Mikrokontrolery Xmega (3)

Sygnały zegarowe

W mikrokontrolerach ATmega i ATtiny układ dystrybucji sygnałów zegarowych był wręcz prymitywny. W szczególności w mikrokontrolerach starszej generacji, takich jak ATmega 8, mogliśmy wybrać źródło sygnału zegarowego za pomocą fusebitów, którego nie można było zmienić, gdy mikrokontroler pracował. Ponadto, błędne ustawienie fusebitów mogło prowadzić do „zablokowania" procesora.

W Xmega obwody dystrybucji sygnału zegarowego są bardzo rozbudowane. Użytkownik ma do dyspozycji różne źródła sygnału zegarowego, takie jak umieszczone w strukturze generatory: szybki (32 MHz), energooszczędny (32 kHz) oraz normalny (2 MHz). Ten ostatni domyślnie uruchamia się po włączeniu zasilania. Częstotliwości wyjściowe generatorów można podzielić preskalerem lub pomnożyć wbudowanym układem PLL. Oprócz tego, można dołączyć różne kwarce, a w razie uszkodzenia kwarcu, procesor samoczynnie przełączy się na wbudowany generator RC. Mało tego – podczas pracy możemy zmieniać nie tylko częstotliwość zegara, ale również źródło sygnału. Różne peryferia mogą być taktowane z różnymi częstotliwościami, a niektóre z nich mogą pracować nawet z częstotliwością 128 MHz!

Uproszczony schemat układu dystrybucji sygnałów zegarowych przedstawiono na **rysunku 1**. Po wybraniu jednego z pięciu dostępnych źródeł, do dyspozycji są aż trzy preskalery umożliwiające taktowanie poszczególnych peryferiów mikrokontrolera z różnymi częstotliwościami sygnału zegarowego.

CLK_{CPU} to przebieg zegarowy dla rdzenia mikrokontrolera, który może mieć częstotliwość maksymalną 32 MHz. CLK_{PER} taktuje większość peryferiów. CLK_{PER2} i CLK_{PER4} służą do taktowania peryferiów zdolnych do pracy z taktowaniem szybszym od CLK_{CPU}. Oprócz tego, mamy jeszcze osobne przebiegi zegarowe dla RTC i USB, jeśli mikrokontroler jest wyposażony w te peryferia.

Sposób wyboru źródła sygnału zegarowego sprowadza się do realizacji następujących kroków wymaganych do wykonania stabilnej zmiany:

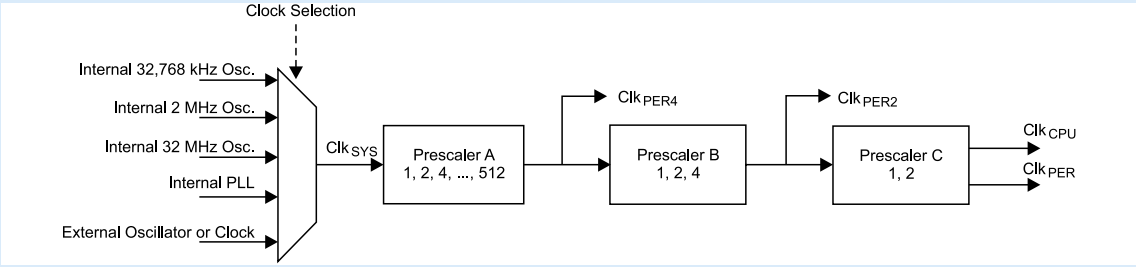
- Konfigurowanie i uruchomienie generatora.
- Oczekiwanie na ustabilizowanie się częstotliwości wyjściowej.
- Wybór źródła.

W tej części kursu napiszemy kilka funkcji, umożliwiających przełączenie źródła sygnału taktującego oraz obserwowanie efektów tej zmiany za pomocą migającej diody LED i wyświetlacza LCD. Wykorzystamy wewnętrzny generator RC 2 MHz, 32 MHz, a także zewnętrzny generator kwarcowy i układ PLL.

