PWM Example Using Servomotor

Servomotor CMakeLists.txt

```
cmake_minimum_required(VERSION 3.13.1)
find_package(Zephyr REQUIRED HINTS $ENV{ZEPHYR_BASE})
project(servo_motor)
FILE(GLOB app_sources src/*.c)
target_sources(app PRIVATE ${app_sources})
```

Servomotor sample.yaml

```
sample:
  name: Servo Motor using PWM
tests:
  sample.basic.servo_motor:
    tags: drivers pwm
    depends_on: pwm
    harness: motor
    platform_allow: nrf52840dk_pca10056
      # platform_allow: nrf52840dk_pca10056
      # integration_platforms:
      #   nrf52840dk_pca10056
```

Servomotor prj.conf

```
CONFIG_STDOUT_CONSOLE=y
CONFIG_PRINTK=y
CONFIG_PWM=y
```


Servomotor Overlay file

```
/ {
        aliases {
        pwm-servo = &sw_pwm;
    };
};

&sw_pwm {
        status = "okay";
    clock-prescaler = <3>;
};
```

An important thing here to note is that we get a **devicetree error** if the *status = "okay";* is not included in the overlay file. By default (vanilla PWM example), this line will not be present.

Servomotor CMakeLists.txt

(Not sure if this file is required. It was added as an addition.)

```
menu "Zephyr"
source "Kconfig.zephyr"
endmenu

# module = SERVO_MOTOR
# module-str = SERVO_MOTOR
rsource "../drivers/pwm/Kconfig"
```

Servomotor Application main.c

```
#include <zephyr.h>
#include <sys/printk.h>
#include <device.h>
#include <drivers/pwm.h>

#define PWM_NODE DT_ALIAS(pwm_servo)

#if !DT_NODE_HAS_STATUS(PWM_NODE, okay)
#error "Unsupported board: pwm-servo devicetree alias is not defined or enabled"
#endif
 /*Unlike pulse width, the PWM period is not a critical parameter for*/
 /*motor control. 20 ms is commonly used.*/
#define PERIOD_USEC (20U * USEC_PER_MSEC)
#define STEP_USEC 100
#define MIN_PULSE_USEC 700
#define MAX_PULSE_USEC 2300

enum direction {
DOWN,
UP,
};
```


First we include the required header files for the PWM example. We need the PWM driver for this example. **DT_ALIAS()** gets a node identifier from /aliases. It returns a node identifier for the node which is aliased.

We then check if the node has the status property as "okay", else we need to throw an error.

We then define a few macros for time-period, step-duration, minimum-pulse duration, and maximum pulse duration (all in μ sec). We also define an enumerator for the direction of rotation (up or down).

```
void main(void)
{
    const struct device *pwm;
    uint32_t pulse_width = MIN_PULSE_USEC;
    enum direction dir = UP;
    int ret;

    printk("Servomotor control\n");

    pwm = DEVICE_DT_GET(PWM_NODE);

    if (!device_is_ready(pwm)) {
        printk("Error: PWM device %s is not ready\n", pwm->name);
        return;
}
```

In the **main()** function, we declare a pointer to the PWM device. As part of initial declaration, we define a local variable called *pulse_width* to hold the minimum pulse duration of a period. We also need a local variable for calculating the direction of rotation and the duration and one for holding the return value (useful for error checking).

DEVICE_DT_GET(): Obtain a pointer to a device object by node_id. If the PWM device is not ready to be used, then error message has to be displayed.


```
while (1)
{
    ret = pwm_pin_set_usec(pwm, 0, PERIOD_USEC, pulse_width, 0);
    if (ret < 0) {
        printk("Error %d: failed to set pulse width\n", ret);
        return;
        }

    if (dir == DOWN) {
        if (pulse_width <= MIN_PULSE_USEC) {
                        dir = UP;
                        pulse_width = MIN_PULSE_USEC;
        } else {
                pulse_width -= STEP_USEC;
        }
        } else {
                pulse_width += STEP_USEC;

        if (pulse_width >= MAX_PULSE_USEC) {
```

```
                        dir = DOWN;
                        pulse_width = MAX_PULSE_USEC;
         }
        }

        k_sleep(K_SECONDS(1));
        }
}
```

In an infinite loop, we call the **pwm_pin_set_usec()** API function which sets the period and pulse width for a single PWM output. It returns 0 if successful.

