Wykład 7

Programowanie sprzętu w C

Struktura programu w C

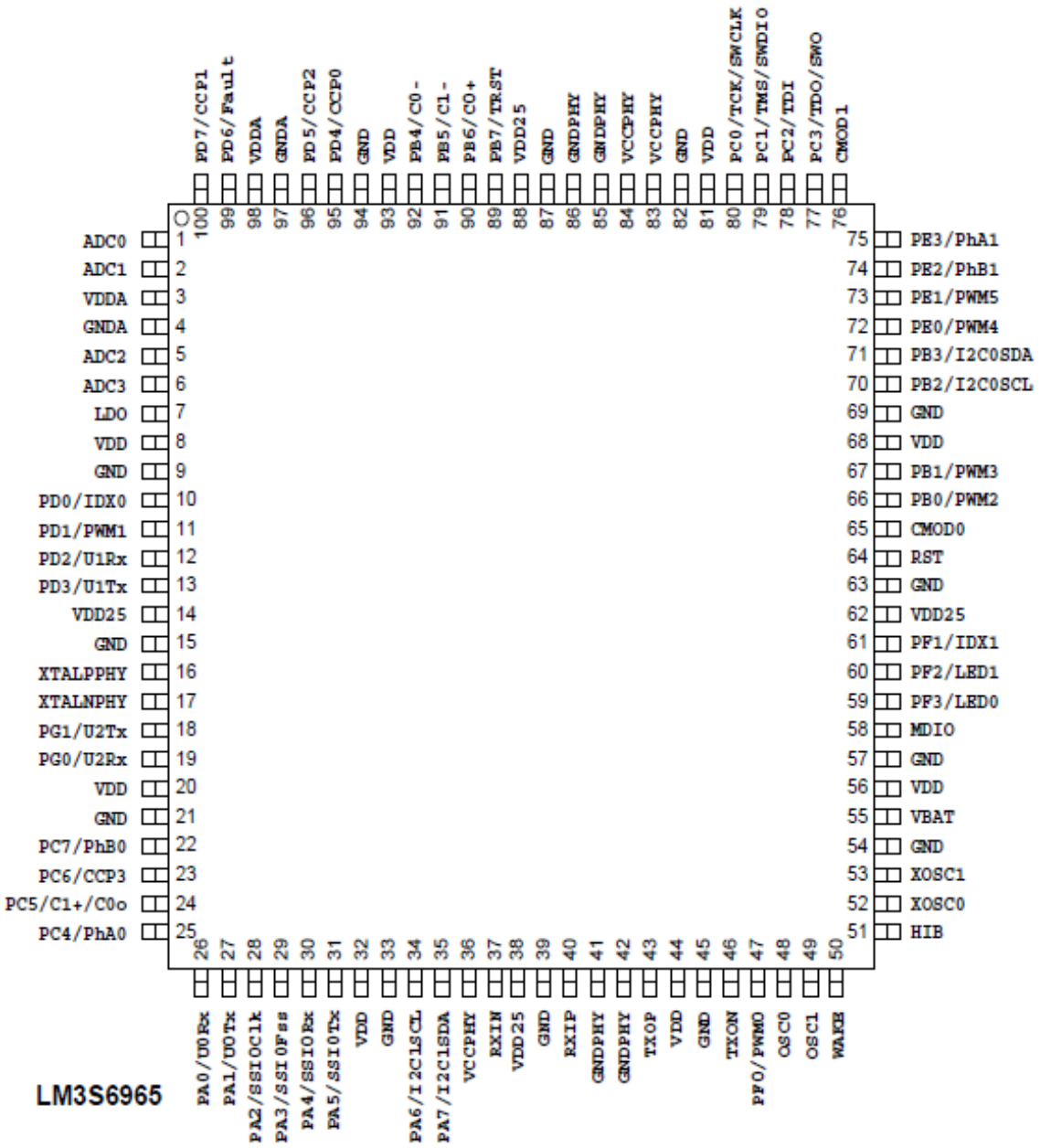
W celu utworzenia programu w języku C należy utworzyć projekt zawierający plik startup.s oraz plik programu z rozszerzeniem C.

