Mikrokontrolery z rdzeniami Cortex-M - STM32F401 w praktyce

Grzegorz Mazur

Politechnika Warszawska

Instytut Informatyki

Co to jest Cortex?

- Nowa generacja procesorów (rdzeni) ARM
- Zestaw instrukcji Thumb 2
  - Instrukcje 16- i 32- bitowe – lepsza gęstość kodu niż w klasycznym ARM
- Nowe mechanizmy obsługi wyjątków
  - Nowatorski system wyjątków o wysokiej wydajności, eliminujący zbędne czynności procesora i łatwy dla programisty

Rodziny Cortex

- Cortex-M – Microcontroller
  - Do zastosowania w mikrokontrolerach, ograniczona ochrona i mechanizmy systemowe, niewielka wydajność, mały pobór mocy
- Cortex-R – Real-time
  - Podbne do M, wyższa wydajność – zastępują procesory sygnałowe
- Cortex-A – Application
  - Kompletne mechanizmy systemowe, do zastosowania w komputerach ogólnego przeznaczenia (smartfony, tablety)

Wersje Cortex M

- M0 – 0.8 DMIPS/MHz, minimalistyczny, zastępuje 8- i 16-bitowe
- M0+ - 0.85 DMIPS/MHz, szybszy i mniejszy od M0, lepsza ochrona i zmniejszona moc
- M1 – podobny do M0, do implementacji w FPGA
- M3 – 0.9..1 DMIPS/MHz, podstawowa wersja dla typowych zastosowań 32-bitowych

Cortex M – c.d.

- M4 – 1.25 DMIPS/MHz, opcjonalna jednostka zmiennopozycyjna i wektorowa („M4F”), może zastępować proste procesory sygnałowe
- M7 (2015) - 2.15 DMIPS/MHz, dwupotokowy

Z czego składa się Cortex-M

- Procesor „post-RISC” o krótkim potoku (2..4 stopni)
- Timer systemowy SysTick do odmierzania stałych odcinków czasu
- Sterownik przerwań NVIC
  - Wielopoziomowy system przerwań z wywłaszczaniem

Procesor ARM Cortex-M

- 16 rejestrów uniwersalnych, w tym:
  - R13 – SP – wskaźnik stosu
  - R14 – LR – rejestr śladu
  - R15 – PC – licznik instrukcji
- Rejestry R0..7 dostępne dla wszystkich instrukcji, R8..R15 – tylko dla niektórych
- Rejestr stanu CPSR
- Zestaw instrukcji Thumb, Thumb2
  - Instrukcje 16- i 32-bitowe

Procesor ARM Cortex-M

- Architektura Load-Store
  - Operacje arytmetyczne i logiczne tylko 32-bitowe, wykonywame na rejestrach
  - Instrukcje ładowania i składowania danych 8-, 16- i 32-bitowych
  - Wymagane wyrównanie naturalne
- Instrukcje 3- i 2-argumentowe
- Model operacji warunkowych ze znacznikami

Procesor ARM Cortex-M

- Dwa poziomy zaufania – wątku i obsługi wyjątków
  - Na poziomie wątku można używać alternatywnego wskaźnika stosu
- 4..16 poziomów priorytetowych procesora
  - Wielopoziomowy system wyjątków
- Podstawowe mechanizmy ochrony
  - Błąd przy nielegalnych operacjach
  - Proste mechanizmy ochrony pamięci

Przestrzeń adresowa

- Jedna, 32-bitowa, liniowa przestrzeń adresowa
  - określony podział na obszary o różnym zastosowaniu (pamięci wewnętrzne i zewnętrzne, rejestry systemowe rdzenia, przestrzeń peryferiali)
  - Wszystkie zasoby mikrokontrolera, w tym rejestry systemowe procesora, są odwzorowane w lokacje przestrzeni adresowej

Mikrokontrolery STM32F4xx

- µC z 32-bitowym rdzeniem Cortex-M4
- Moc obliczeniowa > 20× większa od AVR ATmega
- Łatwo dostępny, popularny w Polsce
- Dostępny płytki uruchomieniowe serii Discovery, wyposażone w interfejs do programowania i debugowania µC, również na płytce docelowej użytkownika

