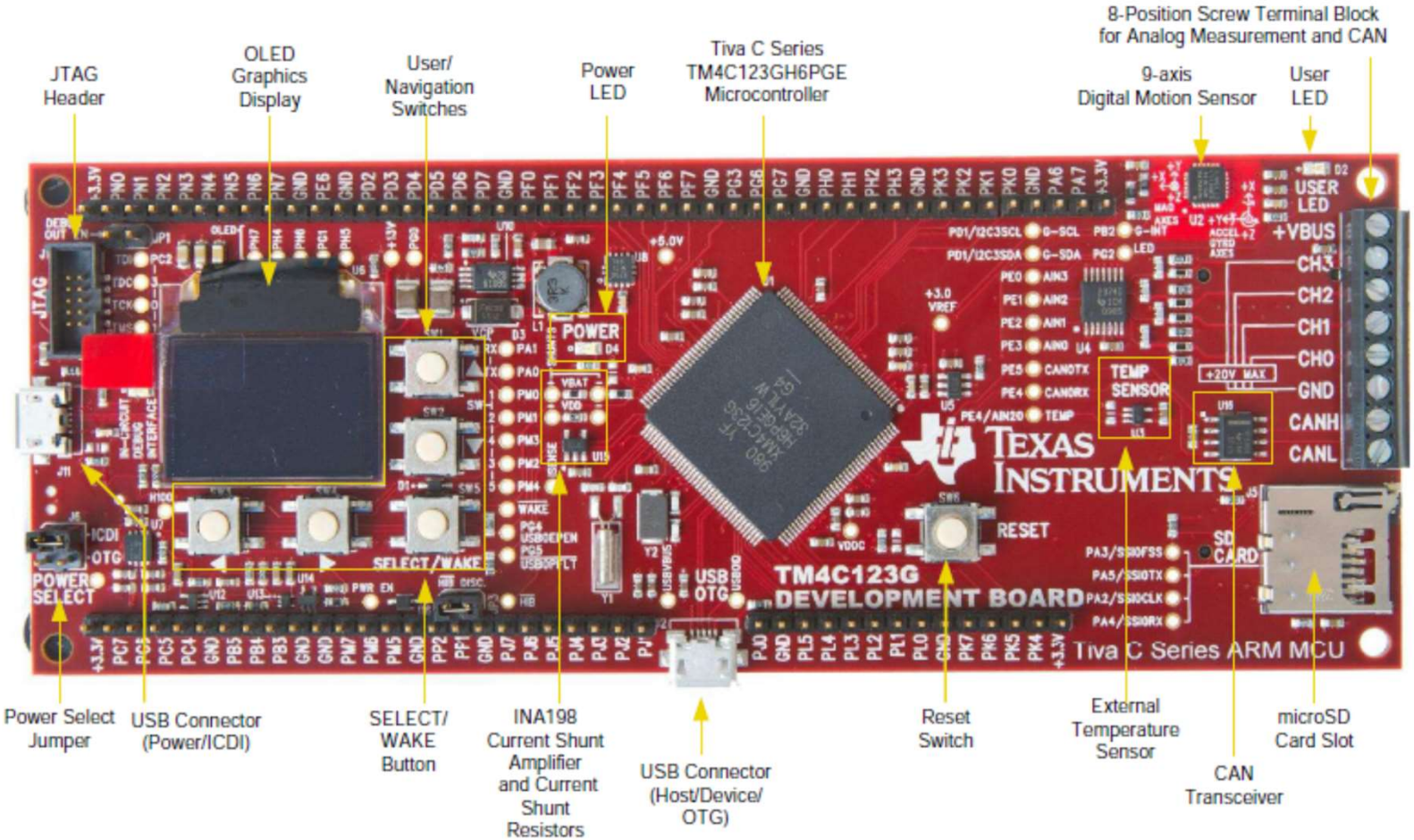
Tiva C Series

ARM MCU


JTAG Header
OLED Graphics Display
User/ Navigation Switches
Power LED
Tiva C Series TM4C123GH6PGE Microcontroller
8-Position Screw Terminal Block for Analog Measurement and CAN
9-axis Digital Motion Sensor
User LED
Power Select Jumper
USB Connector (Power/ICDI)
SELECT/ WAKE Button
INA198 Current Shunt Amplifier and Current Shunt Resistors
USB Connector (Host/Device/ OTG)
Reset Switch
External Temperature Sensor
CAN Transceiver
microSD Card Slot
TM4C123G DEVELOPMENT BOARD
Tiva C Series ARM MCU
TEXAS INSTRUMENTS
POWER
RESET
SELECT/WAKE
TEMP SENSOR
SD CARD
USB OTG
+VBUS
CH3
CH2
CH1
CH0
GND
CANH
CANL

Procesador

- ARM Cortex M4F con referencia TM4C123GH6PM
- 32 bits
- Bajo consumo de energía 370 uA / Mhz
- Operaciones con punto flotante
- Hasta 80 MHz
- Multiplicación y división en un solo ciclo máquina
- Operaciones MAC

Memoria

- Memoria de datos SRAM de 32 kB , donde se almacenan los datos temporales
- Memoria de programa tipo Flash de 256 kB , donde se guardan las instrucciones
- Memoria de datos constantes tipo EEPROM de 2 kB
- Memoria de programa de arranque (Bootloader) , que sirve para programación con TivaWare

Reloj

- Hay cuatro fuentes de reloj para la operación del microprocesador TM4C123GH6PM :
- Oscilador interno de precisión de 16 Mhz
- Oscilador interno de baja frecuencia para modos de bajo consumo de energía
- Oscilador externo con PLL . Puede conectarse un generador de señales externo a un pin específico o poner un cristal de cuarzo entre los dos pines disponibles .
- Oscilador de tiempo real externo del módulo de hibernación . Es un cristal de cuarzo de 32.768 hertz que se coloca entre dos pines disponibles para este propósito .