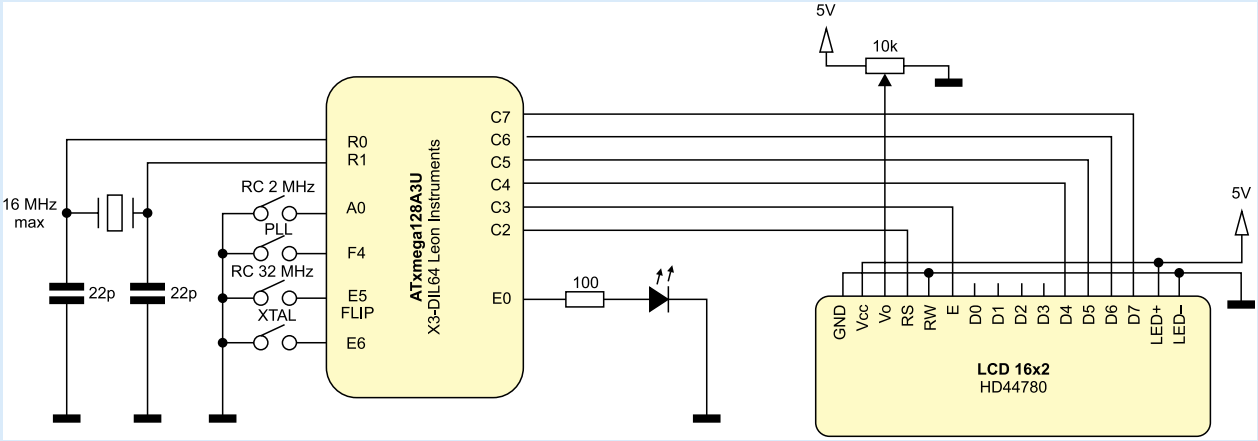
Schemat układu używanego podczas ćwiczeń przedstawiono na **rysunku 2**, a jego wygląd na **fotografii 3**.

Oprogramowanie testowe

Na **listingu 1** zamieszczono funkcję *main()*. Na początek zajmijmy się jej analizą. Zwróćmy uwagę na funkcję



Rysunek 1. Schemat blokowy układu dystrybucji sygnałów zegarowych



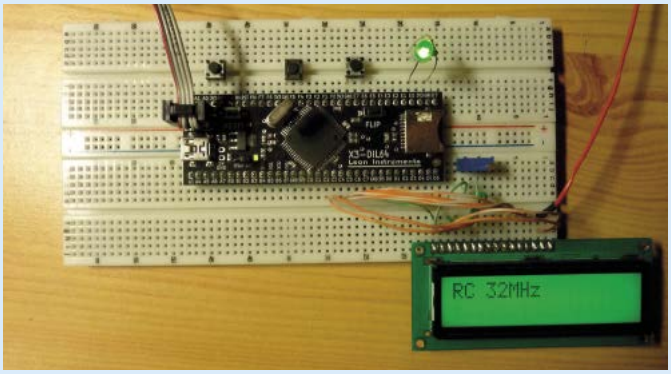
Rysunek 2. Schemat ideowy ważniejszych połączeń układu demonstracyjnego

opóźniającą *_delay_ms(50);* Powoduje ona, że procesor kręci się w pustej pętli nic nie robiąc, aż upłynie żądany czas. Jednak funkcja *_delay_ms()* oblicza ilość potrzebnych cykli na podstawie definicji *#define F_CPU 62000000UL*. W przypadku, kiedy częstotliwość taktowania zmienia się, to pamiętajmy, że standardowe funkcje opóźniające nie uwzględniają tej zmiany, w związku z czym odmierzony czas nie będzie prawidłowy. Zaobserwujemy ten problem w naszym programie testowym – dioda podłączona do E0 będzie mrugać z różną częstotliwością, mimo że w pętli głównej jest *_delay_ms(50)* ze stałym argumentem równym 50.

Taktowanie za pomocą generatora RC o częstotliwości 32 MHz lub 2 MHz

Po wciśnięciu przycisku FLIP dołączonego na płytce X3-DIL64 do pinu E5, zostanie wywołana funkcja uruchamiająca wbudowany generator 32 MHz. Projektanci mikrokontrolerów Xmega zaimplementowali generator RC, który bez dołączania żadnych dodatkowych elementów umożliwia osiągnięcie maksymalnej mocy obliczeniowej. Poza tym, nie wymaga on żadnego konfigurowania, więc możemy od razu przystąpić do jego uruchomienia. Aby to zrobić, musimy wpisać odpowiednią wartość do rejestru kontrolnego oscylatora, a mianowicie do OSC.CTRL. Zwróćmy uwagę na fragment dokumentacji pokazany na **rysunku 4**. Aby uruchomić generator 32 MHz, powinniśmy do rejestru OSC.CTRL wpisać wartość OSC_RC32MEN_bm. Dokumentacja zawiera ostrzeżenie, że musimy poczekać aż częstotliwość oscylatora ustabilizuje się, zanim użyjemy go w charakterze źródła sygnału zegarowego. W tym celu musimy w pętli sprawdzać rejestr OSC.STATUS i czekać tak długo, aż bit OSC_RC32MRDY_bm zostanie ustawiony (**listing 2**). Po wykonaniu tej funkcji, na wyświetlaczu zostanie wyświetlony komunikat o taktowaniu procesora przebiegiem o częstotliwości 32 MHz, a dioda dołączona do pinu E0 będzie migała szybciej.

