Mikrokontrolery Precision32 (2)

Tworzenie aplikacji krok po kroku

Jako forma podsumowania całego cyklu artykułów o mikrokontrolerach Precision32 firmy Silicon Labs, w ostatniej części prezentujemy krok po kroku jak stworzyć kompletną aplikację. Zaprezentowane przykłady (sterownik LED o konfigurowalnej jasności świecenia oraz interfejs 4...20 mA) zostały utworzone w oparciu o zintegrowany w mikrokontrolerze przetwornik C/A.

Mikrokontrolery z rodziny Precision32 zostały wyposażone w pokaźny zestaw peryferiów analogowych. Są to dwa przetworniki A/C, dwa komparatory analogowe, dwa przetworniki prąd-napięcie oraz kontroler mierzący pojemność, przeznaczony do obsługi dla interfejsów dotykowych. Ponadto, do grupy zasobów analogowych należą dwa przetworniki C/A, które zamieniają wartość liczbową na odpowiadającą jej wartość natężenia prądu.

Tabela 1. Podstawowe parametry przetwornika C/A IDAC0 i IDAC1

Parametr	Wartość
Rozdzielczość	10 bitów
Zakres wartości prądu	2 mA (krok 2 μA), 1 mA (krok 1 μA), 0.5 mA (krok 500 nA)
Offset	250 nA
Częstotliwość odświeżania wartości wyjściowej	Ponad 600 ksps

Przetwornik C/A w mikrokontrolerach Precision32

Przetworniki C/A otrzymały oznaczenia IDAC0 oraz IDAC1. Schemat blokowy pojedynczego przetwornika IDAC został przedstawiony na **rysunku 1**.

Zgodnie z tym schematem przetwornik IDAC składa się z modułu konwersji C/A, bloku sterującego, kolejki (bufora) FIFO, bloku sterującego kolejką FIFO, wejścia danych oraz bloku wyzwalającego. Dodatkowo wyjście przetwornika może zostać wewnątrz mikrokontrolera połączone przez rezystor do masy, co w efekcie da na wyjściu przetwornika wartość napięcia.

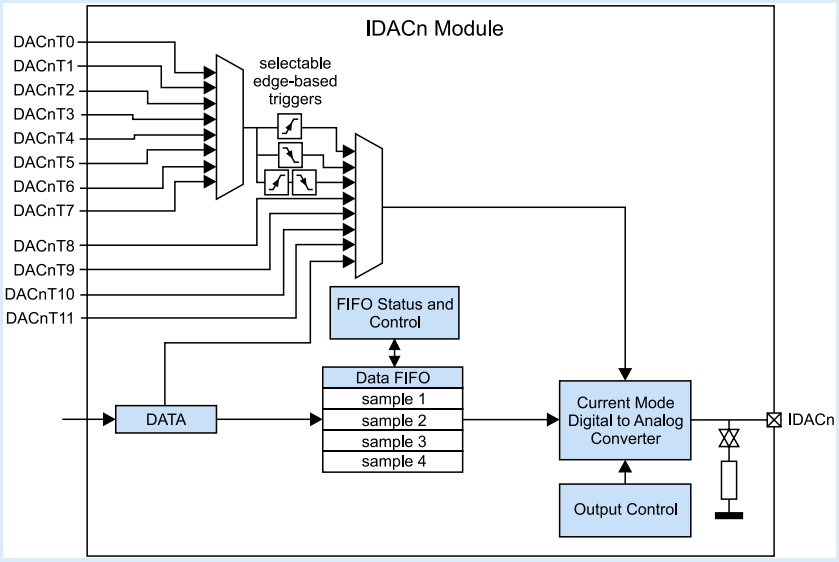
Kolejka FIFO składa się z 32 bitów. Bufor ten może zostać podzielony na cztery 8-bitowe części, dzięki czemu może przechowywać cztery różne wartości liczbowe. Bufor może również zostać podzielony na dwie 16-bitowe części, które mogą służyć do przechowywania jednej lub dwóch 10-bitowych wartości liczbowych.

