
KAMAMI
TECHDAYS

Programowanie µC w języku C - część praktyczna

Grzegorz Mazur

Plan warsztatów

- Środowisko Keil MDK-ARM
  - Tworzenie projektu
  - Ustawienia, składniki projektu
- SysTick, GPIO, przycisk
- ADC, USART, DMA
- Wyświetlacz multipleksowany LED, timer
- Wyświetlanie typu Charlieplexing

Programowanie STM32 – co trzeba mieć

- Płytka Nucleo lub Discovery
- przewód USB A-miniB
- Środowisko Keil MDK-ARM z zainstalowanymi pakietami:
  - ARM::CMSIS
  - Keil::STM32L4xx_DFP
- Zainstalowany driver ST-Link (STSW-LINK009.zip)

Zapoznanie ze środowiskiem Keil MDK-ARM

- Otwieramy Pack Installer
- W lewym panelu wybieramy zakładkę Boards, a w niej płytkę
- W prawym panelu wybieramy zakładkę Examples, przykład Blinky
- Naciskamy przycisk Copy, wybieramy folder i zaznaczamy Launch uVision

Program demonstracyjny

- Naciskamy Build
- Po kompilacji naciskamy Load

Przygotowanie do pracy

- Rozpakowujemy zawartość archiwum STM32_CWkshp do foldera roboczego o tej samej nazwie
- Folder STM32 zawiera kilka pożytecznych plików

Tworzenie własnego projektu w Keil MDK-ARM

- Z menu wybieramy Project – New uVision project
- Tworzymy folder projektu MyBlink w folderze STM32_CWkshp
- Nadajemy projektowi nazwę MyBlink
- Wybieramy typ mikrokontrolera STM32L476R

Tworzenie projektu (2)

- W oknie Manage Run-Time Environment zaznaczamy CMSIS-Core i Device-Startup
- Kopiujemy plik main.c z folderu 1_gpio do folderu MyBlink
- W panelu Project wybieramy Target 1 – Source Group 1, Add Existing Files...; dodajemy plik main.c

Tworzenie projektu (3)

- Target 1 – Options for Target
- Zakładka C/C++ - C99, dodajemy do ścieżki Include folder STM32
- Zakładka Debug – wybieramy ST-Link
  - Settings:
    - Debug – SWD, Max CLK – 1 MHz, Connect under Reset
    - Flash Download – Reset and Run

Tworzenie projektu (4)

- Zmienna tdiv służy do odliczania czasu w przerwaniu SysTick
  - Jak i gdzie zadeklarować zmienną tdiv?
- Build, Load, ...

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

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 definicji dostarczanych przez producenta w dodatkowym, własnym pliku

Nasza przestrzeń robocza

- Folder STM32 – pliki nagłówkowe
  - stm32l4yy.h – pożyteczne definicje dla uC, których brak w pliku stm32l4xx.h
  - Plik włącza inne pliki z definicjami dla STM32
  - Board.h, stm32nucleo64.h, n64comp1.h – definicje zasobów płytek

Sekcje pamięci - logiczne

- Text/code – kod programu
- Data/static – dane statyczne, w tym
  - stałe
  - Zmienne zainicjowane
  - Zmiene niezainicjowane – BSS
    - Zgodnie z wymaganiami standardu są inicjowane na wartość 0
- Stack – stos
- Heap – sterta – tylko przy alokacji dynamiczej

Sekcje w MDK-ARM

- Code – program
- RO-Data – stałe
- RW-Data – zmienne inicjowane na wartości niezerowe
- ZI-Data – zmienne inicjowane na wartość 0
  - w tym „nieinicjowane” statyczne

Zajętość pamięci przez program

- Flash: suma roziarów sekcji Code, RO-Data i RW-Data
- RAM: suma rozmiarów RW-Data i ZI-Data
- RW-Data: wartości początkowe są kopiowane z Flash do RAM przed wywołaniem funkcji main()