Modele serii STM32F4

- F401 – model podstawowy, do 96 KiB RAM, do 84 MHz
- F410 – 32 KiB RAM
- F411 – 128 KiB RAM, do 100 MHz
- F405/407/415/417 – do 192 KiB RAM, do 168 MHz
- F42x/43x – 256 KiB RAM, 180 MHz
- F446 – 128 KiB RAM, 180 MHz
- F46x/47x – 384 KiB RAM, 180 MHz

STM32F401

- 128..512 KiB Flash, 64..96 KiB RAM
- Do 84 MHz
- 48..100 wyprowadzeń, 36..81 linii portów
- 8 timerów, kilkanaście kanałów PWM
- ADC 12-bit
- 3×USART, 4×SPI, 3×I2C
- SDIO, USB2.0 OTG Full Speed

STM32F - dokumentacja

- Reference Manual – zawiera pełny logiczny opis mikrokontrolera i jego peryferiów (działanie, rejestry, programowanie)
- Data Sheet – zawiera specyfikację parametrów elektrycznych i czasowych, zestawienie wyprowadzeń i opis ich funkcji
  - W tym tabele programowania wyboru funkcji poszczególnych wyprowadzeń

STM32 – oprogramowanie narzędziowe

- Środowiska programowania
  - Keil MDK-ARM (darmowa wersja 32 KiB)
  - AC6 (openstm32.org)
  - CooCox
  - DIY Eclipse + GCC-ARM
- Konfigurator/generator kodu STM32CubeMX
- Driver ST/Link STSW-LINK009
- Aktualizacja firmware STSW-LINK007

Cortex-M - programowanie

- Procesory przystosowane do programowania całkowicie w języku C
  - Łącznie z modułem startowym, który jednak w niektórych środowiskach jest pisany w asemblerze
- Procedury obsługi wyjątków nie różnią się od innych procedur
  - Zbędne rozszerzenia języka typu „interrupt”
    - (za wyjątkiem STM32F1 – starsza wersja rdzenia M3)

Start mikrokontrolera

- Początkowa wartość SP i PC pobierana z dwóch pierwszych słów pamięci
- Mikrokontroler startuje z wewnętrznym generatorem zegara RC o niewielkiej częstotliwości

Taktowanie STM32F

- Wewnętrzny generator RC ma precyzję rzędu 1%
  - Precyzja RC wystarcza do transmisji UART, ale nie nadaje się do odmierzania czasu ani do transmisji USB
- Oprogramowanie może wybrać inne źródło taktowania w celu uzyskania lepszej precyzji lub włączyć PLL w celu podwyższenia częstotliwości

Start oprogramowania

- Moduł startowy kolejno:
  - Wywołuje procedurę użytkownika SystemInit()
  - Przygotowuje środowisko pracy dla programu w języku C
  - Wywołuje funkcję main()

SystemInit()

- Funkcja zawiera ustawienia dotyczące taktowania procesora (źródło, programowanie PLL) i sterownika pamięci.
- Wzorcowa funkcja SystemInit() dostarczana przez producenta procesora lub środowiska
- Jeśli nie jest potrzebna – można zdefiniować własną (np. pustą) funkcję SystemInit()
- Funkcja nie może korzystać ze zmiennych zewnętrznych (bo jeszcze ich nie ma)

main()

- Główna funkcja programu w języku C, pisana przez użytkownika
- W niewielkich programach zawiera tylko inicjowanie peryferiów i uśpienie procesora
- Po włączeniu funkcji SleepOnExit procesora wykonanie instrukcji _WFI() powoduje, że procesor nie będzie wykonywał kodu wątku – zatrzyma się

Peryferiale

- Peryferiale znacznie bardziej funkcjonalne i bardziej złożone niż w µC 8-bitowych
- Każdy peryferial jest reprezentowany w języku C przez strukturę, której pola odpowiadają poszczególnym rejestrom sterującym

Zestaw peryferiali

- Blok sterowania RCC – odpowiada za włączanie innych peryferiali
- Porty GPIO – proste wejście-wyjście, konfiguracja połączeń wyprowadzeń z innymi peryferialami
- Timery TIMx – większość z funkcją PWM, wiele timerów wielokanałowych
- UART, SPI, I2C, ADC i wiele innych