Uruchomienie wbudowanych generatorów 2 MHz oraz 32 kHz wykonuje się bardzo podobnie – wystarczy zmienić tylko RC32M na RC2M lub RC32K (**listing 3**).



Fotografia 3. Układ demonstracyjny zbudowany na płytce stykowej

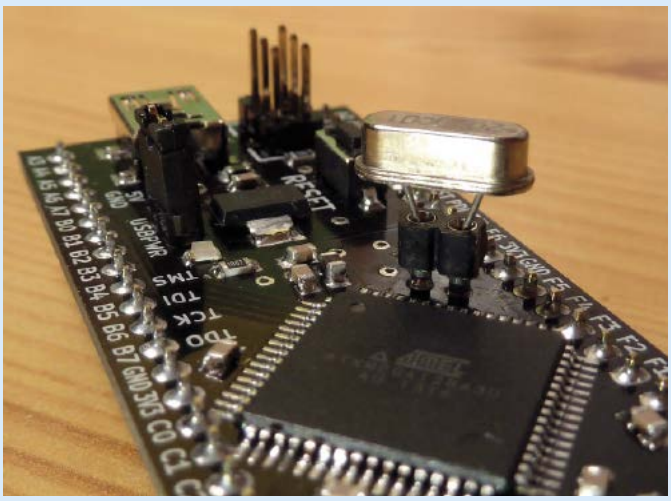
7.10 Register Description – Oscillator

7.10.1 CTRL – Oscillator Control register

Bit	7	6	5	4	3	2	1	0
+0x00	–	–	–	PLLEN	XOSCEN	RC32KEN	RC32MEN	RC2MEN
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	1

- **Bit 7:5 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 4 – PLLEN: PLL Enable**
  Setting this bit enables the PLL. Before the PLL is enabled, it must be configured with the desired multiplication factor and clock source. See "STATUS – Oscillator Status register" on page 94.
- **Bit 3 – XOSCEN: External Oscillator Enable**
  Setting this bit enables the selected external clock source. Refer to "XOSCCTRL – XOSC Control register" on page 95 for details on how to select the external clock source. The external clock source should be allowed time to stabilize before it is selected as the source for the system clock. See "STATUS – Oscillator Status register" on page 94.
- **Bit 2 – RC32KEN: 32.768kHz Internal Oscillator Enable**
  Setting this bit enables the 32.768kHz internal oscillator. The oscillator must be stable before it is selected as the source for the system clock. See "STATUS – Oscillator Status register" on page 94.
- **Bit 1 – RC32MEN: 32MHz Internal Oscillator Enable**
  Setting this bit will enable the 32MHz internal oscillator. The oscillator must be stable before it is selected as the source for the system clock. See "STATUS – Oscillator Status register" on page 94.
- **Bit 0 – RC2MEN: 2MHz Internal Oscillator Enable**
  Setting this bit enables the 2MHz internal oscillator. The oscillator must be stable before it is selected as the source for the system clock. See "STATUS – Oscillator Status register" on page 94.
  By default, the 2MHz internal oscillator is enabled and this bit is set.

Rysunek 4. Rejestr OSC.CTRL oraz generatory dostępne do wyboru



Fotografia 5. Wymiana kwarcu w zestawie X3-DIL64 jest łatwa dzięki zastosowaniu podstawki

Generator kwarcowy

Sercem generatora tego typu jest rezonator kwarcowy, w skrócie zwany kwarcem. Dołączamy go do nóżek R0 i R1 portu R. Są to piny ogólnego przeznaczenia i jeśli nie korzystamy z generatora kwarcowego, możemy ich użyć w innym celu. Oprócz kwarcu, musimy także zastosować kondensatory o niewielkiej pojemności widoczne na schemacie na rys. 2, w sposób znany z mikrokontrolerów ATmega i ATtiny.

Większość producentów płytek testowych narzuca częstotliwość kwarcu lutując go do płytki na stałe. Odlutowanie kwarcu, w szczególności SMD, może spowodować jego uszkodzenie lub oderwanie miedzianych ścieżek od laminatu. Moduł prototypowy X3-DIL64 wyposażono w podstawkę pod kwarc, dzięki czemu użytkownik może szybko i bez lutowania podłączyć taki kwarc, jaki uzna za najlepszy do swoich potrzeb. Wygląd podstawki pokazano na **fotografii 5**.