Przerwanie timera

- Podstawową częścią niemal każdego programu jest procedura obsługi przerwania timera, zgłaszanego ze stałą częstotliwością
- Używamy do tego celu timera SysTick, o ile w projekcie nie używamy innego timera np. do generowania przebiegów PWM

Projekty przykładowe

- Klikamy na pliku STM32_CWorkshop.umpw w folderze STM32_CWkshp

Program 1 – SimpleBlink (SysTick, GPIO)

- Definicje stałych – MCLK, częstotliwość przerwań SysTick
- Ustawienie linii portów LED jako wyjść GPIO, zaświecenie jednej diody
- Zainicjowanie timera SysTick na 100 Hz
- Przerwanie SysTick:
  - Zmiana stanu LED co 100 przerwań

Inicjowanie peryferiów

- W mikrokontrolerach z rdzeniami Cortex-M każdy peryferial wymaga włączenia przed zainicjowaniem i użyciem
- W STM32 służą do tego rejestry RCC->AHBENRx, RCC->APBxENRy

Konfiguracja portów

- W STM32L porty są domyślnie skonfigurowane jako wejścia analogowe
  - Odpowiada to wartości 11 pary bitów konfiguracji dla każdej linii w rejestrze MODER
  - Makro BF2A zwraca maskę bitową z jedynkami poza polem dot. wybranej linii – może ona być użyta z operatorem &
- Należy pamiętać o zachowaniu funkcji linii SWD

Program 1 – Przerwanie SysTick

- Zawiera całą funkcjonalność programu
- Zmienna tdiv służy do odliczenia zadanej liczby przerwań
  - Gdzie i jak ją zadeklarować?
- Jak zmienić stan wyjścia LED?

Program 1

- Co się stanie, gdy pominiemy atrybut static przy tdiv?
- Jak będzie sterowany port LED, jeśli w pętli zdarzeń znajdzie się operacja logiczna nie wpływająca na wyjście LED?

USART/UART

- Przypisanie wyprowadzeń – moduł GPIOx
  - Rejestry AFR – wybór funkcji
  - Rejestr MODER – ustawienie linii jako AF
- Programowanie USART:
  - Dzielnik zegara transmisji - UBRR
  - Format danych, zezwolenie na przerwania/DMA
  - Włączenie – rejestr CR1

Program 2 – UART

- Ustawienie linii portów dla UART
- Programowanie źródeł żądań DMA
- Zainicjowanie UART
- Zaprogramowanie timera SysTick – jak poprzednio
- Przerwanie SysTick – inicjowanie transmisji

Programowanie DMA

- Wybór modułu i kanału – RefMan
- Włączyć moduł DMA
- Wybór źródła żądania – DMA1_CSELR->CSELR
- Programowanie kanału
  - CPAR, CMAR, CNDTR
  - CCR – atrybuty i włączenie
- Podczas (re)programowania kanał musi być nieaktywny (DMA_CCR_EN = 0)!

Jak zadeklarować komunikaty?

- … wszystkie stałe
- … stałe i zmienne

Konfiguracja terminala

- Otwieramy program terminala TeraTerm
- Znajdujemy numer portu VCOM ST-Link
  - Menedżer urządzeń systemu Windows
- W terminalu (Setup-Serial Port) wybieramy port COMx i ustawiamy jego parametry:
  - Szybkość transmisji 115200 b/s
  - 8 bitów danych, bez bitu parzystości, 1 bit stopu
  - Brak sterowania przepływem

Programowanie timera TIM

- Zaczynamy od odblokowania taktowania timera – rejestr RCC->APBxENRy
- Definicje preprocesora umożliwiają wyliczenie wartości rejestrów timera na podstawie zadanych wartości
  - zdefiniowane w pliku disp.h; okres 100 cykli (stopni wypełnienia), częstotliwość przebiegu PWM – 400 Hz