Periféricos

- 43 líneas de entrada / salida digitales
- 16 salidas PWM o moduladores de ancho de pulso
- codificador de cuadratura
- Un módulo analógico que tiene dos convertidores analógicos a digitales (ADC) de 12 bits cada uno , con un total de 12 canales de medición y una velocidad de muestreo de hasta un millón de muestras por segundo
- dos comparadores analógicos y un regulador de voltaje
- 12 contadores de 16 y 32 bits
- Un sistema completo de comunicaciones seriales como UARTs , USB , I2C , SSI y CAN

Programación en C

- Con los conocimientos previos de programación en lenguaje ensamblador la programación en C va a ser más corta.
- Para empezar, vamos a hacer un encendido de los leds RGB de la tarjeta Tiva Launchpad , de igual forma como se hizo al comienzo con lenguaje ensamblador .

- Los pasos a seguir son los siguientes:
- 1. Configuración del puerto F con los bits PF1 , PF2 y PF3 como salidas .
- 2. Poner ‘ 1 ’ en PF1 y ceros en los otros dos bits .
- 3. Hacer un retardo por decremento de variable .
- 4. Poner un ‘ 1 ’ en PF2 y ceros en los otros dos bits .
- 5. Hacer un retardo igual al anterior .
- 6. Poner en ‘ 1 ’ el bit PF3 y ceros en los otros dos bits .
- 7. Hacer el mismo retardo.
- 8. Ir al paso 2 .

Esta misma secuencia al realizarla en lenguaje ensamblador, hay varios registros en memoria SRAM que deben inicializarse.

Para usar un puerto digital primero se debe inicializar y esto se hace en siete pasos .

1. Activar el reloj del puerto con el registro SYSCTL_RCGC2 .
2. Desbloquear el puerto . Sólo para los pines PC3 - 0 , PD7 y PF0 , con el registro GPIO_PORTx_LOCK y el registro GPIO_PORTx_CR , donde x es el puerto .
3. Deshabilitar la función analógica , con el registro GPIO_PORTx_AMSEL .
4. Borrar bits en PCTL para funciones digitales , con el registro GPIO_PORTx_PCTL .
5. Seleccionar la dirección de los pines del puerto , con el registro GPIO_PORTx_DIR .
6. Borrar bits en el registro de funciones alternas , con el registro GPIO_PORTx_AFSEL .
7. Habilitar el puerto con el registro GPIO_PORTx_DEN .

En C se definen estas direcciones de la siguiente manera :

```
# define SYSCTL_RCGC2_R ( * ( ( volatile unsigned long * ) 0x400FE108 ) )
# define GPIO_PORTF_LOCK_R ( * ( ( volatile unsigned long * ) 0x40025520 ) )
# define GPIO_PORTF_CR_R ( * ( ( volatile unsigned long * ) 0x40025524 ) )
# define GPIO_PORTF_AMSEL_R ( * ( ( volatile unsigned long * ) 0x40025528 ) )
# define GPIO_PORTF_PCTL_R ( * ( ( volatile unsigned long * ) 0x4002552C ) )
# define GPIO_PORTF_DIR_R ( * ( ( volatile unsigned long * ) 0x40025400 ) )
# define GPIO_PORTF_AFSEL_R ( * ( ( volatile unsigned long * ) 0x40025420 ) )
# define GPIO_PORTF_PUR_R ( * ( ( volatile unsigned long * ) 0x40025510 ) )
# define GPIO_PORTF_DEN_R ( * ( ( volatile unsigned long * ) 0x4002551C ) )
# define GPIO_PORTF_DATA_R ( * ( ( volatile unsigned long * ) 0x400253FC ) )
```

La configuración de estos registros se puede poner como una función:

```
void PortF_Ini ( void ) {
volatile unsigned long retardo ; SYSCTL_RCGC2_R | = 0x00000020 ; / / 1 . Habilitación del reloj
retardo = SYSCTL_RCGC2_R ; / / un pequeño retardo
GPIO_PORTF_LOCK_R = 0x4C4F434B ; / / 2 . desbloqueo de PF0
GPIO_PORTF_CR_R = 0x1F ; / / permite cambios en PF4 - 0
GPIO_PORTF_AMSEL_R = 0x00 ; / / 3 . deshabilita analógica en PF
GPIO_PORTF_PCTL_R = 0x00000000 ; / / 4 . PCTL GPIO on PF4 - 0
GPIO_PORTF_DIR_R = 0x0E ; / / 5 . PF4 , PF0 in , PF3 - 1 out
GPIO_PORTF_AFSEL_R = 0x00 ; / / 6 . Desh . func . alternas en PF
GPIO_PORTF_PUR_R = 0x11 ; / / Res . pull - up PF0 y PF4
GPIO_PORTF_DEN_R = 0x1F ; / / 7 . Hab . I / O digital en PF4 - 0
}
```

- El programa en C que hace el encendido de los leds queda de la siguiente manera .

```
# define LED_ROJO 2
int main ( void )
 {
 int led = LED_ROJO ; double i ;
PortF_Ini ( ) ;
while ( 1 )
 {
 GPIO_PORTF_DATA_R = led ;
 for ( i = 0 ; i < 100000 ; i + + ) ; / / retardo
led = led * 2 ;
 if ( led = = 16 ) led = LED_ROJO ;
 }
}
```

- Observe que en C se puede escribir directamente sobre el puerto de datos para los pines de salida.
- En el caso de leer una entrada, por ejemplo, para leer el bit 4 del puerto F se usa:
- In = GPIO_PORTF_DATA_R & 0x10; .