W mikrokontrolerach XMEGA możemy korzystać z kwarców zegarkowych 32 kHz oraz kwarców o często-

Listing 1. Funkcja główna main()

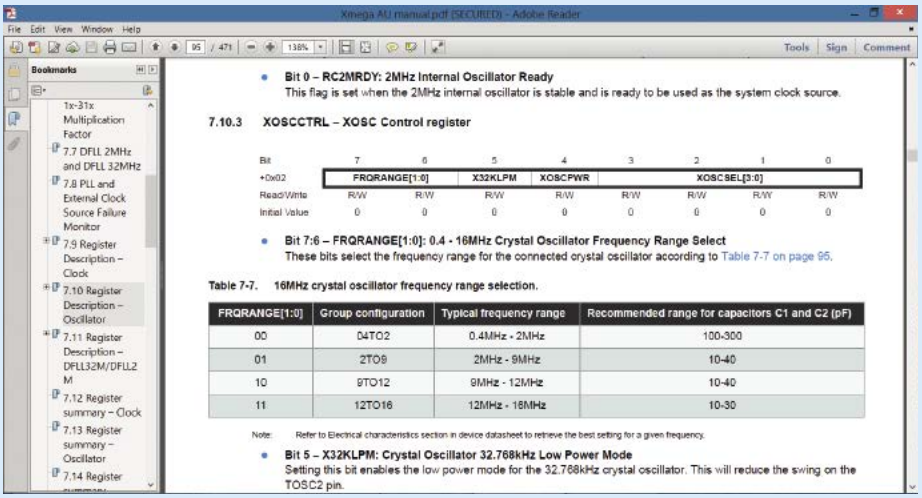
```
#define F_CPU 62000000UL
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include „hd44780.h”

void Osc2MHz(void) {
        OSC.CTRL = OSC_RC2MEN_bm;                   // włączenie oscylatora 2MHz
        while(!(OSC.STATUS & OSC_RC2MRDY_bm));      // czekanie na ustabilizowanie się generatora
        CPU_CCP = CCP_IOREG_gc;                     // odblokowanie zmiany źródła sygnału zegarowego
        CLK.CTRL = CLK_SCLKSEL_RC2M_gc;             // zmiana źródła sygnału zegarowego na RC 2MHz
        LcdClear();                                 // czyszczenie wyświetlacza
        Lcd(„RC 2MHz”);                             // komunikat o uruchomieniu generatora
}

void Osc32MHz(void) {
        OSC.CTRL = OSC_RC32MEN_bm;                  // włączenie oscylatora 32MHz
        while(!(OSC.STATUS & OSC_RC32MRDY_bm));     // czekanie na ustabilizowanie się generatora
        CPU_CCP = CCP_IOREG_gc;                     // odblokowanie zmiany źródła sygnału zegarowego
        CLK.CTRL = CLK_SCLKSEL_RC32M_gc;            // zmiana źródła sygnału zegarowego na RC 32MHz
        LcdClear();                                 // czyszczenie wyświetlacza
        Lcd(„RC 32MHz”);                            // komunikat o uruchomieniu generatora
}

void OscXtal(void) {

                                                    // konfiguracja generatora kwarcowego
        OSC.XOSCCTRL = OSC_FRQRANGE_12TO16_gc |     // wybór kwarcu od 12 do 16 MHZ
        OSC_XOSCSEL_XTAL_16KCLK_gc;                 // czas na uruchomienie generatora
        OSC.CTRL = OSC_XOSCEN_bm;                   // uruchomienie generatora kwarcowego

                                                    // czekanie na ustabilizowanie się generatora
        for(uint8_t i=0; i<255; i++) {
                if(OSC.STATUS & OSC_XOSCRDY_bm) {
                        CPU_CCP = CCP_IOREG_gc;             // odblokowanie zmiany źródła sygnału zegarowego
                        CLK.CTRL = CLK_SCLKSEL_XOSC_gc;     // wybór źródła sygnału zegarowego na XTAL 16MHz
                        LcdClear();                         // czyszczenie wyświetlacza
                        Lcd(„XTAL”);                        // komunikat o uruchomieniu generatora

                                                            // układ nadzorujący kwarc
                        CPU_CCP = CCP_IOREG_gc;             // odblokowanie modyfikacji ważnych rejestrów
                        OSC.XOSCFAIL = OSC_XOSCFDEN_bm;     // włączenie układu detekcji błędu sygnału
                                                            // zegarowego
                        return;                             // wyjście z funkcji jeśli generator się
                                                            // uruchomił
                }
                _delay_us(10);
        }

                                                    // komunikat w przypadku braku uruchomienia
                                                    // generatora kwarcowego
        LcdClear();
        Lcd(„Brak XTAL”);
}

void OscPLL(uint8_t pllfactor) {

                                                    // uruchomienie generatora 2MHz i ustawienie go
                                                    // jako źródła zegara
        OSC.CTRL = OSC_RC2MEN_bm;                   // włączenie oscylatora 2MHz
        while(!(OSC.STATUS & OSC_RC2MRDY_bm));      // czekanie na ustabilizowanie się generatora
        CPU_CCP = CCP_IOREG_gc;                     // odblokowanie zmiany źródła sygnału zegarowego
        CLK.CTRL = CLK_SCLKSEL_RC2M_gc;             // zmiana źródła sygnału zegarowego na RC 2MHz

                                                    // wyłączenie PLL
        OSC.CTRL                &= ~OSC_PLLEN_bm;

                                                    // konfiguracja PLL
        OSC.PLLCTRL = OSC_PLLSRC_RC2M_gc |          // wybór RC 2MHz jako źródło sygnału dla PLL
        pllfactor;                                  // mnożnik częstotliwości (od 1 do 31)
                                                    // uruchomienie PLL
        OSC.CTRL = OSC_PLLEN_bm;                    // włączenie układu PLL

                                                    // czekanie na ustabilizowanie się generatora
        while(!(OSC.STATUS & OSC_PLLRDY_bm));

                                                    // przełączenie źródła sygnału zegarowego
        CPU_CCP = CCP_IOREG_gc;                     // odblokowanie zmiany źródła sygnału zegarowego
        CLK.CTRL = CLK_SCLKSEL_PLL_gc;              // wybór źródła sygnału zegarowego PLL

                                                    // układ nadzorujący PLL
        CPU_CCP = CCP_IOREG_gc;                     // odblokowanie modyfikacji ważnych rejestrów
        OSC.XOSCFAIL = OSC_PLLFDEN_bm;              // włączenie układu detekcji błędu sygnału
                                                    // zegarowego

                                                    // wyświetlenie komunikatu
        LcdClear();
        Lcd(„PLL „);
        LcdDec(pllfactor*2);                        // *2 bo generator RC ma 2MHz
        Lcd(„MHz”);
}

int main(void) {

                                                    // zmienna
        uint8_t pll = 4;

                                                    // diody
        PORTE.DIR = PIN0_bm;                        // dioda LED

                                                    // przyciski
```

