_EINT1;
  //Zbocze opadajace
  EXTPOLAR &= ~EXTINT_EINT1;
  //Kasuj wystapienie przerwania (1 kasuje)
  EXTINT = EXTINT_EINT1;
  //Ustawienie INT1 jako FIQ
  VICIntSelect |= EINT1_VIC;
  //Zalaczenie przerwania
  VICIntEnable = EINT1_VIC;
  //Zalacz FIQ
  enable_fiq();
  return 0;
}
```

List. 6. Program zliczający liczbę wystąpień przerwania EINT1 (klawisz S5), a następnie za pomocą klasy obsługującej wyświetlacz LCD

```
#include "lpc213x.h"
#include "armint.h"
#include „CLcdDisp.h"

#define EINT1_SEL (2<<28)
#define P014_SEL_MASK (3<<28)
#define EINT1_VIC (1<<15)
#define EINT1_VIC_BIT 15
#define VIC_IRQSLOT_EN (1<<5)


//Przerwanie wektoryzowane IRQ
void IrqInt1Handler(void) __attribute__ ((interrupt("IRQ")));

static volatile unsigned int EintCnt;

void IrqInt1Handler(void)
{
  //Zmien stan LEDOW na przeciwny
  EintCnt++;
  //Kasuj zrodlo przerwania
  EXTINT = EXTINT_EINT1;
  //Informacja dla VIC - koniec procedury przerwania
  VICVectAddr = 0;
}


CLcdDisp cout;
/* Funkcja glowna main */
int main(void)
{
  //Uruchomienie na P0.14 funkcji alternatywnej INT1
  PINSEL0 &= ~P014_SEL_MASK;
  PINSEL0 |= EINT1_SEL;
  //Przerwanie zboczem
  EXTMODE |= EXTINT_EINT1;
  //Zbocze opadajace
  EXTPOLAR &= ~EXTINT_EINT1;
  //Kasuj wystapienie przerwania (1 kasuje)
  EXTINT = EXTINT_EINT1;
  //Wektor 0
  VICVectAddr0 = (unsigned int)IrqInt1Handler;
  VICVectCntl0 = EINT1_VIC_BIT | VIC_IRQSLOT_EN;
  //Odblokuj EINT1 w VIC
  VICIntEnable = EINT1_VIC;
  //Zalacz IRQ
  enable_irq();

  cout << "ARM - Vect IRQ";
  cout << pos(1,2) << "IntCnt=";
  while(1)
  {
     cout << pos(8,2) << EintCnt;
  }
  return 0;
}
```

nia będzie przekazany do slotu 0 kontrolera VIC. Wykonanie programu rozpoczyna się od inicjalizacji funkcji alternatywnej portu P0.14 EINT1, a następnie tak samo jak w poprzednim przykładzie ustawiane są przerwania zewnętrzne jako uruchamiane opadającym zboczem. W programie wykorzystywać będziemy slot zerowy (o najwyższym priorytecie) kontrolera, do którego podstawiamy adres procedury obsługi przerwania EINT1:

VICVectAddr0 = (unsigned int)IrqInt1Handler;

Następnie do rejestru konfiguracyjnego wpisujemy numer kanału przerwania EINT, które będzie przyporządkowane do tego slotu oraz włączamy ten slot. Na koniec odblokowujemy w kontrolerze VIC przerwanie EINT1 oraz globalnie przerwania IRQ w jednostce centralnej. Po wykonaniu inicjalizacji program wchodzi do pętli nieskończonej, w której na bieżąco wyświetla stan zmiennej *EintCnt*.

W momencie wciśnięcia klawisza S5 zostaje wywołana procedura obsługi przerwania *IrqInt1Handler*, w której zwiększany jest licznik przerwań, a następnie jest kasowany znacznik zgłoszenia przerwania. Ostatnią czynnością, jaką realizuje ta procedura jest zapis wartości 0 do rejestru *VICVectAddr*, co jest informacją dla kontrolera VIC, że procedura obsługi przerwania dobiegła końca.

Do przedstawienia pozostał nam jeszcze ostatni tryb obsługi przerwań, czyli niewektoryzowane przerwania IRQ. Działanie programu przedstawionego na **list. 7** (*ep6d.zip*, dostępny na CD–EP3/2007B) jest analogiczne jak programu poprzednio opisanego, z tym, że do zwiększania stanu licznika prze-

rwań wykorzystywać będziemy nie wektoryzowane przerwania IRQ.

Inicjalizacja systemu przerwań zewnętrznych odbywa się analogicznie jak w poprzednim programie, różnica pojawia się tylko w przypadku ustawień kontrolera VIC. Adres procedury obsługi przerwania wpisujemy do rejestru *VICDefVectAdr* zawierającego adres procedury obsługi przerwań niewektoryzowanych, a następnie odblokowujemy przerwanie *EINT1* ustawiając bit odpowiadający kanałowi przerwania *EINT1* w rejestrze *VICIntEnable*. Brak aktywacji danego kanału przerwania w slocie wektoryzowanym powoduje, że w momencie wystąpienia przerwania zostanie ono zakwalifikowane jako nie wektoryzowane i wywołana zostanie procedura obsługi przerwania niewektoryzowanego. Sama procedura obsługi przerwania jest również zadeklarowana z modyfikatorem, IRQ. W ciele procedury obsługi przerwania następuje sprawdzenie 24 rejestru *VICIRQStatus* w celu określenia źródła przerwania. Jeżeli przerwanie pochodziło od kanału *EINT1*, wówczas kasowana jest flaga zgłoszenia przerwania w rejestrze EXTINT oraz wysyłana jest informacja o zakończeniu obsługi przerwania do kontrolera VIC.

List. 7. Program o funkcji identycznej z programem pokazanym na list. 6, przy czym do zwiększania stanu licznika przerwań wykorzystano nie wektoryzowane przerwania IRQ

