```
/**
 * @file ctc_laurel.js
 * @author Joey Damico
 * @date September 25, 2019
 * @summary CTC Controller Class for the Laurel Interlocking
 */

// Color Constants For Drawing Routes
const Empty = '#999999';
const Lined = '#75fa4c';
const Occupied = '#eb3323';


/**
 * Class is the Backend for the Laurel Interlocking This class is what
controlls the Laurel Interlocking,
 * it is sort of like a backen, but is the controller, this is what
makes all the train movements possible,
 * and the ReactJS Component class gets information from this class to
display the correct status of the
 * interlocking on the screen
 *
 * MEMBER VARIABLES
 * @member sw_1 -> Bool if Switch #1 is Reveresed or Not
 * @member sw_3 -> Bool if Switch #3 is Reveresed or Not
 * @member sw_7 -> Bool if Switch #7 is Reveresed or Not
 * @member sw_9 -> Bool if Switch #9 is Reveresed or Not
 * @member sw_11 -> Bool if Switch #11 is Reveresed or Not
 * @member sw_13 -> Bool if Switch #13 is Reveresed or Not
 *
 * @member sig_2w -> Bool if Signal #2w is Lined or Not
 * @member sig_4w -> Bool if Signal #4w is Lined or Not
 * @member sig_8w -> Bool if Signal #8w is Lined or Not
 * @member sig_10w -> Bool if Signal #10w is Lined or Not
 * @member sig_2e -> Bool if Signal #2e is Lined or Not
 * @member sig_4e -> Bool if Signal #4e is Lined or Not
 * @member sig_8e -> Bool if Signal #8e is Lined or Not
 * @member sig_12e -> Bool if Signal #12e is Lined or Not
 *
 * @member route_w_trk_1 = The west bound route for track #1
 * @member route_w_trk_2 = The west bound route for track #2
 * @member route_w_trk_3 = The west bound route for track #3
 * @member route_w_trk_4 = The west bound route for track #4
 * @member route_e_trk_1 = The east bound route for track #1
 * @member route_e_trk_2 = The east bound route for track #2
 * @member route_e_trk_3 = The east bound route for track #3
 * @member route_e_trk_4 = The east bound route for track #4
 *
 * @member routed_trk_1 = Bool if track #1 is routed or not
 * @member routed_trk_2 = Bool if track #2 is routed or not
```

```
 * @member routed_trk_3 = Bool if track #3 is routed or not
 * @member routed_trk_4 = Bool if track #4 is routed or not
 * @member trk_1_time = The time track #1 was occupied, used to know
when to clear the route
 * @member trk_2_time = The time track #2 was occupied, used to know
when to clear the route
 * @member trk_3_time = The time track #3 was occupied, used to know
when to clear the route
 * @member trk_4_time = The time track #4 was occupied, used to know
when to clear the route
 * @member trk_1_occupied = Bool if track #1 is occupied or not
 * @member trk_2_occupied = Bool if track #2 is occupied or not
 * @member trk_3_occupied = Bool if track #3 is occupied or not
 * @member trk_4_occupied = Bool if track #4 is occupied or not
 */
class CTC_Laurel {
    /**
     * constructor()
     * @summary The constructor for the CTC_Laurel class
     *
     * @description This will initialize all the member variables when
the program is started
     */
    constructor() {
        // Bools for the switches
        this.sw_1 = false;
        this.sw_3 = false;
        this.sw_7 = false;
        this.sw_9 = false;
        this.sw_11 = false;
        this.sw_13 = false;
        // Bools for the signals
        this.sig_2w = false;
        this.sig_4w = false;
        this.sig_8w = false;
        this.sig_10w = false;
        this.sig_6e = false;
        this.sig_12e = false;
        this.sig_8e = false;
        this.sig_4e = false;
        // Track routes
        this.route_w_trk_3 = null;
        this.route_w_trk_4 = null;
        this.route_w_trk_1 = null;
        this.route_w_trk_2 = null;
        this.route_e_trk_3 = null;
        this.route_e_trk_4 = null;
        this.route_e_trk_1 = null;
        this.route_e_trk_2 = null;
        // Used for routing and occupying the tracks
```

```
    this.routed_trk_1 = false;
    this.routed_trk_2 = false;
    this.routed_trk_3 = false;
    this.routed_trk_4 = false;
    this.occupied_trk_1 = false;
    this.occupied_trk_2 = false;
    this.occupied_trk_3 = false;
    this.occupied_trk_4 = false;
    this.trk_1_time = null;
    this.trk_2_time = null;
    this.trk_3_time = null;
    this.trk_4_time = null;
}
// ---- END constructor() ----

/**
 * get_train_route()
 * @summary Returns the route for the train at a given track