Listing 1. c.d.

```
        PORTA.DIRCLR = PIN0_bm;                         // przycisk - RC 2MHz
        PORTA.PIN0CTRL = PORT_OPC_PULLUP_gc;            // podciągnięcie do zasilania
        PORTE.DIRCLR = PIN5_bm;                         // przycisk FLIP - RC 32MHz
        PORTE.PIN5CTRL = PORT_OPC_PULLUP_gc;            // podciągnięcie do zasilania
        PORTE.DIRCLR = PIN6_bm;                         // przycisk - XTAL
        PORTE.PIN6CTRL = PORT_OPC_PULLUP_gc;            // podciągnięcie do zasilania
        PORTF.DIRCLR = PIN4_bm;                         // przycisk - PLL
        PORTF.PIN4CTRL = PORT_OPC_PULLUP_gc;            // podciągnięcie do zasilania

                                                        // wyświetlacz LCD
        LcdInit();

                                                        // komunikat o źródle sygnału zegarowego
        LcdClear();
        Lcd(„RC 2MHz”);

                                                        // włączenie przerwań
        sei();

        while(1) {
                PORTE.OUTTGL = PIN0_bm;
                _delay_ms(50);

                if(!(PORTA.IN & PIN0_bm)) Osc2MHz();
                if(!(PORTE.IN & PIN5_bm)) Osc32MHz();
                if(!(PORTE.IN & PIN6_bm)) OscXtal();
                if(!(PORTF.IN & PIN4_bm)) {
                        pll++;                          // zwiększ zmienną pll
                        if(pll > 31) pll = 1;           // jeśli pll większe od 31 to ustaw na 1
                        OscPLL(pll);                    // funkcja konfigurująca PLL
                }
        }
}

ISR(OSC_OSCF_vect) {                                    // przerwanie w razie awarii oscylatora
        OSC.XOSCFAIL            |=      OSC_XOSCFDIF_bm; // kasowanie flagi przerwania
        LcdClear();
        Lcd(„Awaria!”);
}
```

tliwości od 0,4 MHz do 16 MHz. Chcąc uzyskać częstotliwość taktowania procesora większą od 16 MHz, musimy zastosować układ PLL, który zostanie omówiony później.

