

System klok

For a 125MHz system clock with a 12MHz input, the minimum VCO frequency is quite high.

```
$ ./vcocalc.py -l 125
Requested: 125.0 MHz
Achieved: 125.0 MHz
FBDIV: 125 (VCO = 1500 MHz)
PD1: 6
PD2: 2
```

We can restrict the search to lower VCO frequencies, so that the script will consider looser frequency matches. Note that, whilst a 750MHz VCO would be ideal here, we can't achieve exactly 750MHz by multiplying the 12MHz input by an integer, which is why the previous invocation returned such a high VCO frequency.

```
$ ./vcocalc.py -l 125 --vco-max 800
Requested: 125.0 MHz
Achieved: 126.0 MHz
FBDIV: 63 (VCO = 756 MHz)
PD1: 6
PD2: 1
```

A 126MHz system clock may be a tolerable deviation from the desired 125MHz, and generating this clock consumes less power at the PLL.

GPIO functies

| | Function | | | | | | | | |
|------|----------|-----------|----------|--------|-----|------|------|----|---------------|
| GPIO | F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 | F9 |
| 0 | SPI0 RX | UART0 TX | I2C0 SDA | PWM0 A | SIO | PI00 | PI01 | | USB OVCUR DET |
| 1 | SPI0 CSn | UART0 RX | I2C0 SCL | PWM0 B | SIO | PI00 | PI01 | | USB VBUS DET |
| 2 | SPI0 SCK | UART0 CTS | I2C1 SDA | PWM1 A | SIO | PI00 | PI01 | | USB VBUS EN |
| 3 | SPI0 TX | UART0 RTS | I2C1 SCL | PWM1 B | SIO | PI00 | PI01 | | USB OVCUR DET |
| 4 | SPI0 RX | UART1 TX | I2C0 SDA | PWM2 A | SIO | PI00 | PI01 | | USB VBUS DET |
| 5 | SPI0 CSn | UART1 RX | I2C0 SCL | PWM2 B | SIO | PI00 | PI01 | | USB VBUS EN |
| 6 | SPI0 SCK | UART1 CTS | I2C1 SDA | PWM3 A | SIO | PI00 | PI01 | | USB OVCUR DET |
| 7 | SPI0 TX | UART1 RTS | I2C1 SCL | PWM3 B | SIO | PI00 | PI01 | | USB VBUS DET |
| 8 | SPI1 RX | UART1 TX | I2C0 SDA | PWM4 A | SIO | PI00 | PI01 | | USB VBUS EN |
| 9 | SPI1 CSn | UART1 RX | I2C0 SCL | PWM4 B | SIO | PI00 | PI01 | | USB OVCUR DET |