 *
 * @param direction, The direction the train is moving
 * @param track, The Track number of the train
 */
get_train_route(direction, track) {
    if (direction === "WEST") {
        if (track === "1") {
            return this.route_w_trk_1;
        }
        else if (track === "2") {
            return this.route_w_trk_2;
        }
        else if (track === "3") {
            return this.route_w_trk_3;
        }
        else {
            return this.route_w_trk_4;
        }
    }
    else {
        if (track === "1") {
            return this.route_e_trk_1;
        }
        else if (track === "2") {
            return this.route_e_trk_2;
        }
        else if (track === "3") {
            return this.route_e_trk_3;
        }
        else {
            return this.route_e_trk_4;
        }
```

```
        }
    }
    // ---- END get_train_route() ----

    /**
     * click_sig_2w()
     * @summary the function that is called when clicking the signal,
creates a route
     *
     * @description When the function is called it will determine if a
route can be created,
     * and if so what the route is and sets it based off of the switch
status
     *
     * @param next_block_1, The next block on Track #1
     * @param next_block_2, The next block on Track #2
     * @param next_block_3, The next block on Track #3
     */
    click_sig_2w(next_block_1, next_block_2, next_block_3) {
        if (this.sw_11 || this.sw_1) {
            return;
        }
        else if (!this.sw_7 && !this.sw_3) {
            if (this.sig_2w) {
                this.route_w_trk_1 = null;
                this.routed_trk_1 = false;
                this.sig_2w = false;
                return;
            }
            else {
                if (next_block_1 === Occupied || next_block_1 ===
Lined) {
                    alert("Cannot Line Route Because Conflict With
Next Block");
                    return;
                }
                this.route_w_trk_1 = "W_1_1__|__2_hx_laurel";
                this.routed_trk_1 = true;
                this.sig_2w = true;
            }
        }
        else if (!this.sw_7 && this.sw_3) {
            if (this.sig_2w) {
                this.route_w_trk_1 = null;
                this.routed_trk_1 = false;
                this.sig_2w = false;
            }
            else {
                if (next_block_3 === Occupied || next_block_3 ===
Lined) {
```

```
                    alert("Cannot Line Route Because Conflict With
Next Block");
                    return;
                }
                this.route_w_trk_1 = "W_1_3__|__3_hx_laurel";
                this.routed_trk_1 = true;
                this.sig_2w = true;
            }
        }
        else if (this.sw_7) {
            if (this.sig_2w) {
                this.route_w_trk_1 = null;
                this.routed_trk_1 = false;
                this.sig_2w = false;
                return;
            }
            else {
                if (next_block_2 === Occupied || next_block_2 ===
Lined) {
                    alert("Cannot Line Route Because Conflict With
Next Block");
                    return;
                }
                this.route_w_trk_1 = "W_1_2__|
__2_westSecaucus_laurel";
                this.routed_trk_1 = true;
                this.sig_2w = true;
            }
        }
    }
    // ---- END click_sig_2w() ----

    /**
     * click_sig_4w()
     * @summary the function that is called when clicking the signal,
creates a route
     *
     * @description When the function is called it will determine if a
route can be created,
     * and if so what the route is and sets it based off of the switch
status
     *
     * @param next_block_1, The next block on Track #1
     * @param next_block_2, The next block on Track #2
     * @param next_block_3, The next block on Track #3
     */
    click_sig_4w(next_block_1, next_block_2, next_block_3) {
        if (this.sw_13 || this.sw_7) {
            return;
        }
```

```
        else if (!this.sw_1) {
            if (this.sig_4w) {
                this.route_w_trk_2 = null;
                this.routed_trk_2 = false;
                this.sig_4w = false;
            }
            else {
                if (next_block_2 === Occupied || next_block_2 ===
Lined) {
                    alert("Cannot Line Route Because Conflict With
