Arduino ESP32 Timer and PWM Excerpt

Espressif Arduino ESP32 Library API V3.0

This excerpt from the original documentation[1] should be helpful for programming applications which use the timer or the PWM module of the ESP32. As many tutorials on the internet still use the old API version 2.x which is not compatible with version 3.0 a short migration section has been added, too. Please refer to the original documentation for further details[2]. To make it easier for beginners the layout has been adapted to the Arduino language reference.

Timer

timerBegin()

Description

This function is used to configure the timer. After successful setup the timer will automatically start.

Syntax

```
timerBegin(frequency)
```

Parameters

- `frequency`: select timer frequency in Hz.

Returns

A timer structure if configuration is successful. NULL if an error occurs.
Datatype: `hw_timer_t *`

timerAttachInterrupt()

Description

This function is used to attach interrupt to timer.

Syntax

```
timerAttachInterrupt(timer, onTimer)
```

Parameters

- `timer`: timer struct.
- `onTimer`: function to be called when interrupt is triggered.
  Datatype: `void ARDUINO_ISR_ATTR onTimer()`

Returns

Nothing

[1] https://docs.espressif.com/projects/arduino-esp32/en/latest/libraries.html#apis
[2] https://docs.espressif.com/projects/arduino-esp32/en/latest/migration_guides/2.x_to_3.0.html

timerAlarm()

Description

This function is used to configure alarm value and auto reload of the timer. Alarm is automatically enabled.

Syntax

```
timerAlarm(timer, alarm_value, autoreload, reload_count)
```

Parameters

- `timer`: timer struct.
- `alarm_value`: alarm value to generate event.
- `autoreload`: enabled/disabled auto reload.
- `reload_count`: number of auto reloads (0 = unlimited).

Returns

Nothing

timerStart()

Description

This function is used to start the counter of the timer.

Syntax

```
timerStart(timer)
```

Parameters

- `timer`: timer struct.

Returns

Nothing

timerStop()

Description

This function is used to stop counter of the timer.

Syntax

```
timerStop(timer)
```

Parameters

- `timer`: timer struct.

Returns

Nothing

PWM (LEDC)

ledcAttach()

Description

This function is used to setup LEDC pin with given frequency and resolution. LEDC channel will be selected automatically.

Syntax

```
ledcAttach(pin, frequency, resolution)
```

Parameters

- `pin`: select LEDC pin.
- `frequency`: select frequency of PWM.
- `resolution`: select resolution for LEDC channel. Range is 1-20 bits.

Returns

`true` if configuration was successful. `false` if an error occurred.

ledcWrite()

Description

This function is used to set duty for the LEDC pin.

Syntax

```
ledcWrite(pin, duty)
```

Parameters

- `pin`: select LEDC pin.
- `duty`: select duty to be set for selected LEDC pin.

Returns

`true` if setting duty is successful. `false` if an error occurred.

analogWrite()

Description

This function is used to write an analog value (PWM wave) on the pin. It is compatible with Arduinos analogWrite function.

Syntax

```
analogWrite(pin, value)
```

Parameters

- `pin`: select the GPIO pin.
- `value`: select the duty cycle of PWM. Range is from 0 (always off) to 255 (always on).

Returns

Nothing

Migration

Timer

Timer Overflow Interrupt

Version 2.x API

```
void IRAM_ATTR onTimer()
```

Version 3.0 API

```
void ARDUINO_ISR_ATTR onTimer()
```

Timer Configuration

Version 2.x API

```
timerBegin(num, divider, countUp)
```

Version 3.0 API

```
timerBegin(frequency)
```

Interrupt Attachment

Version 2.x API

```
timerAttachInterrupt(timer, onTimer, edge)
```

Version 3.0 API

```
timerAttachInterrupt(timer, onTimer)
```

Alarm Configuration

Version 2.x API

```
timerAlarmWrite(timer, alarm_value, autoreload)
timerAlarmEnable(timer)
```

Version 3.0 API

```
timerAlarm(timer, alarm_value, autoreload, reload_count)
```

PWM

PWM Configuration

Version 2.x API

```
ledcSetup(channel, frequency, resolution)
ledcAttachPin(pin, channel)
```

Version 3.0 API

```
ledcAttach(pin, frequency, resolution)
```

Duty Setting

Version 2.x API

```
ledcWrite(channel, duty)
```

Version 3.0 API

```
ledcWrite(pin, duty)
```

Example Code[3]

```
const int redLedPin = 33;
const int yellowLedPin = 15;

const int dutyCycle= 25;
const int pwmResolution = 8;
const int pwmFrequency = 5000;

const int ledFrequnecy = 1000000;
const int ledAlarm = 500000;

hw_timer_t* timer = NULL;
volatile int led_state = HIGH;

void ARDUINO_ISR_ATTR onTimer()
{
  digitalwrite(redLedPin, led_state);
  led_state = (led_state == HIGH) ? LOW : HIGH;
}

void setup()
{
  pinMode(redLedPin, OUTPUT);
  pinMode(yellowLedPin; OUTPUT)

  timer = timerBegin(ledFrequency);
  timerAttachInterrupt(timer, &onTimer);
  timerAlarm(timer, ledAlarm, true, 0);

  ledcAttach(yellowLedPin, pwmFrequency, pwmResolution);
  ledcWrite(yellowLedPin, dutyCycle);
}

void loop()
{
}
```

[3] For pedagogical reasons, this program will compile with errors.