Inicjowanie peryferiów

- W mikrokontrolerach z rdzeniami Cortex-M każdy peryferial wymaga włączenia przed zainicjowaniem i użyciem
- Włączenie następuje poprzez ustawienie bitu w odpowiednim rejestrze bloku RCC

Taktowanie

- Poszczególne bloki są podłączone do różnych szyn – AHB1/2, APB1, APB2
- Każda szyna jest taktowana niezależnie zegarem uzyskanym przez podział zegara podstawowego HCLK
- Częstotliwość APBx jest zwykle niższa od częstotliwości HCLK (przy wyższych częstotliwościach HCLK)
- Jeśli jest włączony dzielnik częstotliwości APB,timery są taktowane częstotliwością 2×wyższą niż częstotliwość danej szyny APB

Jak programować?

- Biblioteka SPL dostarczana przez producenta
  - Wydaje się prosta w użyciu, ale wymaga dużo pisania - „przegadane” programy
  - Powolne działanie, duża zajętość pamięci
- Biblioteka HAL (z STM32CubeMX)
- Operacje na rejestrach peryferiali
  - Krótsze w zapisie i znacznie szybsze w działaniu
  - Wymagają zapoznania się z dokumentacją
    - … co i tak kiedyś musi nastąpić przy projektowaniu rzeczywistych urządzeń

Płytka STM32F401DISCOVERY

- Płytka wyposażona w µC STM32F401VCT i interfejs do programowania i debugowania (USB)
- Zasilanie z interfejsu USB
- Przycisk, 4×LED, Audio DAC, mikrofon MEMS, Akcelerometr/żyroskop/magnetometr
- Większość wyprowadzeń µC dostępna na złączach

Piszemy program

- Tworzymy własny plik nagłówkowy, zawierający definicje zasobów sprzętowych – wejścia, wyjścia, np. „mojaplytka.h”
- Wygodnie jest w tym pliku umieścić dyrektywę włączającą plik definicji rejestrów µC
- W praktyce zwykle istnieje potrzeba uzupełnienia tych definicji w dodatkowym, własnym pliku

Instalacja środowiska Keil

- Po pobraniu i zainstalowaniu środowiska należy zainstalować pakiety ARM::CMSIS i Keil::STM32F4xx_DFP
- Zainstalować driver ST/Link STSW-LINK009
- Zaktualizować oprogramowanie ST/Link – STSW-LINK007

Szybki start – pierwszy projekt

- Uruchamiamy Keil MDK-ARM
- Project->New uVision Project
- Tworzymy i wybieramy folder Documents\STM32F4\First
- Nazwa projektu: First
- Wybieramy model uC: STM32F401VCTx
- Manage Run-Time Environment: zaznaczamy CMSIS:CORE i Device:Startup

Pierwszy projekt (2)

- Folder Target1/Source Group1: AddNew Item, C File, first.c
- Definujemy pustą funkcję main: void main(void) {}
- Oglądamy pliki startowe
- Kompilujemy projekt i sprawdzamy raport
  - Zajętość pamięci – kod programu i stałe – łącznie ok. 1 KiB

Pierwszy projekt (3)

- Kopiujemy plik stm32f4discovery.h do folderu projektu
- Kopiujemy folder STM32 z plikami nagłówkowymi do folderu z projektami (STM32F4)
- Włączamy plik stm32f4discovery.h do first.c
- Oglądamy schemat płytki i plik stm32f4discovery.h
  - Połączenia linii portów, kanały timera do sterowania LED

Pierwszy projekt – ustawienia!

- Project, Options for Target:
  - Zakładka C/C++
    - Ustawiamy zgodność z C99
    - Dodajemy folder STM32 do ścieżki poszukiwania plików nagłówkowych
  - Zakładka Debug
    - Use ST-Link Debugger
      - Settings: Debug – Port: SW, Connect: wybtrać Under reset, częstotliwość 480 kHz
      - Flash Download – zaznaczyć Reset and Run
  - Zakładka Utilities: Use Debug Driver

Pierwszy projekt – main()