Plik ten musi zawierać pętlę główną main.

Struktura programu w C

```
// hello.c - Simple hello world example.

#include "../../../hw_types.h"
#include "../../../src/debug.h"
#include "../../../src/sysctl.h"
#include "../../../utils/diag.h"
#include "../osram128x64x4.h"



// Print "Hello world!" to the OLED on the Stellaris evaluation board.
int
main(void)
{
    // Set the clocking to run directly from the crystal.
    SysCtlClockSet(SYSCTL_SYSDIV_1 | SYSCTL_USE_OSC | SYSCTL_OSC_MAIN |
                   SYSCTL_XTAL_8MHZ);

    // Initialize the OLED display.
    OSRAM128x64x4Init(1000000);

    // Hello!
    OSRAM128x64x4StringDraw("Hello World!", 30, 24, 15);

    // Exit.
    DiagExit(0);
}
```

Obsługa wejść-wyjść



Obsługa wejść-wyjść

- Moduł wejścia-wyjścia (GPIO) składa się z siedmiu bloków.
- Każdy z bloków odpowiada indywidualnemu portowi GPIO (Port A-G).
- Moduł GPIO może zawierać od 0-42 wejść-wyjść w zależności od konfiguracji.
- Każdy port jest niezależną jednostką fizyczną, w związku z czym LM3S6965 zawiera siedem niezależnych bloków GPIO.

Obsługa wejść-wyjść

- Własności portu GPIO:
  - programowe ustawienie przerwań GPIO,
  - maskowanie przerwań,
  - wyzwalanie zboczem przednim, tylnym lub obydwoma zboczami,
  - odporność na napięcie 5V,
  - programowe ustawianie konfiguracji pinów,
  - możliwość ustawiania prądu 2mA, 4mA, i 8mA na poszczególnych wejściach,
  - kontrola szybkości narastania dla prądu 8mA.

Obsługa wejść-wyjść

- Ważne! Wszystkie piny są w trybie domyślnym są trzy-stanowe (GPIOAFSEL=0, GPIODEN=0, GPIOPDR=0 i GPIOPUR=0), za wyjątkiem pięciu pinów JTAG/SWD (GPIOAFSEL=1,GPIODEN=1 and GPIOPUR=1).
- Po resecie porty są ustawiane w domyślnym trybie i należy je przeprogramować.

Sterowanie portami

- Rejestry danych umożliwiają programowe ustawienie trybów pracy GPIO.
- Rejestr kierunku danych konfiguruje wyprowadzenia jako wejścia lub jako wyjścia, w związku z tym dane są zapisywane do rejestru danych lub przepisywane na poszczególne piny kontrolera.
- Do sterowania przepływem danych wykorzystywany jest rejestr GPIO Direction (GPIODIR).
- Umożliwia on indywidualne skonfigurowanie poszczególnych pinów jako wejścia lub wyjścia.

