Wbudy do budy

Dawid Gradowski (puckmoment na dc)

Luty 2025

Spis treści

1 Prolog **2**

2 GPIO **2**
2.1 Informacje ogólne 2
2.2 Guziczki 6

3 SPI **6**
3.1 Informacje ogólne 6
3.2 Wyświetlacz 7
3.3 Zapis na kartę pamięci 7

4 I^2C **7**
4.1 Czujnik natężenia światła 7
4.2 Termometr 7

5 RTC **7**

1 Prolog

Notatki robione w oparciu o projekt, który robiłem sam na zajęciach (Licznik jak coś). Proszę nie udostępniać Panience Ś.

Komponenciki		
Płytka	LPC1768/9	Intrukcja
Ekran OLED	Rodzaj	Intrukcja
Termometr	LM75	Instrukcja

2 GPIO

2.1 Informacje ogólne

GPIO (General Purpose Input/Output) jest interfejsem, który możemy wykorzystać zarówno jako wejście jak i wyjście. To jak zachowuje się ten interfejs zależne jest od stanu Enable Line. Jeśli stan Enable Line jest 1 to interfejs robi za wejście, a jeśli 0 to robi za wyjście.

Płytka LPC1768 ma 5 portów (oznaczone od 0 do 4) i każdemu z nich odpowiadają 4 rejestry 32 bitowe. Rejestry te w pliku oraz w programie możemy odnaleźć pod nazwami:

- **FIOPIN** - odczytywanie wartości na pinach
- **FIOSET** - ustawianie wartości 1 na pinach (ustawionych na output)
- **FIOCLR** - zerowanie wartości na pinach (ustawionych na output)
- **FIODIR** - ustalanie kierunku pinu (0: wejście| 1: wyjście)

Oznaczenie FIO oznacza Fast Input/Output, czyli IO (Input/Output) tylko szybsze (tak przeczytałem na forum to nie żart). Teraz trochę jak korzystać z tych rejestrów w praktyce. Mimo, że rejestry są 32 bitowe w przykładzie będę operował na pierwszych 4 bitach bo nie chce mi się tyle pisać. Na początku uznajmy, że wszystkie bity w rejestrach są ustawione na LOW (czyli 0).

Rejestr	3	2	1	0
FIOPIN	0	0	0	0
FIOSET	0	0	0	0
FIOCLR	0	0	0	0
FIODIR	0	0	0	0

Interfejs taki ma 32 piny i w tym momencie są ustawione na wejście. Urządzenie więc nie może tym interfejsem wysyłać sygnału. Załóżmy, że jakieś urządzenie jest podłączone do pinu oznaczonego numerem 3 i nadaje sygnał wysoki (1). Wtedy nasza tabelka będzie wyglądała tak:

Rejestr	3	2	1	0
FIOPIN	1	0	0	0
FIOSET	0	0	0	0
FIOCLR	0	0	0	0
FIODIR	0	0	0	0

Jako iż wszystkie bity w FIODIR są ustawione na 0, to żaden z naszych pinów nie jest wyjściem więc nie możemy zmieniać wartości na pinach. Aby zmienić wartość na konkretnym bicie możemu użyć następującej funkcji:

```
1  void GPIO_SetDir(uint8_t port, uint32_t bitValue, uint8_t dir↩
       )
2  {
3    LPC_GPIO_TypeDef *pGPIO = GPIO_GetPointer(port);
4
5    if (pGPIO != NULL) {
6      // Enable Output
7      if (dir) {
8        pGPIO->FIODIR |= bitValue;
9      }
10     // Enable Input
11     else {
12       pGPIO->FIODIR &= ~bitValue;
13     }
14   }
15 }
```

Jeśli chcielibyśmy na przykład używać pinu 2 jako wyjścia to wartość binarna w rejestrze FIODIR musi wyglądać następująco 0100 co jest równe 4 w dziesiętnym. Z tą wiedzą wiemy, że bit ten możemy zmienić wywołując funkcję na jeden z poniższych sposobów:

```
1 GPIO_SetDir(0, (1 << 2), 1); // preferowany
2 GPIO_SetDir(0, 0x4, 1);
3 GPIO_SetDir(0, 4, 1);
```

Po wywołaniu funkcji nasze rejestry będą wyglądały następująco:

Rejestr	3	2	1	0
FIOPIN	1	0	0	0
FIOSET	0	0	0	0
FIOCLR	0	0	0	0
FIODIR	0	1	0	0

Teraz możemy zmienić wartość bitu 2 w rejestrze FIOPIN, nie możemy tego jednak zrobić bezpośrednio. Aby zmienić wartość na pinie oznaczonym numerem 2 naszego interfejsu musimy wykorzystać rejestry **FIOSET** i **FIOCLR**. Aby zmienić wartość w tych 2 rejestrach możemy wykorzystać poniższe 2 funkcje:

```
1  void GPIO_SetValue(uint8_t portNum, uint32_t bitValue)
2  {
3      LPC_GPIO_TypeDef *pGPIO = GPIO_GetPointer(portNum);
4
5      if (pGPIO != NULL) {
6          pGPIO->FIOSET = bitValue;
7      }
8  }
9
10 void GPIO_ClearValue(uint8_t portNum, uint32_t bitValue)
11 {
12   LPC_GPIO_TypeDef *pGPIO = GPIO_GetPointer(portNum);
13
14   if (pGPIO != NULL) {
15     pGPIO->FIOCLR = bitValue;
16   }
17 }
```

Te 2 rejestry i funkcje działają w sposób bardzo podobny. Obydwa odpowiadają za zmiane wartości w rejestrze FIOPIN. FIOSET ustawia wartość na 1, FIOCLR ustawia wartość na 0. Obydwa rejestry po zmianie bitu w rejestrze FIOPIN od razu są zerowane. A więc by zmieniać wartość pinu 2, trzeba wywołać metodę w taki sposób:

```
1 GPIO_SetValue(0, (1 << 2)); // preferowany
2 GPIO_SetValue(0, 0x4);
3 GPIO_SetValue(0, 4);
```

Po wykonaniu tej funkcji wartości w rejestrach będą wyglądały następująco:

Rejestr	3	2	1	0
FIOPIN	1	0	0	0
FIOSET	0	1	0	0
FIOCLR	0	0	0	0
FIODIR	0	1	0	0

I od razu zostanie zmienione na

Rejestr	3	2	1	0
FIOPIN	1	1	0	0
FIOSET	0	0	0	0
FIOCLR	0	0	0	0
FIODIR	0	1	0	0

Wartość pinu zostanie zmieniona praktycznie natychmiast, według czatu GPT zazwyczaj w 1 lub 2 cyklach CPU, dlatego od razu po zmianie można odczytywać wartość w rejestrze FIOPIN oraz nie ma sensu odczytywać wartości z rejestrów FIOSET i FIOCLR bo w większości przypadków będą one równe 0. Metoda FIOCLR działa w ten sam sposób więc nie będę jej tłumaczył. Zarówno FIOSET jak i FIOCLR nie zadziałają na piny, dla których odpowiadające im wartości w rejestrze FIODIR są równe 0.

Aby odczytać wartości z rejestru FIOPIN należy użyć następującej funkcji:

```
uint32_t GPIO_ReadValue(uint8_t portNum)
{
  LPC_GPIO_TypeDef *pGPIO = GPIO_GetPointer(portNum);

  if (pGPIO != NULL) {
    return pGPIO->FIOPIN;
  }

  return (0);
}
```

Dana funkcja odczytuje jednak całą wartość rejestru a nie pojedynczego bitu. Jęśli chcemy napisać warunek zależny od tego czy mamy na bicie 3 wartość niską bądź wysoką, to możemy to napisać na parę sposobów:

```
// preferowane
if if ((GPIO_ReadValue(0) & (1 << 3)) == 0) {
    // niski sygnal
} else {
    // wysoki sygnal
}

if (((GPIO_ReadValue(0) >> 3) & 0x01) == 0) {
    // niski sygnal
} else {
    // wysoki sygnal
}

if if ((GPIO_ReadValue(0) & 8) == 0) {
    // niski sygnal
} else {
    // wysoki sygnal
}
```

2.2 Guziczki

3 SPI

3.1 Informacje ogólne

SPI jest szeregowym interfejsem urządzeń peryferyjnych. Jest nazywany protokołem master-slave. Za mastera uznaje się kontroler i jest on zawsze jeden a funkcję slave pełni urządzenie peryfejryne, które musi być jedno bądź więcej. Każde połączenie między masterem a slavem ma do 4 kabli, które są odpowiedzialne za 4 różne sygnały logiczne (logic signals). Te sygnały to:

- **CS** / **SS** (Chip Select lub Slave Select) - odpowiada za wybór urządzenia do komunikacji
- **SCLK** (Synchronous Clock) - odpowiada za synchronizacje i timing(?)
- **MOSI** (Master Out Slave In) - dane przesyłane przez mastera
- **MISO** (Master In Slave Out) - dane odbierane przez mastera

3.2 Wyświetlacz

3.3 Zapis na kartę pamięci

4 I^2C

4.1 Czujnik natężenia światła

4.2 Termometr

W przypadku termomemetru LM75 adres jest ustalany następująco:

				A_2	A_1	A_0
1	0	0	1	X	X	X

Pierwsze 4 bity są odczytane z instrukcji. Bity oznaczone A_x są ustalane zależnie od termometra na podstawie lutowania. Jeśli A_x jest przylutowany do gruntu (ground) to w adresie mamy 0, a jeśli do $+V_S$ to 1.

5 RTC