- Włączamy zegar GPIOD RCC->AHB1ENR
- Opóźnienie, np. dostęp do RCC->APB1ENR
- Ustawiamy linię portu jako wyjście (MODER)
- Ustawiamy stan początkowy 1 (BSRR)
- Programujemy SysTtick na 100 Hz
- Włączamy usypianie (SleepOnExit) i usypiamy __WFI();

Pierwszy projekt – Systick_Handler()

- Deklarujemy zmienną timera programowego
- Na końcu okresu zerujemy timer i włączamy LED
- W połowie okresu wyłączamy LED

Pierwszy projekt - uruchomienie

- Kompilacja powinna zakończyć się bez błędów i ostrzeżeń
- Programowanie pamięci Flash uC - przycisk Load
  - Błąd oznacza zwykle nieprawidłowe ustawienia w zakładce Debug (-Settings) – opisane wcześniej
- Po zaprogramowaniu wybrna dioda LED powinna zacząć migać

Drugi projekt – PWM (1)

- Tworzymy projekt tak, jak poprzednio
- Możemy skopiować plik główny z pierwszego projektu i wyedytować go
- Definiujemy pożyteczne stałe:
  - Częstotliwość PWM – PWM_FREQ
  - Liczba kroków – PWM_STEPS
  - Minimalna i maksymalna jasność LED – LED_DIM, LED_MAX

Programowanie timera PWM

- Włączamy timer TIM4 w APB1ENR
  - Po zapisie AHB1ENR, przed inicjowaniem GPIO
- Ustawiamy preskaler (PRE) i okres (ARR)
- Włączamy tryb PWM w CCMR1 i CCMR2
- Włączamy sterowanie wyjść w CCER
- Włączamy przerwanie końca okresu w DIER
- Włączamy ARPE i timer w CR1

Obsługa przerwania timera PWM

- Przerwanie zgłaszana na końcu każdego okresu, z częstotliwością PWM_FREQ
- Obsługę zaczynamy od skasowania znacznika końca okresu w TIM4->SR

Płynne rozjaśnianie/ściemnianie LED

- Pomocnicza zmienna target przechowuje zadaną jasność diody
- W przerwaniu timera porównujemy jasność bieżącą z zadaną; jeśli są one różne zmieniamy jasność bieżącą o krok jednostkowy

Sterowanie serwomechanizmu modelarskiego

- Używamy kanału timera w trybie PWM
- Okres timera – 20 ms
- Czas trwania impulsu – 1..2 ms
- Określamy skok wypełnienia SERVO_STEP w µs
- Ustawiamy preskaler timera tak, by okres zegara był równy skokowi wypełnienia (APBx_TFREQ / 1000000 * SERVO_STEP)
- Definiujemy potrzebne stałe, wyrażone w jednostkach skoku wypełnienia

Interfejs I2C

- Jeden z typowych interfejsów międzyukładowych, stosowany w wielu układach czujników wielkości fizycznych
- Dwie linie sygnałowe:
  - SCL – zegar transmisji, sterowany przez urządzenie nadrzędne
    - Może być opcjonalnie „rozciągany” przez urządzenie podrzędne
  - SDA – dane, linia dwukierunkowa
- Częstotliwość zegara – do 100 kHz (400 kHz Fast Mode)

I2C - zdarzenia

- START – generowane przez układ nadrzędny przed transmisją adresu
- Wszystkie transmisje są 8-bitowe
- Transmisja każdego bajtu jest potwierdzana przez odbiorcę (ACK/NACK)
- Transmisja adresu (7 bitów) i bitu kierunku transmisji danych (odczyt/zapis)
- Transmisja danych (kierunek określony w bajcie adresu)
- STOP – koniec transmisji

I2C – odczyt z układu z wieloma rejestrami

- START
- Bajt adresu układu + zapis
- Zapis polecenia (np. numeru odczytywanego rejestru)
- START (bez uprzedniego STOP)
- Bajt adresu układu + odczyt
- Odczyt danych
- STOP

Obsługa wyświetlacza LCD z HD44780

- Czytamy kartę katalogową
  - Przy zasilaniu 5 V poziomy wejściowe zgodne z logiką 3.3 V
  - Wyzerowanie znacznika BUSY może następować na 4 µs przed zakończeniem operacji (przy 100 MHz jest to ok. 400 instrukcji procesora)
  - Krytyczne czasy: t_{AS}, t_{DSW}, min. połokres E
- Wnioski (dla Cortex-M o częstotliwości do 100 MHz): interfejs jednokierunkowy, transmisja w przerwaniu timera