Next Block");
                    return;
                }
                this.route_w_trk_2 = "W_2_2__|
__2_westSecaucus_laurel";
                this.routed_trk_2 = true;
                this.sig_4w = true;
            }
        }
        else if (this.sw_1 && !this.sw_3) {
            if (this.sig_4w) {
                this.route_w_trk_2 = null;
                this.routed_trk_2 = false;
                this.sig_4w = false;
            }
            else {
                if (next_block_1 === Occupied || next_block_1 ===
Lined) {
                    alert("Cannot Line Route Because Conflict With
Next Block");
                    return;
                }
                this.route_w_trk_2 = "W_2_1__|__2_hx_laurel";
                this.routed_trk_2 = true;
                this.sig_4w = true;
            }
        }
        else if (this.sw_1 && this.sw_3) {
            if (this.sig_4w) {
                this.route_w_trk_2 = null;
                this.routed_trk_2 = false;
                this.sig_4w = false;
            }
            else {
                if (next_block_3 === Occupied || next_block_3 ===
Lined) {
                    alert("Cannot Line Route Because Conflict With
Next Block");
                    return;
                }
```

```
                this.route_w_trk_2 = "W_2_3__|__3_hx_laurel";
                this.routed_trk_2 = true;
                this.sig_4w = true;
            }
        }
    }
    // ---- END click_sig_4w() ----

    /**
     * click_sig_8w()
     * @summary the function that is called when clicking the signal,
creates a route
     *
     * @description When the function is called it will determine if a
route can be created,
     * and if so what the route is and sets it based off of the switch
status
     *
     * @param next_block_1, The next block on Track #1
     * @param next_block_2, The next block on Track #2
     * @param next_block_3, The next block on Track #3
     * @param next_block_4, The next block on Track #4
     */
    click_sig_8w(next_block_1, next_block_2, next_block_3,
next_block_4) {
        if (!this.sw_13) {
            if (this.sig_8w) {
                this.route_w_trk_4 = null;
                this.routed_trk_4 = false;
                this.sig_8w = false;
            }
            else {
                if (next_block_4 === Occupied || next_block_4 ===
Lined) {
                    alert("Cannot Line Route Because Conflict With
Next Block");
                    return;
                }
                this.route_w_trk_4 = "W_4_4__|
__4_westSecaucus_laurel";
                this.routed_trk_4 = true;
                this.sig_8w = true;
            }
        }
        else if (this.sw_13 && !this.sw_7 && !this.sw_1) {
            if (this.sig_8w) {
                this.route_w_trk_4 = null;
                this.routed_trk_4 = false;
                this.sig_8w = false;
            }
```

```
            else {
                if (next_block_2 === Occupied || next_block_2 ===
Lined) {
                    alert("Cannot Line Route Because Conflict With
Next Block");
                    return;
                }
                this.route_w_trk_4 = "W_4_2__|
__2_westSecaucus_laurel";
                this.routed_trk_4 = true;
                this.sig_8w = true;
            }
        }
        else if (this.sw_13 && !this.sw_7 && this.sw_1 && !this.sw_3)
{
            if (this.sig_8w) {
                this.route_w_trk_4 = null;
                this.routed_trk_4 = false;
                this.sig_8w = false;
            }
            else {
                if (next_block_1 === Occupied || next_block_1 ===
Lined) {
                    alert("Cannot Line Route Because Conflict With
Next Block");
                    return;
                }
                this.route_w_trk_4 = "W_4_1__|__2_hx_laurel";
                this.routed_trk_4 = true;
                this.sig_8w = true;
            }
        }
        else if (this.sw_13 && !this.sw_7 && this.sw_1 && this.sw_3) {
            if (this.sig_8w) {
                this.route_w_trk_4 = null;
                this.routed_trk_4 = false;
                this.sig_8w = false;
            }
            else {
                if (next_block_3 === Occupied || next_block_3 ===
Lined) {
                    alert("Cannot Line Route Because Conflict With
Next Block");