Konfiguracja generatora kwarcowego jest trochę bardziej skomplikowana niż generatory omawiane dotychczas, aczkolwiek nie jest to problem, z którym sobie nie poradzimy. Za nastawy generatora kwarcowego jest odpowiedzialny rejestr OSC.XOSCCTRL. Zwróćmy uwagę na fragment dokumentacji na **rysunku 6**.



Rysunek 6. Fragment dokumentacji rejestru XOSCCTRL

Pierwszy z parametrów, FRQRANGE, określa przedział, wewnątrz którego musi znajdywać się częstotliwość kwarcu. X32KLPM służy do uruchamiania trybu oszczędzania energii dla kwarcu 32 kHz. XOSCPWR zwiększa prąd w obwodzie rezonatora, co w większości przypadków nie jest potrzebne. XOSCSEL wyznacza m. in. czas potrzebny do ustabilizowania się generatora.

W tym przykładzie zastosujemy kwarc o częstotliwości 16 MHz. Jego typowy czas startu wynosi 16 tys. cykli zegarowych. Aby poprawnie skonfigurować generator, wystarczy umieścić w programie następujące polecenie:

```
OSC.XOSCCTRL = OSC_FRQRANGE_12TO16_gc|  // wybór kwarcu od 12 do 16 MHZ
OSC_XOSCSEL_XTAL_16KCLK_gc;             // czas na uruchomienie generatora
```

Następnie, uruchamiamy generator kwarcowy.

```
OSC.CTRL = OSC_XOSCEN_bm;  // uruchomienie generatora kwarcowego
```

Powinniśmy teraz poczekać, aż częstotliwość wyjściowa generatora ustabilizuje się. Jednak nie możemy czekać w pustej pętli na ustawienie się odpowiedniego bitu w rejestrze statusowym, bo jeśli generator się nie uruchomi, to procesor będzie czekał w nieskończoność. Dlatego musimy wprowadzić pętlę, która odliczy przykładowo 255 cykli i sprawdzi status 255 razy – jeśli w tym czasie generator nie ustabilizuje się, program uzna, że z jakiś powodów generator nie daje się uruchomić (np. uszkodzony rezonator kwarcowy) i będzie mógł powiadomić użytkownika o błędzie (**listing 4**).

Mikrokontrolery Xmega wyposażono w szereg rozwiązań zwiększających bezpieczeństwo i stabilność pracy. Jednym z nich jest układ nadzorujący pracę generatora kwarcowego. Jeśli zostanie wykryta nieprawidłowość

w pracy generatora, układ zgłosi przerwanie niemaskowalne (wektor OSC_OSCF_vect) i samoczynnie przełączy źródło sygnału na wbudowany generator RC 2 MHz. Starsze mikrokontrolery ATtiny i ATmega w takiej sytuacji zawieszały się, a w skrajnych wypadkach nawet nie mogły nawiązać połączenia z programatorem.

Zanim zostanie uruchomiony układ detekcji błędu generatora kwarcowego musimy wpisać wartość CCP_IOREG_gc do rejestru CPU_CCP, aby zezwolić na modyfikowanie rejestrów kluczowych dla funkcjonowania procesora. Następnie, do rejestru XOSCFAIL wpisujemy wartość OSC_XOSCFDEN_bm i układ detekcji zostanie uruchomiony. Trzeba zaznaczyć, że tego układu nie można wyłączyć bez restartu procesora.

```
//odblokowanie możliwości modyfikowania
ważnych rejestrów
CPU_CCP = CCP_IOREG_gc;
//włączenie układu detekcji błędu
OSC.XOSCFAIL = OSC_XOSCFDEN_bm;
```

W razie stwierdzenia usterki generatora, zostanie wygenerowane przerwanie OSC_OSCF_vect, a generator RC 2 MHz włączy się automatycznie.

```
ISR(OSC_OSCF_vect) {
  OSC.XOSCFAIL |= OSC_XOSCFDIF_bm;  //
kasowanie flagi przerwania
  LcdClear();
  Lcd(„Awaria!”);
}
```

Generator z pętlą PLL

PLL, czyli pętla synchronizacji fazy, jest przeciwieństwem preskalera i służy do zwiększania częstotliwości sygnału. W mikrokontrolerach Xmega to rozwiązanie jest dostępne we wszystkich modelach, nawet w tych najtańszych.