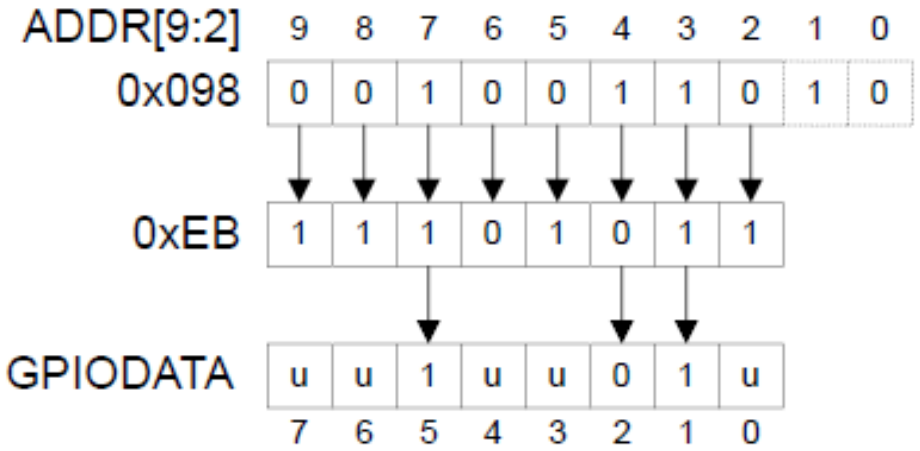
Sterowanie portami

- Kiedy wartość bitu kierunku jest 0 to port jest skonfigurowany jako wejście i odpowiadający mu rejestr danych odczytuje i zapamiętuje informacje z odpowiednich pinów.
- Jeżeli bit jest ustawiony na 1 to port pracuje w trybie wyjścia i dane z rejestru danych przepisywane są na piny wyjściowe.
- Dla zwiększenia efektywności programu, porty GPIO umożliwiają modyfikację poszczególnych bitów w rejestrze danych GPIO Data (GPIODATA) poprzez użycie bitów [9:2] szyny danych jako maski.

Sterowanie portami

- Przykład pokazujący zapis wartości 0xEB pod adres GPIODATA+0x098

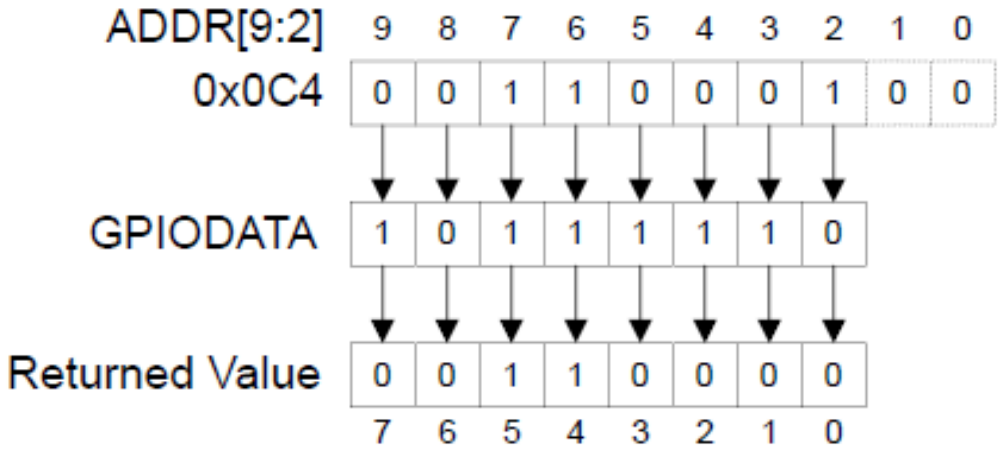
Figure 9-1. GPIODATA Write Example



Sterowanie portami

- Przykład odczytu danych z portu GPIODATA+0x0C4

Figure 9-2. GPIODATA Read Example



Ustawienia portów

- Piny portów GPIO mogą być sterowane sprzętowo lub programowo.
- Kiedy sterowanie sprzętowe jest uaktywnione w rejestrze GPIO Alternate Function Select (GPIOAFSEL), stan pinów jest kontrolowany przez dodatkowe funkcje związane z urządzeniami peryferyjnymi.
- Sterowanie programowe odpowiada trybowi pracy, w którym wykorzystywany jest rejestr GPIODATA do zapisu i odczytu poszczególnych pinów.

Ustawienia portów

- Do sterowania krytycznych urządzeń peryferyjnych można wykorzystać warstwę potwierdzania, która zapewnia ochronę przed przypadkowym przeprogramowaniem portów.
- Przy zapisie do zabezpieczonych bitów rejestru GPIO AFSEL dane nie są wpisywane, dopóki rejestr GPIO Lock (GPIOLOCK) nie ma odblokowanych odpowiadających bitów i bity rejestru GPIO Commit (GPIOCR) nie są ustawione na 1.

Ustawienia portów

- Sterowanie rejestrami pinów umożliwia programową konfigurację ich fizycznych własności, w zależności od wymagań aplikacji. można tego dokonać programując rejestry:
  - GPIODR2R,
  - GPIODR4R,
  - GPIODR8R,
  - GPIOODR,
  - GPIOPUR,
  - GPIOPDR,
  - GPIOSLR,
  - GPIODEN.