                    return;
                }
                this.route_w_trk_4 = "W_4_3__|__3_hx_laurel";
                this.routed_trk_4 = true;
                this.sig_8w = true;
            }
        }
```

```
    }
    // ---- END click_sig_8w() ----

    /**
     * click_sig_10w()
     * @summary the function that is called when clicking the signal,
creates a route
     *
     * @description When the function is called it will determine if a
route can be created,
     * and if so what the route is and sets it based off of the switch
status
     *
     * @param next_block_1, The next block on Track #1
     * @param next_block_2, The next block on Track #2
     * @param next_block_3, The next block on Track #3
     */
    click_sig_10w(next_block_1, next_block_2, next_block_3) {
        if (!this.sw_11 && !this.sw_3) {
            if (this.sig_10w) {
                this.route_w_trk_3 = null;
                this.routed_trk_3 = false;
                this.sig_10w = false;
            }
            else {
                if (next_block_3 === Occupied || next_block_3 ===
Lined) {
                    alert("Cannot Line Route Because Conflict With
Next Block");
                    return;
                }
                this.route_w_trk_3 = "W_3_3__|__3_hx_laurel";
                this.routed_trk_3 = true;
                this.sig_10w = true;
            }
        }
        else if (this.sw_11 && !this.sw_7 && !this.sw_3 && !this.sw_1)
{
            if (this.sig_10w) {
                this.route_w_trk_3 = null;
                this.routed_trk_3 = false;
                this.sig_10w = false;
            }
            else {
                if (next_block_1 === Occupied || next_block_1 ===
Lined) {
                    alert("Cannot Line Route Because Conflict With
Next Block");
                    return;
                }
```

```
                this.route_w_trk_3 = "W_3_1__|__1_hx_laurel";
                this.routed_trk_3 = true;
                this.sig_10w = true;
            }
        }
        else if (this.sw_11 && this.sw_7 && !this.sw_1) {
            if (this.sig_10w) {
                this.route_w_trk_3 = null;
                this.routed_trk_3 = false;
                this.sig_10w = false;
            }
            else {
                if (next_block_2 === Occupied || next_block_2 ===
Lined) {
                    alert("Cannot Line Route Because Conflict With
Next Block");
                    return;
                }
                this.route_w_trk_3 = "W_3_2__|
__2_westSecaucus_laurel";
                this.routed_trk_3 = true;
                this.sig_10w = true;
            }
        }
    }
    // ---- END click_sig_10w() ----

    /**
     * click_sig_6e()
     * @summary the function that is called when clicking the signal,
creates a route
     *
     * @description When the function is called it will determine if a
route can be created,
     * and if so what the route is and sets it based off of the switch
status
     *
     * @param next_block_1, The next block on Track #1
     * @param next_block_2, The next block on Track #2
     * @param next_block_3, The next block on Track #3
     * @param next_block_4, The next block on Track #4
     */
    click_sig_6e(next_block_1, next_block_2, next_block_3,
next_block_4) {
        if (!this.sw_3 && !this.sw_11) {
            if (this.sig_6e) {
                this.route_e_trk_3 = null;
                this.routed_trk_3 = false;
                this.sig_6e = false;
            }
```

```
            else {
                if (next_block_3 === Occupied || next_block_3 ===
Lined) {
                    alert("Cannot Line Route Because Conflict With
Next Block");
                    return;
                }
                this.route_e_trk_3 = "E_3_3__|__3_laurel_westEnd";
                this.routed_trk_3 = true;
                this.sig_6e = true;
            }
        }
        else if (this.sw_3 && !this.sw_1 && !this.sw_7) {
            if (this.sig_6e) {
                this.route_e_trk_3 = null;
                this.routed_trk_3 = false;
                this.sig_6e = false;
            }
            else {
                if (next_block_1 === Occupied || next_block_1 ===
Lined) {
                    alert("Cannot Line Route Because Conflict With
Next Block");
                    return;
                }