LCD – analiza czasowa

- Proces odświeżania wyświetlacza powinien być niewidoczny dla użytkownika
  - Odświeżenie całej zawartości w czasie poniżej 1/30 s
- Czas wykonania większości poleceń w HD44780 wynosi do 40 μs
- Dla wyświetlacza 2×16:
  - Odświeżenie całości wymaga 34 poleceń – częstotliwość przesyłania poleceń powinna być nie niższa od 1 kHz

LCD – obsługa w przerwaniu timera

- Możliwe realizacje (interfejs 4-bitowy):
  - Jedno polecenie na przerwanie
    - Częstotliwość przerwań 1..2 kHz
    - Konieczność odczekiwania programowego tDSW i dolnego półokresu E
  - Jedna tetrada na przerwanie
    - Częstotliwość przerwań 2..4 kHz
    - Konieczność odczekiwania tDSW
  - Jedno zbocze E na przerwanie
    - Częstotliwość przerwań 4..8 kHz, bez oczekiwania

LCD - inicjowanie

- Polecenia używane podczas inicjowania mają dłuższe czasy wykonania i wymagane odstępy.
- Istotne jest wykonanie inicjowania w określonym czasie, poszczególne odstępy mogą być dłuższe od minimalnych wymaganych
- Inicjowanie może być wykonywane również w przerwaniu timera, ale z dodatkowym odliczaniem czasu pomiędzy kolejnymi etapami.

Współpraca z LCD

- Struktura zawiera zawartość dwóch wierszy tekstu i znaczniki żądania ich aktualizacji
- W celu wyświetlenia nowej zawartości zapisujemy tekst do bufora i ustawiamy znacznik aktualizacji wiersza
- Aktualizacja jest wykonywana w procedurze obsługi przerwania timera

STM32CubeMX

- Środowisko wspomagające projektowanie
- Graficzna konfiguracja zasbów uC
- Generowanie kodu w postaci gotowych projektów
  - Keil, AC6, IAR...
- Korzysta z biblioteki HAL
- Pakiety dla poszczególnych serii uC zawierają również programy demonstracyjne dla płytek Discovery

USB CDC z CubeMX

- Urządzenie USB widziane jako port szeregowy
  - Możliwa komunikacja przy uyciu programu termnala na PC
  - Łatwa interakcja z użytkownikiem
- Potrzebne:
  - CubeMX z pakietem dla wybranego uC
  - Driver ST VCOM STSW-STM32102 Virtual COM Port Driver
- Instalacja drivera
  - Uruchomić instalator instalatorów i zainstalować instalator
  - Uruchomić instalator drivera z Program Files(x86)/ST.../Software

USB CDC – konfiguracja peryferiów F401

- New Project – wybrać uC STM32F401VET
- Peryferiale:
  - RCC – HSE z zewn. oscylatorem
  - SYS – SWD
  - USB_OTG_FS – Device
- Middleware
  - USB CDC

USB CDC – konfiguracja taktowania

- Zakładka Clock Configuration
  - Źródło dla PLL – HSE
  - Częstotliwość HSE – 8 MHz
  - Predivider 8
  - PLL M – 192
  - Ustawić Q tak, by uzyskać 48 MHz
  - Źródło SYSCLK – PLL
  - Ustawić P tak, by uzyskać 72 MHz
  - Dzielnik AHB1 - 2

USB CDC – generowanie projektu

- Project-Settings
  - Folder dla projektu
  - Keil MDK-ARM 5.xx
  - Copy only...

USB CDC – niezbędne poprawki

- Moduł startowy – powiększyć dwukrotnie rozmiar stosu i sterty
- Plik usb_cdc_if.c
  - Ustawić rozmiary buforów na 64 B
  - W funkcji CDC_Receive_FS() przetworzyć dane i wywołać USBD_CDC_ReceivePacket(hUSBDevice_0)
  - Wysyłanie danych;
    - USBD_CDC_SetTXBuffer(hUsbDevice_0, buf, len)
    - USBD__CDC_TransmitPacket(hUSBDevice_0)