| | Function | | | | | | | | |
|----|----------|-----------|----------|--------|-----|------|------|--------------|---------------|
| 10 | SPI1 SCK | UART1 CTS | I2C1 SDA | PWM5 A | SIO | PI00 | PI01 | | USB VBUS DET |
| 11 | SPI1 TX | UART1 RTS | I2C1 SCL | PWM5 B | SIO | PI00 | PI01 | | USB VBUS EN |
| 12 | SPI1 RX | UART0 TX | I2C0 SDA | PWM6 A | SIO | PI00 | PI01 | | USB OVCUR DET |
| 13 | SPI1 CSn | UART0 RX | I2C0 SCL | PWM6 B | SIO | PI00 | PI01 | | USB VBUS DET |
| 14 | SPI1 SCK | UART0 CTS | I2C1 SDA | PWM7 A | SIO | PI00 | PI01 | | USB VBUS EN |
| 15 | SPI1 TX | UART0 RTS | I2C1 SCL | PWM7 B | SIO | PI00 | PI01 | | USB OVCUR DET |
| 16 | SPI0 RX | UART0 TX | I2C0 SDA | PWM0 A | SIO | PI00 | PI01 | | USB VBUS DET |
| 17 | SPI0 CSn | UART0 RX | I2C0 SCL | PWM0 B | SIO | PI00 | PI01 | | USB VBUS EN |
| 18 | SPI0 SCK | UART0 CTS | I2C1 SDA | PWM1 A | SIO | PI00 | PI01 | | USB OVCUR DET |
| 19 | SPI0 TX | UART0 RTS | I2C1 SCL | PWM1 B | SIO | PI00 | PI01 | | USB VBUS DET |
| 20 | SPI0 RX | UART1 TX | I2C0 SDA | PWM2 A | SIO | PI00 | PI01 | CLOCK GPIN0 | USB VBUS EN |
| 21 | SPI0 CSn | UART1 RX | I2C0 SCL | PWM2 B | SIO | PI00 | PI01 | CLOCK GPOUT0 | USB OVCUR DET |
| 22 | SPI0 SCK | UART1 CTS | I2C1 SDA | PWM3 A | SIO | PI00 | PI01 | CLOCK GPIN1 | USB VBUS DET |
| 23 | SPI0 TX | UART1 RTS | I2C1 SCL | PWM3 B | SIO | PI00 | PI01 | CLOCK GPOUT1 | USB VBUS EN |
| 24 | SPI1 RX | UART1 TX | I2C0 SDA | PWM4 A | SIO | PI00 | PI01 | CLOCK GPOUT2 | USB OVCUR DET |
| 25 | SPI1 CSn | UART1 RX | I2C0 SCL | PWM4 B | SIO | PI00 | PI01 | CLOCK GPOUT3 | USB VBUS DET |
| 26 | SPI1 SCK | UART1 CTS | I2C1 SDA | PWM5 A | SIO | PI00 | PI01 | | USB VBUS EN |
| 27 | SPI1 TX | UART1 RTS | I2C1 SCL | PWM5 B | SIO | PI00 | PI01 | | USB OVCUR DET |
| 28 | SPI1 RX | UART0 TX | I2C0 SDA | PWM6 A | SIO | PI00 | PI01 | | USB VBUS DET |
| 29 | SPI1 CSn | UART0 RX | I2C0 SCL | PWM6 B | SIO | PI00 | PI01 | | USB VBUS EN |

Temperatuur sensor

4.9.5. Temperature Sensor

The temperature sensor measures the Vbe voltage of a biased bipolar diode, connected to the fifth ADC channel (AINSEL=4). Typically, Vbe = 0.706V at 27 degrees C, with a slope of -1.721mV per degree. Therefore the temperature can be approximated as follows:

$$T = 27 - (\text{ADC_voltage} - 0.706)/0.001721$$

As the Vbe and the Vbe slope can vary over the temperature range, and from device to device, some user calibration may be required if accurate measurements are required.

The temperature sensor's bias source must be enabled before use, via [CS.TS_EN](#). This increases current consumption on ADC_AVDD by approximately 40µA.

NOTE

The on board temperature sensor is very sensitive to errors in the reference voltage. If the ADC returns a value of 891 this would correspond to a temperature of 20.1°C. However if the reference voltage is 1% lower than 3.3V then the same reading of 891 would correspond to 24.3°C. You would see a change in temperature of over 4°C for a small 1% change in reference voltage. Therefore if you want to improve the accuracy of the internal temperature sensor it is worth considering adding an external reference voltage.

4.9.6. List of Registers

The ADC registers start at a base address of [0x4004c000](#) (defined as [ADC_BASE](#) in SDK).

| Offset | Name | Info |
|--------|------------------------|--|
| 0x00 | CS | ADC Control and Status |
| 0x04 | RESULT | Result of most recent ADC conversion |
| 0x08 | FCS | FIFO control and status |
| 0x0c | FIFO | Conversion result FIFO |
| 0x10 | DIV | Clock divider. If non-zero, CS_START_MANY will start conversions at regular intervals rather than back-to-back. The divider is reset when either of these fields are written. Total period is 1 + INT + FRAC / 256 |
| 0x14 | INTR | Raw Interrupts |
| 0x18 | INTE | Interrupt Enable |
| 0x1c | INTF | Interrupt Force |
| 0x20 | INTS | Interrupt status after masking & forcing |