                this.route_e_trk_3 = "E_3_1__|__1_laurel_westEnd";
                this.routed_trk_3 = true;
                this.sig_6e = true;
            }
        }
        else if (this.sw_3 && this.sw_1 && !this.sw_7 && !this.sw_13)
{
            if (this.sig_6e) {
                this.route_e_trk_3 = null;
                this.routed_trk_3 = false;
                this.sig_6e = false;
            }
            else {
                if (next_block_2 === Occupied || next_block_2 ===
Lined) {
                    alert("Cannot Line Route Because Conflict With
Next Block");
                    return;
                }
                this.route_e_trk_3 = "E_3_2__|__2_laurel_westEnd";
                this.routed_trk_3 = true;
                this.sig_6e = true;
            }
        }
        else if (this.sw_3 && this.sw_1 && !this.sw_7 && this.sw_13) {
```

```
            if (this.sig_6e) {
                this.route_e_trk_3 = null;
                this.routed_trk_3 = false;
                this.sig_6e = false;
            }
            else {
                if (next_block_4 === Occupied || next_block_4 ===
Lined) {
                    alert("Cannot Line Route Because Conflict With
Next Block");
                    return;
                }
                this.route_e_trk_3 = "E_3_4__|__4_laurel_westEnd";
                this.routed_trk_3 = true;
                this.sig_6e = true;
            }
        }
    }
    // ---- END click_sig_6e() ----

    /**
     * click_sig_12e()
     * @summary the function that is called when clicking the signal,
creates a route
     *
     * @description When the function is called it will determine if a
route can be created,
     * and if so what the route is and sets it based off of the switch
status
     *
     * @param next_block_1, The next block on Track #1
     * @param next_block_2, The next block on Track #2
     * @param next_block_3, The next block on Track #3
     * @param next_block_4, The next block on Track #4
     */
    click_sig_12e(next_block_1, next_block_2, next_block_3,
next_block_4) {
        if (this.sw_3 || this.sw_7) {
            return;
        }
        else if (!this.sw_1 && !this.sw_11) {
            if (this.sig_12e) {
                this.route_e_trk_1 = null;
                this.routed_trk_1 = false;
                this.sig_12e = false;
            }
            else {
                if (next_block_1 === Occupied || next_block_1 ===
Lined) {
                    alert("Cannot Line Route Because Conflict With
```

```
Next Block");
                    return;
                }
                this.route_e_trk_1 = "E_1_1__|__1_laurel_westEnd";
                this.routed_trk_1 = true;
                this.sig_12e = true;
            }
        }
        else if (!this.sw_1 && this.sw_11) {
            if (this.sig_12e) {
                this.route_e_trk_1 = null;
                this.routed_trk_1 = false;
                this.sig_12e = false;
            }
            else {
                if (next_block_3 === Occupied || next_block_3 ===
Lined) {
                    alert("Cannot Line Route Because Conflict With
Next Block");
                    return;
                }
                this.route_e_trk_1 = "E_1_3__|__3_laurel_westEnd";
                this.routed_trk_1 = true;
                this.sig_12e = true;
            }
        }
        else if (this.sw_1 && !this.sw_13) {
            if (this.sig_12e) {
                this.route_e_trk_1 = null;
                this.routed_trk_1 = false;
                this.sig_12e = false;
            }
            else {
                if (next_block_2 === Occupied || next_block_2 ===
Lined) {
                    alert("Cannot Line Route Because Conflict With
Next Block");
                    return;
                }
                this.route_e_trk_1 = "E_1_2__|__2_laurel_westEnd";
                this.routed_trk_1 = true;
                this.sig_12e = true;
            }
        }
        else if (this.sw_1 && this.sw_13) {
            if (this.sig_12e) {
                this.route_e_trk_1 = null;
                this.routed_trk_1 = false;
                this.sig_12e = false;
            }
```

```
            else {
                if (next_block_4 === Occupied || next_block_4 ===
Lined) {
                    alert("Cannot Line Route Because Conflict With