Ustawienia portów

- Domyślne wartości po resecie dla rejestrów GPIOAFSEL, GPIOPUR i GPIODEN są następujące:
- 0x0000.0000 dla wszystkich pinów GPIO, za wyjątkiem pięciu pinów JTAG/SWD (PB7 i PC[3:0]).
- Poprawne ustawienie tych pinów jest niezbędne dla zapewnienia funkcjonowania interfejsu JTAG/SWD.
- W wyniku tego domyślne wartości po resecie dla tych rejestrów wynoszą dla GPIO Port B 0x0000.0080, a dla Port C 0x0000.000F.

Ustawienia portów

Table 9-1. GPIO Pad Configuration Examples

Configuration	GPIO Register Bit Value[a]									
	AFSEL	DIR	ODR	DEN	PUR	PDR	DR2R	DR4R	DR8R	SLR
Digital Input (GPIO)	0	0	0	1	?	?	X	X	X	X
Digital Output (GPIO)	0	1	0	1	?	?	?	?	?	?
Open Drain Input (GPIO)	0	0	1	1	X	X	X	X	X	X
Open Drain Output (GPIO)	0	1	1	1	X	X	?	?	?	?
Open Drain Input/Output (I^2C)	1	X	1	1	X	X	?	?	?	?
Digital Input (Timer CCP)	1	X	0	1	?	?	X	X	X	X
Digital Input (QEI)	1	X	0	1	?	?	X	X	X	X
Digital Output (PWM)	1	X	0	1	?	?	?	?	?	?
Digital Output (Timer PWM)	1	X	0	1	?	?	?	?	?	?
Digital Input/Output (SSI)	1	X	0	1	?	?	?	?	?	?
Digital Input/Output (UART)	1	X	0	1	?	?	?	?	?	?
Analog Input (Comparator)	0	0	0	0	0	0	X	X	X	X
Digital Output (Comparator)	1	X	0	1	?	?	?	?	?	?

a. X=Ignored (don't care bit)

Ustawienia portów

Table 9-2. GPIO Interrupt Configuration Example

Register	Desired Interrupt Event Trigger	Pin 2 Bit Value[a]							
		7	6	5	4	3	2	1	0
GPIOIS	0=edge 1=level	X	X	X	X	X	0	X	X
GPIOIBE	0=single edge 1=both edges	X	X	X	X	X	0	X	X
GPIOIEV	0=Low level, or negative edge 1=High level, or positive edge	X	X	X	X	X	1	X	X
GPIOIM	0=masked 1=not masked	0	0	0	0	0	1	0	0

a. X=Ignored (don't care bit)

Mapa rejestrów GPIO

- Adresy bazowe odpowiadające poszczególnym portom GPIO są następujące
  - GPIO Port A: 0x4000.4000
  - GPIO Port B: 0x4000.5000
  - GPIO Port C: 0x4000.6000
  - GPIO Port D: 0x4000.7000
  - GPIO Port E: 0x4002.4000
  - GPIO Port F: 0x4002.5000
  - GPIO Port G: 0x4002.6000

Mapa rejestrów GPIO

Offset	Name	Type	Reset	Description
0x000	GPIODATA	R/W	0x0000.0000	GPIO Data
0x400	GPIODIR	R/W	0x0000.0000	GPIO Direction
0x404	GPIOIS	R/W	0x0000.0000	GPIO Interrupt Sense
0x408	GPIOIBE	R/W	0x0000.0000	GPIO Interrupt Both Edges
0x40C	GPIOIEV	R/W	0x0000.0000	GPIO Interrupt Event
0x410	GPIOIM	R/W	0x0000.0000	GPIO Interrupt Mask
0x414	GPIORIS	RO	0x0000.0000	GPIO Raw Interrupt Status
0x418	GPIOMIS	RO	0x0000.0000	GPIO Masked Interrupt Status
0x41C	GPIOICR	W1C	0x0000.0000	GPIO Interrupt Clear
0x420	GPIOAFSEL	R/W	-	GPIO Alternate Function Select
0x500	GPIODR2R	R/W	0x0000.00FF	GPIO 2-mA Drive Select
0x504	GPIODR4R	R/W	0x0000.0000	GPIO 4-mA Drive Select
0x508	GPIODR8R	R/W	0x0000.0000	GPIO 8-mA Drive Select