Generator PLL w Xmega umożliwia podwyższenie częstotliwości przebiegu wybranego generatora do 31 razy. Do wyboru są następujące generatory:

- wbudowany RC 2 MHz,
- wbudowany RC 32 MHz, ale wstępnie podzielony przez 4, czyli 8 MHz,
- kwarcowy 0,4…16 MHz,
- zewnętrzny sygnał zegarowy.

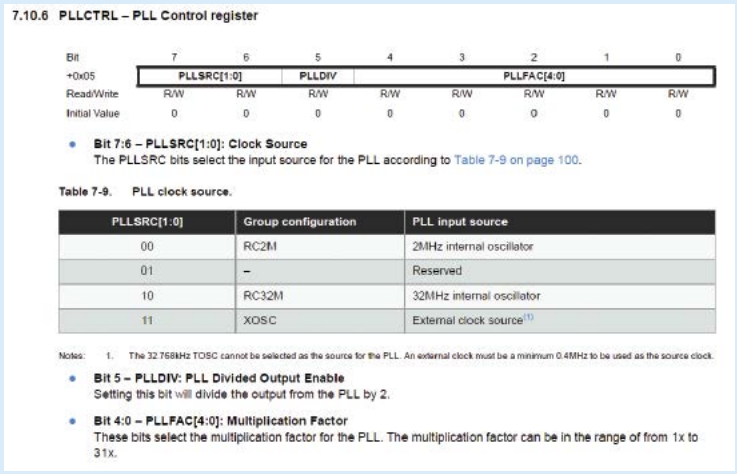
Układ PLL nie może współpracować z generatorami 32 kHz.

Niewłaściwie konfigurując układ PLL można mocno przekroczyć dopuszczalna częstotliwość taktowania mikrokontrolera, co może prowadzić do jego niestabilnej pracy. Rdzeń mikrokontrolera może pracować taktowany przebiegiem o częstotliwości do 32 MHz. Częstotliwość na wyjściu układu PLL nie powinna być niższa niż 10 MHz, ani wyższa niż 200 MHz. W razie potrzeby można zastosować preskalery (widoczne na rys. 1), aby zmniejszyć częstotliwość przebiegu.

Napiszemy program, który pozwala zmienić konfigurację układu PLL podczas pracy procesora. Wciskając przycisk dołączony do wyprowadzenia F4 na płytce X3-DIL64, będziemy zwiększali mnożnik PLL w zakresie 1…31. Jako źródło sygnału zastosujemy generator 2 MHz, a częstotliwości uzyskane dzięki PLL będą sięgały nawet 62 MHz, co daleko przekracza dopuszczalny limit. Zmiany sygnału zegarowego będziemy obserwować dzięki mrugającej diodzie, a dodatkową informacją będzie wyświetlenie częstotliwości na wyświetlaczu LCD.

Prześledźmy, co dzieje się w funkcji OscPLL. Pierwszym krokiem jest uruchomienie generatora, który będzie źródłem sygnału dla PLL i ustawienie go jako źródła. Trzeba w tym momencie wyraźnie zaznaczyć, że nie można zmieniać konfiguracji układu PLL podczas, gdy jest on uruchomiony, a tym bardziej kiedy jest źródłem sygnału zegarowego (**listing 5**).

Następnie, możemy przystąpić do konfiguracji układu PLL. Jednak jeśli jest on już włączony, to koniecznie musimy go najpierw wyłączyć. W przeciwnym razie próba zmiany konfiguracji będzie nieskuteczna. Kluczowy



7.10.6 PLLCTRL – PLL Control register

Bit	7	6	5	4	3	2	1	0
+0x05	PLLSRC[1:0]		PLLDIV	PLLFAC[4:0]				
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:6 – PLLSRC[1:0]: Clock Source**
  The PLLSRC bits select the input source for the PLL according to Table 7-9 on page 100.

Table 7-9. PLL clock source.

PLLSRC[1:0]	Group configuration	PLL input source
00	RC2M	2MHz internal oscillator
01	–	Reserved
10	RC32M	32MHz internal oscillator
11	XOSC	External clock source(1)

Notes: 1. The 32.768kHz TOSC cannot be selected as the source for the PLL. An external clock must be a minimum 0.4MHz to be used as the source clock.

- **Bit 5 – PLLDIV: PLL Divided Output Enable**
  Setting this bit will divide the output from the PLL by 2.
- **Bit 4:0 – PLLFAC[4:0]: Multiplication Factor**
  These bits select the multiplication factor for the PLL. The multiplication factor can be in the range of from 1x to 31x.