Next Block");
                    return;
                }
                this.route_e_trk_1 = "E_1_4__|__4_laurel_westEnd";
                this.routed_trk_1 = true;
                this.sig_12e = true;
            }
        }
    }
    // ---- END click_sig_12e() ----

    /**
     * click_sig_4e()
     * @summary the function that is called when clicking the signal,
creates a route
     *
     * @description When the function is called it will determine if a
route can be created,
     * and if so what the route is and sets it based off of the switch
status
     *
     * @param next_block_1, The next block on Track #1
     * @param next_block_2, The next block on Track #2
     * @param next_block_3, The next block on Track #3
     * @param next_block_4, The next block on Track #4
     */
    click_sig_4e(next_block_1, next_block_2, next_block_3,
next_block_4) {
        if (this.sw_1) {
            return;
        }
        else if (!this.sw_7 && !this.sw_13) {
            if (this.sig_4e) {
                this.route_e_trk_2 = null;
                this.routed_trk_2 = false;
                this.sig_4e = false;
            }
            else {
                if (next_block_2 === Occupied || next_block_2 ===
Lined) {
                    alert("Cannot Line Route Because Conflict With
Next Block");
                    return;
                }
                this.route_e_trk_2 = "E_2_2__|__2_laurel_westEnd";
                this.routed_trk_2 = true;
```

```
                this.sig_4e = true;
            }
        }
        else if (!this.sw_7 && this.sw_13) {
            if (this.sig_4e) {
                this.route_e_trk_2 = null;
                this.routed_trk_2 = false;
                this.sig_4e = false;
            }
            else {
                if (next_block_4 === Occupied || next_block_4 ===
Lined) {
                    alert("Cannot Line Route Because Conflict With
Next Block");
                    return;
                }
                this.route_e_trk_2 = "E_2_4__|__4_laurel_westEnd";
                this.routed_trk_2 = true;
                this.sig_4e = true;
            }
        }
        else if (this.sw_7 && !this.sw_11) {
            if (this.sig_4e) {
                this.route_e_trk_2 = null;
                this.routed_trk_2 = false;
                this.sig_4e = false;
            }
            else {
                if (next_block_1 === Occupied || next_block_1 ===
Lined) {
                    alert("Cannot Line Route Because Conflict With
Next Block");
                    return;
                }
                this.route_e_trk_2 = "E_2_1__|__1_laurel_westEnd";
                this.routed_trk_2 = true;
                this.sig_4e = true;
            }
        }
        else if (this.sw_7 && this.sw_11) {
            if (this.sig_4e) {
                this.route_e_trk_2 = null;
                this.routed_trk_2 = false;
                this.sig_4e = false;
            }
            else {
                if (next_block_3 === Occupied || next_block_3 ===
Lined) {
                    alert("Cannot Line Route Because Conflict With
Next Block");
```

```
                    return;
                }
                this.route_e_trk_2 = "E_2_3__|__3_laurel_westEnd";
                this.routed_trk_2 = true;
                this.sig_4e = true;
            }
        }
    }
    // ---- END click_sig_4e() ----

    /**
     * click_sig_8e()
     * @summary the function that is called when clicking the signal,
creates a route
     *
     * @description When the function is called it will determine if a
route can be created,
     * and if so what the route is and sets it based off of the switch
status
     *
     * @param next_block_4, The next block on Track #4
     */
    click_sig_8e(next_block_4) {
        if (this.sw_13) {
            return;
        }
        else {
            if (this.sig_8e) {
                this.route_e_trk_4 = null;
                this.routed_trk_4 = false;
                this.sig_8e = false;
            }
            else {
                if (next_block_4 === Occupied || next_block_4 ===
Lined) {
                    alert("Cannot Line Route Because Conflict With
Next Block");
                    return;
                }
                this.route_e_trk_4 = "E_4_4__|__4_laurel_westEnd";