Istnieją trzy mechanizmy wyzwalania, pozwalające zaktualizować wartość wyjściową przetwornika. Pierwszy mechanizm to żądanie bezpośrednie (zmiana wartości wyjściowej przetwornika następuje natychmiast po zapisie nowej wartości liczbowej do bufora FIFO). Dwa kolejne mechanizmy pozwalają uzależnić sterowanie przetwornika od innych peryferiów: odpowiednio liczników lub portów. W pierwszym przypadku zmiana wartości wyjściowej przetwornika następuje po przekroczeniu wartości licznika („Timer 0 Low overflow", „Timer 0 High overflow", „Timer 1 Low overflow", „Timer 2 High overflow"). W drugim przypadku przetwornik może zaktualizować wartość wyjściową po wykryciu zbocza (narastającego, opadającego lub narastającego bądź opadającego) na jednym z ośmiu portów. Podstawowe parametry techniczne przetwornika podano w **tabeli 1**.

Dostępne są cztery tryby pracy przetwornika IDAC: „On-Demand", „Periodic FIFO Wrap", „Periodic FIFO-Only" oraz „Periodic with DMA".

„On-Demand" to tryb bezpośredniej kontroli, w którym wartość wyjściowa przetwornika jest zmieniana na żądanie wynikające z działania programu w losowych momentach czasu. W trybie tym wykorzystywana jest tylko jedna 10-bitowa wartość liczbowa z bufora FIFO.

Pozostałe trzy tryby są przeznaczone do okresowych zmian wartości wyjściowej przetwornika. W trybie „Periodic FIFO-Only" przetwornik aktualizuje dane pobierając je kolejno z kolejki FIFO. Mikrokontroler w tym trybie musi



Rysunek 1. Schemat blokowy pojedynczego przetwornika IDAC

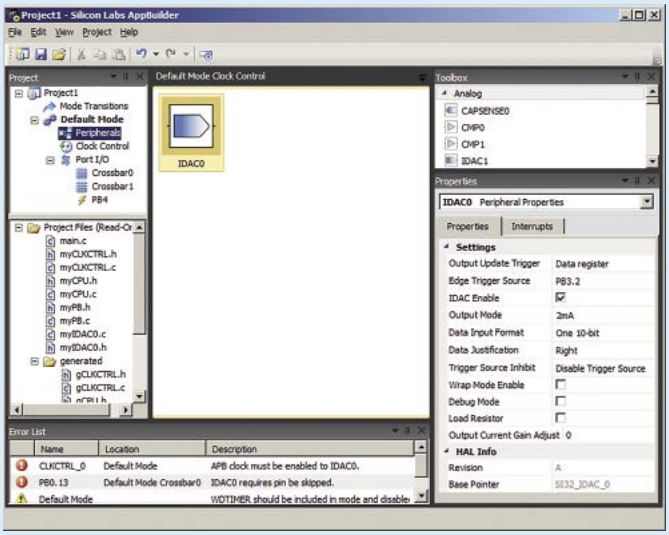
Listing 1. Zawartość pliku „gIDAC0.c". Wygenerowany przez program AppBuilder kod konfigurujący przetwornik IDAC0

```
#include „gIDAC0.h"
// Include peripheral access modules used in this file
#include <SI32_IDAC_A_Type.h>
#include <si32_device.h>
//============================
// Configuration Functions
//============================
void IDAC0_enter_default_mode_from_reset(void)
{
  SI32_IDAC_A_set_output_update_trigger(SI32_IDAC_0,SI32_
IDAC_A_CONTROL_OUPDT_DACNT15_VALUE);
  SI32_IDAC_A_enable_module(SI32_IDAC_0);
  SI32_IDAC_A_disable_trigger(SI32_IDAC_0);
  SI32_IDAC_A_select_output_fullscale_2ma(SI32_IDAC_0);
}
```