```
#include "lpc213x.h"
#include "armint.h"
#include „CLcdDisp.h"

#define EINT1_SEL (2<<28)
#define P014_SEL_MASK (3<<28)
#define EINT1_VIC (1<<15)
#define EINT1_VIC_BIT 15
#define VIC_IRQSLOT_EN (1<<5)

//Przerwanie wektoryzowane IRQ
void IrqInt1Handler(void) __attribute__ ((interrupt("IRQ")));

static volatile unsigned int EintCnt;

void IrqInt1Handler(void)
{
  if(VICIRQStatus & EINT1_VIC)
  {
     //Zmien stan LEDOW na przeciwny
     EintCnt++;
     //Kasuj zrodlo przerwania
     EXTINT = EXTINT_EINT1;
  }
  //Informacja dla VIC - koniec procedury przerwania
  VICVectAddr = 0;
}

CLcdDisp cout;
/* Funkcja glowna main */
int main(void)
{
  //Uruchomienie na P0.14 funkcji alternatywnej INT1
  PINSEL0 &= ~P014_SEL_MASK;
  PINSEL0 |= EINT1_SEL;
  //Przerwanie zboczem
  EXTMODE |= EXTINT_EINT1;
  //Zbocze opadajace
  EXTPOLAR &= ~EXTINT_EINT1;
  //Kasuj wystapienie przerwania (1 kasuje)
  EXTINT = EXTINT_EINT1;
  //Wektor default
  VICDefVectAddr = (unsigned int)IrqInt1Handler;
  //Zalaczenie przerwania
  VICIntEnable = EINT1_VIC;
  //Zalacz IRQ
  enable_irq();

  cout << "ARM - NoVect IRQ";
  cout << pos(1,2) << "IntCnt=";
  while(1)
  {
     cout << pos(8,2) << EintCnt;
  }
  return 0;
}
```

Zakończenie

W ten sposób zapoznaliśmy się z obsługą przerwań oraz nauczyliśmy się obsługiwać przerwania zewnętrzne EINT0...EINT3. System przerwań mikrokontrolerów LPC213x jest bardzo bogaty i jest on bardziej zbliżony do sposobu obsługi systemu przerwań w komputerach klasy PC niż w typowych mikrokontrolerach 8–bitowych. Zgłaszane przerwania mogą być zaklasyfikowane w trzech trybach: jako przerwanie FIQ, wektoryzowane IRQ oraz niewektoryzowane IRQ. Jednak najczęstszą konfiguracją, z jaką będziemy mieli do czynienia to: jedno przerwanie FIQ przydzielone dla urządzenia, które będzie wymagać czasowo krytycznej reakcji, oraz kilkanaście wektoryzowanych przerwań IRQ, dla mniej krytycznych przerwań. Tylko w bardzo zaawansowanych aplikacjach, gdzie liczba potrzebnych przerwań przekroczy 16 zmusi nas do użycia najwolniejszego nie wektoryzowanego przerwania IRQ. Mając już odpowiednią dawkę wiedzy na temat systemu przerwań w następnym odcinku zajmiemy się układami czasowo–licznikowymi mikrokontrolera, układem *watchdog* oraz zegarem RTC.

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