Mapa rejestrów GPIO

Offset	Name	Type	Reset	Description
0x50C	GPIOODR	R/W	0x0000.0000	GPIO Open Drain Select
0x510	GPIOPUR	R/W		GPIO Pull Up Select
0x514	GPIOPDR	R/W	0x0000.0000	GPIO Pull-Down Select
0x518	GPIOSLR	R/W	0x0000.0000	GPIO Slew Rate Control Select
0x51C	GPIODEN	R/W	-	GPIO Digital Enable
0x520	GPIOLOCK	R/W	0x0000.0001	GPIO Lock
0x524	GPIOCR	-	-	GPIO Commit
0xFD0	GPIOPeriphID4	RO	0x0000.0000	GPIO Peripheral Identification 4
0xFD4	GPIOPeriphID5	RO	0x0000.0000	GPIO Peripheral Identification 5
0xFD8	GPIOPeriphID6	RO	0x0000.0000	GPIO Peripheral Identification 6
0xFDC	GPIOPeriphID7	RO	0x0000.0000	GPIO Peripheral Identification 7
0xFE0	GPIOPeriphID0	RO	0x0000.0061	GPIO Peripheral Identification 0

API w języku C

- unsigned long GPIODirModeGet (unsigned long ulPort, unsigned char ucPin)
- void GPIODirModeSet (unsigned long ulPort, unsigned char ucPins, unsigned long ulPinIO)
- unsigned long GPIOIntTypeGet (unsigned long ulPort, unsigned char ucPin)
- void GPIOIntTypeSet (unsigned long ulPort, unsigned char ucPins, unsigned long ulIntType)
- void GPIOPadConfigGet (unsigned long ulPort, unsigned char ucPin, unsigned long #pulStrength, unsigned long #pulPinType)
- void GPIOPadConfigSet (unsigned long ulPort, unsigned char ucPins, unsigned long ulStrength, unsigned long ulPinType)
- void GPIOPinIntClear (unsigned long ulPort, unsigned char ucPins)