                this.routed_trk_4 = true;
                this.sig_8e = true;
            }
        }
    }
    // ---- END click_sig_8e() ----

    /**
     * set_trk_1_occupied()
     * @summary Sets track #1 as occupied
```

```
     *
     * @param n_state, The new state of the track
     * This was used to test, and never removed passing the state as a
paramemter, which is not needed anymore
     */
    set_trk_1_occupied(n_state) {
        if (n_state === true) {
            this.occupied_trk_1 = n_state;
            this.routed_trk_1 = false;
            this.trk_1_time = new Date().getTime() / 1000;
        }
        else {
            console.log("ERROR");
        }
    }
    // ---- END set_trk_1_occupied() ----

    /**
     * set_trk_2_occupied()
     * @summary Sets track #2 as occupied
     *
     * @param n_state, The new state of the track
     * This was used to test, and never removed passing the state as a
paramemter, which is not needed anymore
     */
    set_trk_2_occupied(n_state) {
        if (n_state === true) {
            this.occupied_trk_2 = n_state;
            this.routed_trk_2 = false;
            this.trk_2_time = new Date().getTime() / 1000;
        }
        else {
            console.log("ERROR");
        }
    }
    // ---- END set_trk_2_occupied() ----

    /**
     * set_trk_3_occupied()
     * @summary Sets track #3 as occupied
     *
     * @param n_state, The new state of the track
     * This was used to test, and never removed passing the state as a
paramemter, which is not needed anymore
     */
    set_trk_3_occupied(n_state) {
        if (n_state === true) {
            this.occupied_trk_3 = n_state;
            this.routed_trk_3 = false;
            this.trk_3_time = new Date().getTime() / 1000;
```

```
        }
        else {
            console.log("ERROR");
        }
    }
    // ---- END set_trk_3_occupied() ----

    /**
     * set_trk_4_occupied()
     * @summary Sets track #4 as occupied
     *
     * @param n_state, The new state of the track
     * This was used to test, and never removed passing the state as a
paramemter, which is not needed anymore
     */
    set_trk_4_occupied(n_state) {
        if (n_state === true) {
            this.occupied_trk_4 = n_state;
            this.routed_trk_4 = false;
            this.trk_4_time = new Date().getTime() / 1000;
        }
        else {
            console.log("ERROR");
        }
    }
    // ---- END set_trk_4_occupied() ----

    /**
     * can_clear()
     * @summary Checks if a track could be cleared, meaning a train is
no longer in the interlocking
     *
     * @description Check both track if a train has been in the
interlocking for more then 4 seconds, if so it
     * clears that track
     */
    can_clear() {
        // Get the current time
        let current_time = new Date().getTime() / 1000;
        // Track #1
        if (current_time - this.trk_1_time > 4 && current_time -
this.trk_1_time< 100000) {
            this.sig_2w = false;
            this.sig_12e = false;

            this.route_w_trk_1 = null;
            this.route_e_trk_1 = null;
            this.routed_trk_1 = false;

            this.occupied_trk_1 = false;
```

```
            this.trk_1_time = null;
        }
        // Track #2
        if (current_time - this.trk_2_time > 4 && current_time -
this.trk_2_time< 100000) {
            this.sig_4w = false;
            this.sig_4e = false;

            this.route_w_trk_2 = null;
            this.route_e_trk_2 = null;
            this.routed_trk_2 = false;

            this.occupied_trk_2 = false;
            this.trk_2_time = null;
        }
        // Track #3
        if (current_time - this.trk_3_time > 4 && current_time -
this.trk_3_time< 100000) {
            this.sig_10w = false;
            this.sig_6e = false;

            this.route_w_trk_3 = null;
            this.route_e_trk_3 = null;
            this.routed_trk_3 = false;

            this.occupied_trk_3 = false;
            this.trk_3_time = null;
        }
        // Track #4
        if (current_time - this.trk_4_time > 4 && current_time -
this.trk_4_time< 100000) {
            this.sig_8w = false;
            this.sig_8e = false;

            this.route_w_trk_4 = null;
            this.route_e_trk_4 = null;
            this.routed_trk_4 = false;

            this.occupied_trk_4 = false;