Listing 2. Plik „main.c" z wywołaniem funkcji ustawiającej wartość wyjściową przetwornika IDAC0

```
#include „gModes.h"
#include <SI32_PBCFG_A_Type.h>
#include <SI32_PBSTD_A_Type.h>
#include <si32_device.h>

int idac0_output_value = 500;
//====================
// My application
//====================
int main(void)
{
  // Enter the default operating mode for this
application
  enter_default_mode_from_reset();
  while (1)
  {
    SI32_IDAC_A_write_data(SI32_IDAC_0, idac0_output_
value);
  }
}
```



Rysunek 2. Program AppBuilder: etap dodawania przetwornika IDAC0 do projektu i jego konfigurowania

dostarczać kolejne dane do bufora. Tryb „Periodic FIFO Wrap" różni się od „Periodic FIFO-Only" tym, że mikrokontroler nie musi dostarczać kolejnych próbek do kolejki FIFO, gdyż przetwornik wykorzystuje tylko cztery dane, używając ich w pętli. Tryb „Periodic with DMA" pozwala odciążyć mikrokontroler, gdyż za dostarczanie danych do kolejki FIFO odpowiada w tym trybie moduł DMA.

Program sterujący przetwornikiem C/A

Program sterujący przetwornikiem IDAC przygotowany zostanie przy użyciu dostępnych narzędzi programistycznych: programu komputerowego Silicon Labs AppBuilder, środowiska Keil MDK-ARM. Jako platformę sprzętową zastosowano płytkę uruchomieniową, której projekt został przedstawiony w EP8/2013. W pierwszej kolejności użyty zostanie program AppBuilder. Pozwala on za pomocą graficznego interfejsu użytkownika wygenerować kod konfiguracyjny dla mikrokontrolera. Szczegółowy opis użycia programu AppBuilder został zamieszczony w EP8/2013.

Po stworzeniu w programie AppBuilder nowego projektu należy dodać do niego blok reprezentujący przetwornik IDAC0, a następnie ustawić parametry jego pracy (**rysunek 2**). Na potrzeby zaprezentowanych przykładowych aplikacji należy zmienić zakres wartości przetwornika z 0,5 do 2 mA (wybór „2mA" dla pola „Output Mode") oraz sposób jego wyzwalania z wyzwalania licznikiem na wyzwalanie przez zapis do bufora FIFO (wybór „Data register" dla pola „Output Update Trigger"). Dla wygody warto też ustawić opcję odpowiedzialną za włączenie przetwornika IDAC0 automatycznie w momencie rozpoczęcia pracy mikrokontrolera (zaznaczone pole „IDAC Enable").

AppBuilder w oknie błędów wyświetli dwa komunikaty: pierwszy o konieczności włączenia sygnału zegarowego dedykowanego przetwornikowi IDAC0, drugi o konieczności zmiany ustawień dla portu wejścia/wyjścia będącego wyjściem przetwornika IDAC0 (wyprowadzenie PB0.13). Zrealizowanie obu tych czynności przedstawiono odpowiednio na **rysunku 3** i **rysunku 4**. Proces konfiguracji projektu w programie AppBuilder jest zakończony. Można w tym momencie utworzyć na jego podstawie projekt programistyczny.

Stworzony z wykorzystaniem programu AppBuilder projekt programistyczny można otworzyć za pomocą środowiska Keil MDK-ARM. Wygenerowany kod konfiguracyjny dla przetwornika IDAC0 znajduje się w katalogu „generated" w pliku „gIDAC0.c". Przedstawiono go na **listingu 1**. Funkcją odpowiedzialną za skonfigurowanie przetwornika jest *IDAC0_enter_default_mode_from_reset()*. Wewnątrz niej program AppBuilder umieścił wywołania funkcji, z których każda odpowiada za jeden parametr pracy przetwornika. Funkcje te pochodzą z zestawu przygotowanych przez firmę Silicon Labs bibliotek dla peryferiów o nazwie HAL. Funkcja *IDAC0_enter_default_mode_from_reset()* jest wykonywana automatycznie przez mikrokontroler tuż po jego uruchomieniu.