Vector tabel

| Exception number | IRQ number | Vector | Offset |
|------------------|------------|-------------------------|---------|
| 16+n | n | IRQn | 0x40+4n |
| · | · | · | · |
| · | · | · | · |
| · | · | · | · |
| 18 | 2 | IRQ2 | 0x48 |
| 17 | 1 | IRQ1 | 0x44 |
| 16 | 0 | IRQ0 | 0x40 |
| 15 | -1 | SysTick, if implemented | 0x3C |
| 14 | -2 | PendSV | 0x38 |
| 13 | | Reserved | |
| 12 | | | |
| 11 | -5 | SVCall | 0x2C |
| 10 | | | |
| 9 | | | |
| 8 | | | |
| 7 | | Reserved | |
| 6 | | | |
| 5 | | | |
| 4 | | | |
| 3 | -13 | HardFault | 0x10 |
| 2 | -14 | NMI | 0x0C |
| | | | 0x08 |
| 1 | | Reset | 0x04 |
| | | Initial SP value | 0x00 |

Figure 2-1 Vector table

| IRQ | Interrupt Source | IRQ | Interrupt Source | IRQ | Interrupt Source | IRQ | Interrupt Source | IRQ | Interrupt Source |
|-----|------------------|-----|------------------|-----|------------------|-----|------------------|-----|------------------|
| 0 | TIMER_IRQ_0 | 6 | XIP_IRQ | 12 | DMA_IRQ_1 | 18 | SPI0_IRQ | 24 | I2C1_IRQ |
| 1 | TIMER_IRQ_1 | 7 | PIO0_IRQ_0 | 13 | IO_IRQ_BANK0 | 19 | SPI1_IRQ | 25 | RTC_IRQ |
| 2 | TIMER_IRQ_2 | 8 | PIO0_IRQ_1 | 14 | IO_IRQ_QSPI | 20 | UART0_IRQ | | |
| 3 | TIMER_IRQ_3 | 9 | PIO1_IRQ_0 | 15 | SIO_IRQ_PROC0 | 21 | UART1_IRQ | | |
| 4 | PWM_IRQ_WRAP | 10 | PIO1_IRQ_1 | 16 | SIO_IRQ_PROC1 | 22 | ADC_IRQ_FIFO | | |
| 5 | USBCTRL_IRQ | 11 | DMA_IRQ_0 | 17 | CLOCKS_IRQ | 23 | I2C0_IRQ | | |

IRQ tabel

Timer en alarm registers

| | | |
|------|----------|--|
| 0x0c | TIMELR | Read from bits 31:0 of time |
| 0x10 | ALARM0 | Arm alarm 0, and configure the time it will fire. Once armed, the alarm fires when <code>TIMER_ALARM0 == TIMELR</code> . The alarm will disarm itself once it fires, and can be disarmed early using the ARMED status register. |
| 0x14 | ALARM1 | Arm alarm 1, and configure the time it will fire. Once armed, the alarm fires when <code>TIMER_ALARM1 == TIMELR</code> . The alarm will disarm itself once it fires, and can be disarmed early using the ARMED status register. |
| 0x18 | ALARM2 | Arm alarm 2, and configure the time it will fire. Once armed, the alarm fires when <code>TIMER_ALARM2 == TIMELR</code> . The alarm will disarm itself once it fires, and can be disarmed early using the ARMED status register. |
| 0x1c | ALARM3 | Arm alarm 3, and configure the time it will fire. Once armed, the alarm fires when <code>TIMER_ALARM3 == TIMELR</code> . The alarm will disarm itself once it fires, and can be disarmed early using the ARMED status register. |
| 0x20 | ARMED | Indicates the armed/disarmed status of each alarm. A write to the corresponding <code>ALARMx</code> register arms the alarm. Alarms automatically disarm upon firing, but writing ones here will disarm immediately without waiting to fire. |
| 0x24 | TIMERAWH | Raw read from bits 63:32 of time (no side effects) |
| 0x28 | TIMERAWL | Raw read from bits 31:0 of time (no side effects) |
| 0x2c | DBGPAUSE | Set bits high to enable pause when the corresponding debug ports are active |
| 0x30 | PAUSE | Set high to pause the timer |
| 0x34 | INTR | Raw Interrupts |
| 0x38 | INTE | Interrupt Enable |