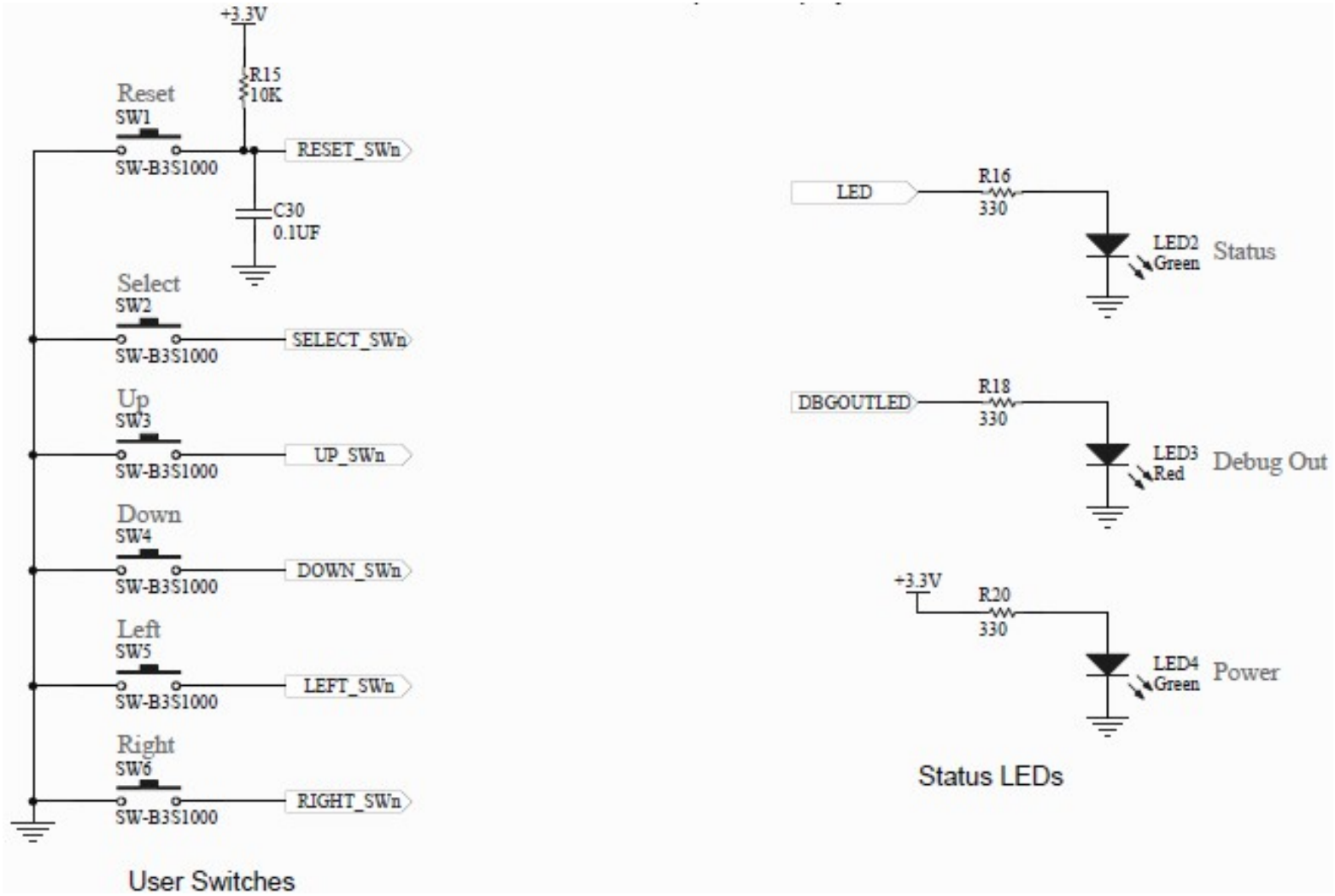
API w języku C

- void GPIOPinIntDisable (unsigned long ulPort, unsigned char ucPins)
- void GPIOPinIntEnable (unsigned long ulPort, unsigned char ucPins)
- long GPIOPinIntStatus (unsigned long ulPort, tBoolean bMasked)
- long GPIOPinRead (unsigned long ulPort, unsigned char ucPins)
- void GPIOPinTypeCAN (unsigned long ulPort, unsigned char ucPins)
- void GPIOPinTypeComparator (unsigned long ulPort, unsigned char ucPins)
- void GPIOPinTypeI2C (unsigned long ulPort, unsigned char ucPins)
- void GPIOPinTypePWM (unsigned long ulPort, unsigned char ucPins)