First we check if the direction of rotation is *DOWN*. If yes, then we check if the current pulse width is below the minimum pulse width for downward direction and if that is true, we reverse the direction to *UP*, else we decrement the pulse width.

Similar process is carried out if the direction is *UP* and we compare the current pulse width with the maximum pulse duration.

Problems faced in building the application

Here are some of the issues which may arise when trying to build the PWM driven servomotor.

- The first issue is with respect to the devicetree. The alias for the PWM node is defined but not enabled. **Solution:** Any other board apart from the default board for which the sample was written for, has to be enabled in the <BOARD>.overlay file using the statement *status = "okay";* for the new node being defined.
- The current issue and the next one are related as the next one is the consequence of the current error. The following error message shows shows up when servo motor example is built for the board *nrf52840_nrf52811*: **undefined reference to '__device_dts_ord_6' collect2: error: ld returned 1 exit status**. From previous learnings of *TMP*100, the possible but erroneous solution is to manually include the required drivers i.e. PWM driver. A fresh *Kconfig* might need to be written in the application directory along with some changes in *sample.yaml* and *CMakeLists.txt*. Although this does seem to erased the ordinal number error, it gives rise to numerous warnings and Kconfig errors, which is described in the next point.
- When the application is built after introducing the Kconfig file, the warning shown is **warning: the default selection PWM_LOG_LEVEL_INF (defined at subsys/logging/Kconfig.template.log_config:17, subsys/logging/Kconfig.template.log_config:17) of <choice> defined**

at subsys/logging/Kconfig.template.log_config:3 is not contained in the choice. Along with the above warning, 10 more similar warnings arise. The accompanying error message is **error: Aborting due to Kconfig warnings.** This also leads to *cmake* more errors.

Fixes

The appropriate overlay file is given below:

```
/ {
    aliases {
        pwm-servo = &pwm0;
    };
};

&pwm0 {
    status = "okay";
};
```

This is the fix need to be done in order to get the PWM driven DC motor (servo motor) working.

Output and observation

The requirements to view the final output are:

- nRF52840 Development Kit
- Jumper wires (3 no.s)
- *SG*90 Micro Servo

Power-up the board with a micro USB cable and make the connections as shown in the table below:

*SG*90	*nRF*52840*DK*
Vcc	5*V*
GND	GND
SIG	*P*0.13

Table 2: Pin connections for *SG*90 with *nRF*52840*DK*

*P*0.13 pin is used as the PWM signal pin for the servo motor, as it is specified in *nrf*52840*dk_nrf*52811.*dts* present in **zephyr/boards/arm/nrf52840dk_nrf52811** directory.

```
&pwm0 {
    status = "okay";
    ch0-pin = <13>;
};
```

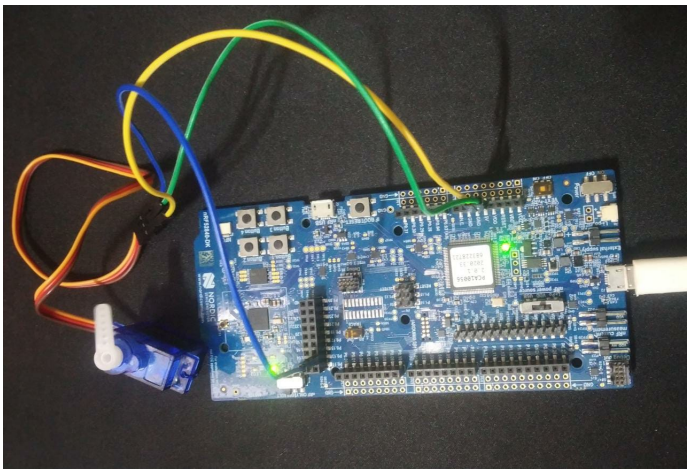

Figure 15: Servomotor connections

The image below shows the *SG*90 connected with *nRF*52840*DK*.
Run the following commands to view the output:

```
west build -p auto -b nrf52840dk_nrf52811 servo_motor
west flash --erase
```