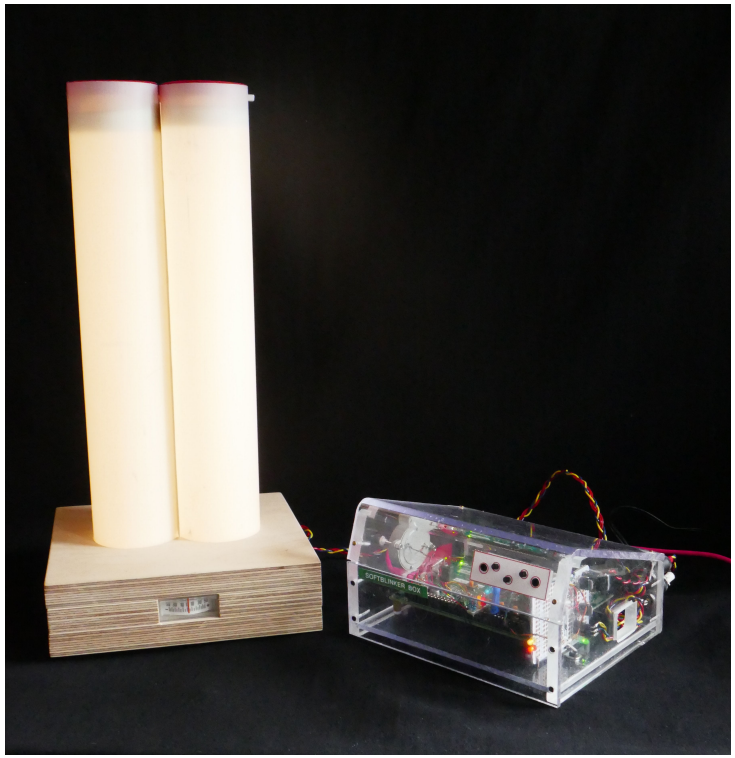
_Softblinker_PWM _notes.txt (_menu.pages)



```
// 2May2021 (in PWM_SOFTBLINKER.h in lib_pwm_softblinker)

typedef enum {  // Div by 4 ok  // Including zero
    steps_0012          =   12, //   13 values
    steps_0100          =  100, //  101 values
    steps_0256          =  256, //  257 values (*)
    steps_1000          = 1000, // 1001 values
    NUM_INTENSITY_STEPS = 4 // Those above
} intensity_steps_e;

#define DEFAULT_INTENSITY_STEPS  steps_1000
#define DEFAULT_PWM_FREQUENCY_HZ 222
#define PERIOD_MS_LIST { \
            SOFTBLINK_PERIOD_MAX_MS,       /* 10000 */ \
            SOFTBLINK_PERIOD_MAX_MS / 2,   /*  5000 */ \
            SOFTBLINK_PERIOD_MAX_MS / 10,  /*  1000 */ \
            SOFTBLINK_PERIOD_MIN_MS * 2.5, /*   500 */ \
            SOFTBLINK_PERIOD_MIN_MS}       /*   200 */

// 2May2021: (in _Softblinker_user_interface.xc)
typedef enum {
    state_red_LED_default,           // 0 beeep
    state_all_LEDs_stable_intensity, // 1 beeep beep .. plus some extra beeps on 0, 10 and 100% intensity ++
    state_red_LED_steps_0012,        // 2 beeep beep beep                    steps_0012
    state_red_LED_steps_0100,        // 3 beeep beep beep beep               steps_0100
    state_red_LED_steps_0256,        // 4 beeep beep beep beep beep          steps_0256 (steps_1000 is default)
    state_red_LED_half_range,        // 5 beeep beep beep beep beep beep beep
    state_all_LEDs_synched,          // 6 beeep beeeep + BLUE LED WHEN IN BARRIER
    NUM_RED_LED_STATES // ==7 those above
    //
} state_LED_views_e;

// 2May2021: (in _Softblinker_user_interface.xc)
// ------------------------------------------------------------------------------------------------------------------------
// BUTTONS          | LEFT                                | CENTER                                | RIGHT
// ------------------------------------------------------------------------------------------------------------------------
// pressed_now      | if also CENTER set red/right period | ...                                   | if also CENTER set yellow/left period
//                  | else next yellow/left period        | ...                                   | else next red/right period
// ------------------------------------------------------------------------------------------------------------------------
// released_now     | if steady light LEDs less, but      | if LEFT or RIGHT pressed_now handle it | if steady light LEDs more, but
//                  | if also RIGHT either halt the       | else swap phase and start black/full   | if also RIGHT either halt the
//                  | LED or below 1% down                |                                       | LED or below 1% down
// ------------------------------------------------------------------------------------------------------------------------
// pressed_for_long |                                     | Increase state_LED_views_e            | ...
// ========================================================================================================================
// pressed_for_long | LEFT       | if LEFT and RIGHT: clear to init state, but arbitrary starts    | RIGHT
// ------------------------------------------------------------------------------------------------------------------------
// Beeping as some pattern to distinguish button actions. If no beeping then that press has been disabled by previous press to
// avoid some present state becoming changed when not wanted. See inhibit_next_button_released_now_.. (some pressed_now must not
// be overwritten by released_now)
```