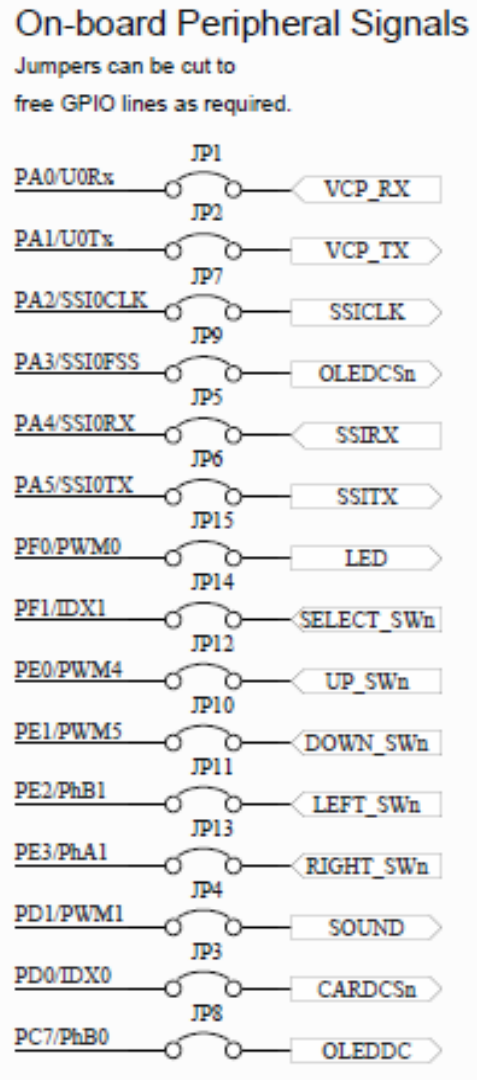
API w języku C

- void GPIOPinTypeQEI (unsigned long ulPort, unsigned char ucPins)
- void GPIOPinTypeSSI (unsigned long ulPort, unsigned char ucPins)
- void GPIOPinTypeTimer (unsigned long ulPort, unsigned char ucPins)
- void GPIOPinTypeUART (unsigned long ulPort, unsigned char ucPins)
- void GPIOPinWrite (unsigned long ulPort, unsigned char ucPins, unsigned char ucVal)
- void GPIOPortIntRegister (unsigned long ulPort, void (#pfIntHandler) (void))
- void GPIOPortIntUnregister (unsigned long ulPort)

Płyta ewaluacyjna



Płyta ewaluacyjna



Obsługa wyświetlacza

- void OSRAM128x64x4Clear (void)
- void OSRAM128x64x4Disable (void)
- void OSRAM128x64x4DisplayOff (void)
- void OSRAM128x64x4DisplayOn (void)
- void OSRAM128x64x4Enable (unsigned long ulFrequency)
- void OSRAM128x64x4ImageDraw (const unsigned char pucImage, unsigned long ulX, unsignedlong ulY, unsigned long ulWidth, unsigned long ulHeight)
- void OSRAM128x64x4Init (unsigned long ulFrequency)
- void OSRAM128x64x4StringDraw (const char pcStr, unsigned long ulX, unsigned long ulY,unsigned char ucLevel)

Obsługa wyświetlacza

- void OSRAM128x64x4Init(unsigned long ulFrequency)
- Funkcja inicjuje port SSI do którego podłączony jest wyświetlacz oraz konfiguruje układ kontrolny panelu OLED SSD0323.
- Parametry:
  - ulFrequency określa SSI Clock Frequency.

Obsługa wyświetlacza

- void OSRAM128x64x4StringDraw(const char *pcStr, unsigned long ulX, unsigned long ulY, unsigned char ucLevel)
- Wyświetlenie tekstu na wyświetlaczu OLED
- Parametry:
  - pcStr wskaźnik na ciąg znaków'
  - ulX kolumna,
  - ulY wiersz,
  - ucLevel poziom jasności tekstu, określony jako 4-bitowa wartość.
- Ponieważ wyświetlacz OLED pakuje dane dla dwóch pikseli w jeden bajt, parametr ulX musi być liczbą parzystą (np. 0, 2, 4, itp.).

Płyta ewaluacyjna

```
#include "../../../hw_memmap.h"
#include "../../../hw_types.h"
#include "../../../src/debug.h"
#include "../../../src/gpio.h"
#include "../../../src/sysctl.h"

// Sterowanie dioda LED
Int  main(void)
{
    volatile unsigned long ulLoop;

  // Ustawienie zegarów.
    SysCtlClockSet(SYSCTL_SYSDIV_1 | SYSCTL_USE_OSC | SYSCTL_OSC_MAIN | SYSCTL_XTAL_8MHZ);

    // Konfiguracja portow sterujacych dioda LED.
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    GPIOPadConfigSet(GPIO_PORTF_BASE, GPIO_PIN_0, GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD);
    GPIODirModeSet(GPIO_PORTF_BASE, GPIO_PIN_0, GPIO_DIR_MODE_OUT);

    while(1)
    {
        // Wlaczenie diody.
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, GPIO_PIN_0);

         for(ulLoop = 0; ulLoop < 200000; ulLoop++)        // Opoznienie.
        {
        }

        // Wylaczenie diody./
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, 0);

        for(ulLoop = 0; ulLoop < 200000; ulLoop++)        // Opoznienie.
        {
        }
    }
}
```

Podłączenie silnika do I/O