            this.trk_4_time = null;
        }
    }
    // ---- END can_clear() ----

    /**
     * get_routes()
     * @summary Gets all the routes from the interlocking
     *
     * @returns An Array holding every route variable from the
interlocking
```

```
     */
    get_routes() {
        let routes = [
            this.route_e_trk_4, this.route_e_trk_3,
            this.route_e_trk_1, this.route_e_trk_2,
            this.route_w_trk_4, this.route_w_trk_3,
            this.route_w_trk_2, this.route_w_trk_1,
        ];

        return routes;
    }
    // ---- END get_routes() ----

    /**
     * @summary Function to throw switch #1 in the interlocking
     *
     * The function sets the status of the switch, whether it is is
the normal possition
     * of reversed, (True = Reversed / False = Normal)
     */
    throw_sw_1() {
        if (this.sw_1 === false) {
            this.sw_1 = true;
        }
        else {
            this.sw_1 = false;
        }
    }
    // ---- END throw_sw_1() ----

    /**
     * @summary Funtion to throw switch #3 in the interlocking
     *
     * The function sets the status of the switch, whether it is is
the normal possition
     * of reversed, (True = Reversed / False = Normal)
     */
    throw_sw_3() {
        if (this.sw_3 === false) {
            this.sw_3 = true;
        }
        else {
            this.sw_3 = false;
        }
    }
    // ---- END throw_sw_3() ----

    /**
     * @summary Funtion to throw switch #7 in the interlocking
     *
```

```
     * The function sets the status of the switch, whether it is is
the normal possition
     * of reversed, (True = Reversed / False = Normal)
     */
    throw_sw_7() {
        if (this.sw_7 === false) {
            this.sw_7 = true;
        }
        else {
            this.sw_7 = false;
        }
    }
    // ---- END throw_sw_7() ----

    /**
     * @summary Funtion to throw switch #9 in the interlocking
     *
     * The function sets the status of the switch, whether it is is
the normal possition
     * of reversed, (True = Reversed / False = Normal)
     */
    throw_sw_9() {
        if (this.sw_9 === false) {
            this.sw_9 = true;
        }
        else {
            this.sw_9 = false;
        }
    }
    // ---- END throw_sw_9() ----

    /**
     * @summary Funtion to throw switch #11 in the interlocking
     *
     * The function sets the status of the switch, whether it is is
the normal possition
     * of reversed, (True = Reversed / False = Normal)
     */
    throw_sw_11() {
        if (this.sw_11 === false) {
            this.sw_11 = true;
        }
        else {
            this.sw_11 = false;
        }
    }
    // ---- END throw_sw_11() ----

    /**
     * @summary Funtion to throw switch #13 in the interlocking
```

```
     *
     * The function sets the status of the switch, whether it is is
the normal possition
     * of reversed, (True = Reversed / False = Normal)
     */
    throw_sw_13() {
        if (this.sw_13 === false) {
            this.sw_13 = true;
        }
        else {
            this.sw_13 = false;
        }
    }
    // ---- END throw_sw_13() ----

    /**
     * get_interlocking_status()
     * @summary returns the status of the interlocking that would be
needed by the ReactJS Components
     *
     * @description All the information that is returned here is what
is needed by the ReactJS Component
     * for the interlocking that is need to draw the interlocking to
the screen
     *
     * @returns Object with the status of the interlocking
     */
    get_interlocking_status() {
        let status = {
            sw_1: this.sw_1,
            sw_3: this.sw_3,
            sw_7: this.sw_7,
            sw_9: this.sw_9,
            sw_11: this.sw_11,
            sw_13: this.sw_13,
            routed_1: this.routed_trk_1,
            routed_2: this.routed_trk_2,
            routed_3: this.routed_trk_3,
            routed_4: this.routed_trk_4,
            occupied_1: this.occupied_trk_1,
            occupied_2: this.occupied_trk_2,
            occupied_3: this.occupied_trk_3,
            occupied_4: this.occupied_trk_4,
            routes: this.get_routes()
        }

        return status;
    }
    // ---- END get_interlocking_status() ----
}
```

```
// This is required when using ReactJS
export default CTC_Laurel;
```