Programowanie timera TIM

- Ustawienie preskalera i okresu (PSC, ARR)
- Tryby pracy kanałów - CCMRx
- Uaktywnienie i polaryzacja wyjść PWM
- Zezwolenia na przerwania i żądania DMA - DIER
- Włączenie timera - CR1

Program 3 – Wyświetlacz LED, ADC

- Wyświetlacz multipleksowany z regulacją jasności
- Przełączanie wartości wyświetlanej
- Pomiar napięcia zasilania
- Pomiar jasności oświetlenia
- Timer - sekundnik

Sterowanie LED - PWM

- Stałe MPX_FREQ, MPX_STEPS
- Jasność zadawana przez zmienną brightness_target
  - zakres 0..MPX_STEPS-2
  - Płynna zmiana jasności następuje w przerwaniu timera PWM
- Zamiast przerwania SysTick używamy przerwania końca okresu TIM - UI

ADC

- Przed użyciem wymagany start (z opóźnieniem) i kalibracja
  - Opis w RefMan
  - Sekwencję startową można wykonać przy użyciu przerwania timera
- Użyjemy najprostszego sposobu obsługi ADC – kolejne, pojedyncze pomiary w przerwaniu timera

Sprzętowa obsługa wyświetlacza - DMA

- Odświeżanie wyświetlacza można zrealizować bez użycia przerwania timera, korzystając z modułu DMA
  - Również regulacja jasności

Warianty projektu w MDK-ARM

- Na bazie aktualnego wariantu (target) tworzymy nowy, który początkowo jest jego kopią
- Po utworzeniu i wybraniu nowego wariantu możemy go modyfikować, np. poprzez zmianę ustawień
  - Definiujemy symbol preprocesora MPX_DMA

Program 4 - Charlieplexing

- Technika sterowania multipleksowanego przy użyciu n linii, n * (n-1) diod
  - 3 linie – 6 diod, 4 linie – 12 diod
- Trzy stany linii:
  - Wyłączona
  - Źródło napięciowe dla wspólnej elektrody
  - źródło prądowe dla jednej diody

KA_NUCLEO_MULTISENSOR

Charlieplexing

- Liczba faz = liczbie linii
- Dla konfiguracji ze wspólną anodą
  - W każdej fazie linia wspólnej anody podaje napięcie dodatnie (stan wysoki)
  - Aktywne katody wysterowane w stan niski
  - Nieaktywne katody – brak sterowania
    - Ustawione jako wejścia lub OD w stanie wysokim
- Wartości wyjściowe portów dla poszczególnych faz są stałe
  - Zmienia się sterowanie włączeniem linii

Charlieplexing – struktury danych

- Ponieważ maski sterowań portów jest trudno wyliczyć na podstawie pozycji diody, wygodnie jest stablicować maski diod
  - Mogą to być np. maski konfiguracji wyprowadzeń dla rejestru MODER
- Pętla przetwarza listę aktywnych diod na dane sterujące aktywnością linii portu w poszczególnych fazach

Pamięć Flash

- Możliwy zapis pod kontrolą oprogramowania (IAP- In-Aplication Programming)
- Może być dowolnie używana do przechowywania kodu lub danych
- Jednostka kasowalna – strona 2 KiB
- Jednostka programowalna – słowo 64 b

Zapis danych do Flash

- Włączenie zapisu w rejestrze FLASH->CR
- Zapis danych (2 słowa 32-bit) pod kolejne adresy w pamięci Flash
- Oczekiwanie na zakończenie i wyzerowanie znacznika EOP

Kasowanie strony Flash

- Zapis polecenia kasowania i numeru strony do rejestru FLASH->CR

Monitor zasilania

- Może zgłaszać przerwania przy obniżeniu napięcia zasilania poniżej określonej wartości
- Programowanie:
  - Włączenie detektora napięcia i ustawienie parametrów – rejestr PWR->CR2
    - Czekamy na stabilizację zasilania! - PWR->SR2
  - Włączenie przerwania od detektora napięcia – rejestr EXTI->
  - Włączenie przerwania EXTI w NVIC