Mając zaimplementowany mechanizm konfiguracji parametrów pracy przetwornika IDAC0 można uzupełnić kod źródłowy o fragment odpowiedzialny za ustawianie wartości prądu na jego wyjściu. Odpowiada za to funkcja *SI32_IDAC_A_write_data()*. Można ją wywołać np. w funkcji *main()* w pliku „main.c" (**listing 2**). Drugi argument funkcji *SI32_IDAC_A_write_data()*, tu nazwany *idac0_output_value*, to wartość liczbowa, która zostanie zamieniona na wartość natężenia prądu. Jako że dopuszczalna wartość maksymalna tej zmiennej, a więc 1023 (gdyż przetwornik jest 10-bitowy) oznacza wartość natężenia prądu równą 2 mA, to wynikająca z zasady proporcji zależność między wartością idac0_output_value i wartością prądu na wyjściu przetwornika jest następująca:

$$I = \frac{\text{idac0\_output\_value} \cdot 2\ mA}{1023}$$

Zgodnie z powyższym wzorem, ustawienie wartości zmiennej idac0_output_value na 500, tak jak w pokazanym w **listingu 2** przykładzie, odpowiada wartości prądu 0.9775 mA.

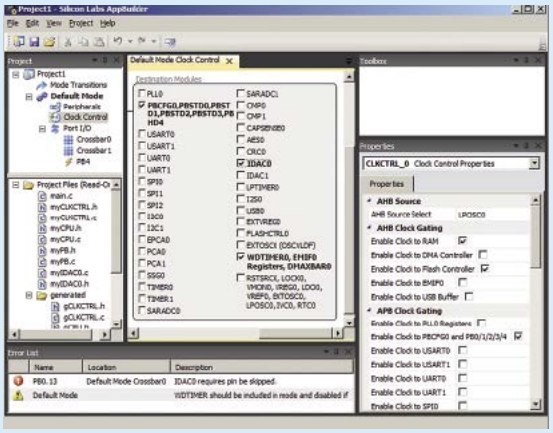
Przykładowe aplikacje

Przetwornik C/A to zasób, który może być niezwykle przydatny w różnych aplikacjach wymagających

generowania sygnałów analogowych. Jako przykład zaprezentowane zostaną dwie takie aplikacje.

Pierwszą z nich jest sterownik LED o konfigurowalnej jasności świecenia. Jako że przetwornik IDAC0 charakteryzuje się zakresem prądowym do 2 mA, więc nie jest w stanie sterować bezpośrednio jasnością diody LED w pełnym zakresie, gdyż jest to wartość zbyt mała. W celu zwiększenia wydajności prądowej (do około 8 mA) zastosowano obwód sterujący z tranzystorem bipolarnym typu NPN. Schemat elektryczny drivera przedstawiono na **rysunku 5**. W rozwiązaniu tym wyjście przetwornika IDAC0 (wyprowadzenie PB0.13) jest dołączone przez rezystor R3 do bazy tranzystora T1, szeregowo połączone diody LED1, LED2 i rezystor ograniczający prąd R2 wpięte są między kolektor tranzystora T1 i napięcie o potencjale 5 V, natomiast emiter tranzystora T1 jest połączony z masą. Regulując prąd bazy tranzystora T1 (I_B) za pomocą prądu przetwornika IDAC0 (I_{WY}) można uzyskać odpowiednio wzmocniony prąd kolektora ($I_C = I_B \times \beta$) tranzystora T1, a tym samym różne jasności świecenia diod LED1 i LED2. Rezystory R1 i R3 służą do pochylenia krzywej przedstawiającej zależność płynącego przez diody LED1 i LED2 prądu I_C od prądu I_{WY} (**rysunek 6**). Zdjęcie działającego sterownika przedstawiono na **rysunku 7**.