Rysunek 7. Fragment dokumentacji rejestru PLLCTRL

Listing 5.

```
void OscPLL(uint8_t pllfactor) {

                                              // uruchomienie generatora 2MHz i ustawienie go
                                              // jako źródła zegara
        OSC.CTRL = OSC_RC2MEN_bm;             // włączenie oscylatora 2MHz
        while(!(OSC.STATUS & OSC_RC2MRDY_bm)); // czekanie na ustabilizowanie się generatora
        CPU_CCP = CCP_IOREG_gc;               // odblokowanie zmiany źródła sygnału zegarowego
        CLK.CTRL = CLK_SCLKSEL_RC2M_gc;       // zmiana źródła sygnału zegarowego na RC 2MHz
```

Listing 6.

```
                                              // wyłączenie PLL
        OSC.CTRL                  &= ~OSC_PLLEN_bm;

                                              // konfiguracja PLL
        OSC.PLLCTRL = OSC_PLLSRC_RC2M_gc |    // wybór RC 2MHz jako źródło sygnału dla PLL
                                  pllfactor;  // mnożnik częstotliwości (od 1 do 31)
                                              // uruchomienie PLL
        OSC.CTRL = OSC_PLLEN_bm;              // włączenie układu PLL
```

w tym fragmencie jest rejestr OSC.PLLCTRL, którego opis przedstawiono na **rysunku 7**. Musimy wybrać źródło sygnału, współczynnik mnożący (zmienna *pllfactor* jest argumentem funkcji *OscPLL*), a opcjonalnie możemy częstotliwość sygnału wyjściowego podzielić przez dwa (**listing 6**).

Podobnie jak w przypadku innych generatorów, poczekać musimy aż sygnał zegarowy się ustabilizuje, poprzez sprawdzanie czy już został ustawiony odpowiedni bit w rejestrze statusowym. Dopiero wtedy możemy przełączyć źródło sygnału taktującego mikrokontroler:

```
//oczekiwanie na ustabilizowanie się
generatora
while(!(OSC.STATUS & OSC_PLLRDY_bm));
//odblokowanie możliwości zmiany źródła
sygnału zegarowego
CPU_CCP = CCP_IOREG_gc;
//wybór źródła sygnału zegarowego PLL
CLK.CTRL = CLK_SCLKSEL_PLL_gc;
```

Układ PLL może stracić synchronizację, jeśli sygnał zegarowy będzie zbyt wolny, zbyt szybki lub niestabilny. Dlatego mikrokontrolery Xmega mają możliwość monitorowania układu PLL, podobnie jak generatora kwarcowego. W razie stwierdzenia nieprawidłowości, jako źródło przebiegu taktującego zostanie wybrany generator 2 MHz oraz zostanie zgłoszone przerwanie OSC_OSCF_vect.

```
// układ nadzorujący PLL
//odblokowanie możliwości modyfikowania
ważnych rejestrów
CPU_CCP = CCP_IOREG_gc;
//włączenie układu detekcji błędu sygnału
zegarowego
OSC.XOSCFAIL = OSC_PLLFDEN_bm;
```

Podczas ćwiczeń przekroczyliśmy częstotliwość taktowania rdzenia mikrokontrolera prawie dwukrotnie. Zgodnie z danymi firmy Atmel, układ powinien być taktowany w zakresie od 10 MHz (minimalna częstotliwość wyjściowa PLL) do 32 MHz (maksymalna częstotliwość rdzenia). Jestem ciekaw, czy Czytelnicy zauważyli jakieś nieprawidłowości w działaniu mikrokontrolera poza tym zakresem. W moim przypadku wszystko działało bez zarzutu. Mimo tego, w normalnych zastosowaniach nigdy nie należy przekraczać dopuszczalnych zakresów podanych przez producenta układu. Na **listingu 7** pokazano program demonstrujący możliwości układu generowania i dystrybucji sygnałów zegarowych.

Na zakończenie

W ramach pojedynczego artykułu nie sposób wyczerpać tematu wszystkich zagadnień związanych z sygnałów zegarowych w mikrokontrolerach Xmega. Oprócz opisanych wyżej zagadnień, użytkownik Xmegi ma do dyspozycji następujące bloki i funkcje:

- układ synchronizujący DFLL,
- zegar czasu rzeczywistego RTC,
- generatory energooszczędne,
- generator sygnału zegarowego do USB,
- kalibrację generatorów.

Te możliwości zostały opisane np. w książce Tomasza Francuza „AVR. Praktyczne projekty” oraz w dokumentacji technicznej.

Dominik Leon Bieczyński
http://leon-instruments.blogspot.com