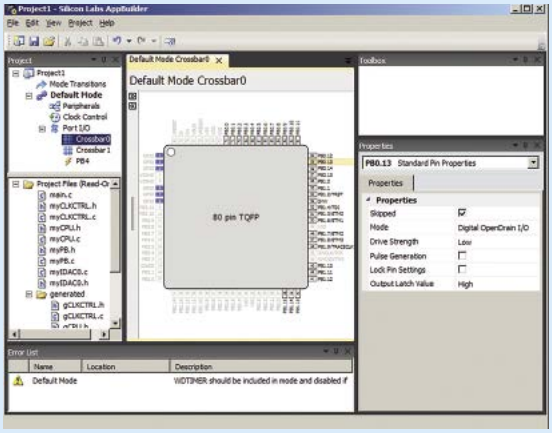
Drugim zaprezentowanym przykładem aplikacji jest interfejs 4…20 mA stosowany w urządzeniach elektronicznych do przesyłania informacji. Interfejs ten ma postać pętli prądowej. Oznacza to, że przesyłana informacja reprezentowana jest przez sygnał prądowy o określonym natężeniu (od 4 do 20 mA). Jest to standard stosowany



Rysunek 3. Program AppBuilder: włączenie sygnału zegarowego dla przetwornika IDAC0

w różnych przemysłowych systemach kontrolno-pomiarowych, gdyż sygnał prądowy w porównaniu z sygnałem napięciowym cechuje się dużą odpornością na zaburzenia zewnętrzne i pozwala na transmisję danych na relatywnie większe odległości. Przykładem zastosowania interfejsu 4…20 mA są czujniki, które za jego pomocą przesyłają wartość mierzonego parametru – temperatury, wilgotności, ciśnienia itp.

Interfejs 4…20 mA nie jest modułem integrowanym w mikrokontrolerach, dlatego aby móc korzystać z jego funkcjonalności jest konieczne zastosowanie dodatkowych elementów. Najprostszym rozwiązaniem jest użycie jednego z dostępnych, dedykowanych układów. Może



Rysunek 4. Program AppBuilder: etap skonfigurowania portu dla przetwornika IDAC0

to być np. AD421 firmy Analog Devices lub MAX5661 firmy Maxim Integrated. Są to przetworniki o wejściu cyfrowym i wyjściu analogowym prądowym od 4 do 20 mA.

W przypadku mikrokontrolerów Precision32 taką samą funkcjonalność osiągnąć można używając wbudowanego przetwornika C/A, wzmacniacza operacyjnego i kilku komponentów biernych. Jest to rozwiązanie tańsze w porównaniu z wykorzystaniem układu specjalizowanego.

Schemat elektryczny interfejsu 4…20 mA wykorzystującego mikrokontroler Precision32 zaprezentowano na **rysunku 8**. W zaproponowanym rozwiązaniu można wyodrębnić dwa bloki funkcjonalne. Pierwszy odpowiada za konwersję prądu wyjściowego przetwornika IDAC0 (I_1) na napięcie (U_1). Proces ten realizowany jest przez rezystor R_1 wpięty pomiędzy wyjście przetwornika IDAC0 i masę. Uzyskaną w ten sposób zależność napięcia od prądu można obliczyć ze wzoru:

$$U_1 = R_1 \cdot I_1$$

Drugi blok odpowiada za konwersję wartości tak otrzymanego napięcia na wartość prądu. Blokiem tym jest wzmacniacz operacyjny IC1 razem z rezystorami R1 i R_L. Obwód ten pracuje w konfiguracji przetwornika napięcia na prąd. Płynący przez rezystor obciążenia R_L prąd I_2 na wyjściu wzmacniacza obliczyć można ze wzoru:

$$I_2 = \frac{U_1}{R_2}$$

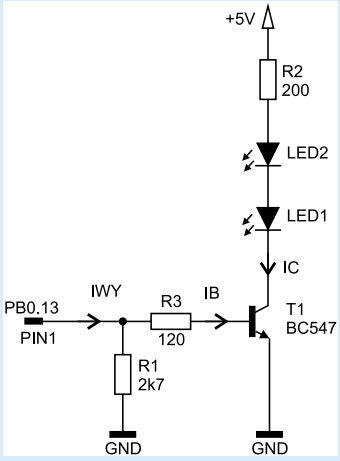
Podstawiając wzór na napięcie U_1 do wzoru na prąd I_2 otrzymujemy zależność między prądem wyjściowym przetwornika IDAC0 (I_1) a prądem wyjściowym wzmacniacza I_2:

$$I_2 = \frac{R_1}{R_2} \cdot I_1$$

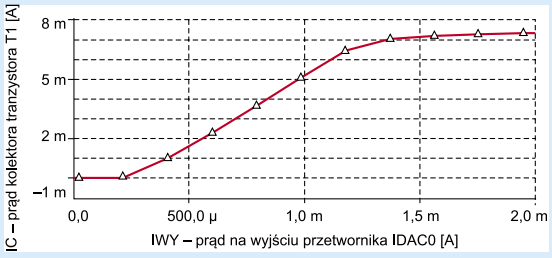
Maksymalny prąd wyjściowy przetwornika IDAC0 to 2 mA, dlatego w celu uzyskania na wyjściu wzmacniacza prądu o wartości do 20 mA dobrano odpowiednio wartości rezystorów: R_1 = 1.1 kΩ, R_2 = 50 Ω. Dla tak przyjętych wartości współczynnik wzmocnienia określony jako stosunek wartości R_1 do wartości R_2 wynosi 22. Dzięki temu minimalną i maksymalną wartość prądu płynącego przez obciążenie R_L (50 Ω), a więc 4 i 20 mA, uzyskamy dla prądu przetwornika IDAC0 odpowiednio 0,182 mA i 0,909 mA. Zdjęcie działającego interfejsu 4…20 mA pokazano na **fotografii 9**.

Autor składa podziękowania Panu Pawłowi Bardowskiemu za pomoc w realizacji artykułu.

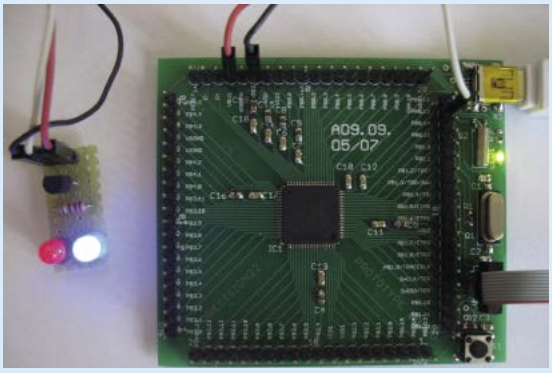
Szymon Panecki
Wydział Elektroniki
Politechnika Wrocławska
szymon.panecki@pwr.wroc.pl



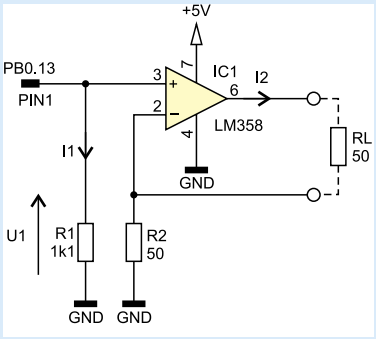
Rysunek 5. Schemat elektryczny sterownika diod LED



Rysunek 6. Krzywa przedstawiająca zależność prądu kolektora (I_C) tranzystora T1 od prądu wyjściowego przetwornika IDAC0 (I_{WY})

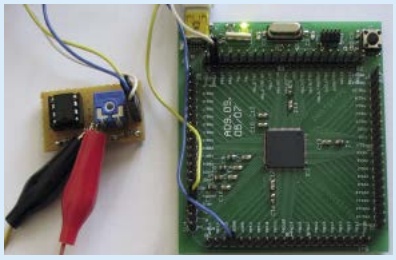


Fotografia 7. Zdjęcie wykonanego sterownika diod LED



Rysunek 8. Schemat elektryczny interfejsu 4-20 mA

Literatura
[1] www.silabs.com SIM3U1xx/SIM3C1xx datasheet
[2] www.silabs.com SIM3U1xx/SIM3C1xx reference manual



Fotografia 9. Zdjęcie wykonanego interfejsu 4…20 mA