306 Car Command Interface Guide

This README describes the structure, usage, and behavior of commands used to control the ECE306 car project. The interface uses simple serial string commands to trigger movement or behavior.

General Command Format

Each command follows this format:

```
^<4digitPIN><Command_Letter><Modifier#>
```

- `^`: **Required prefix** for all commands.
- `4digitPIN`: Security pin required for command execution. **Current PIN: 3061**
- `Command_Letter`: Character specifying which action to perform.
- `Modifier#`: A single digit (0-9) used to adjust distance, direction, or variant of behavior.

> Example: `^3061F9` moves the car forward with maximum intensity.

To change the required command pin (In `main.c`):

```
char pin[] = "3061"; // Change this line in code
```

Command List

Basic Movement Commands

Command	Letter	Description	Modifier Meaning
Forward	`F`	Move forward	0 = tiny movement, 9 = max distance
Back	`B`	Move backward	0 = tiny movement, 9 = max distance
Right	`R`	Turn right	0-9 scale for rotation
Left	`L`	Turn left	0-9 scale for rotation

Example (FORWARDS):

```
case FORWARDS:
    move_test(5); // Move forward (5 = both wheels forward)
    // Modifier# is used to control setTime, which affects distance
    if (run_time > setTime){
        run_time_flag = 0;
        motor_off();
```

```
            run_time = 0;
            commanding_send = WAIT;
            movement = 0;
        }
        break;
```

Specialized Commands

A - ARRIVED

Used to indicate the car has reached a specific pad.

```
^3061A5 → Display "Arrived at Pad 5"
```

Modifier represents the pad number (0–9).

X - INTERCEPT (Line Following Start)

Starts the black line intercept logic after an initial movement (based on modifier).

Each `setTime` triggers a different action in `initialMovementBL()`:

Modifier	Behavior
0	No movement before intercept begins
1	Small forward movement before intercept
2	Large forward movement before intercept
3	Small reverse movement before intercept
4	Large reverse movement before intercept
5	180-degree spin **(SEE WARNING BELOW)**
6	Small left + forward
7	Large left + forward
8	Small right + forward
9	Large right + forward

Example:

```
case INTERCEPT:
    HexToBCD(ADC_Left_Detect);
    adc_line(2,2);
    HexToBCD(ADC_Right_Detect);
    adc_line(3,2);
```

```
        display_changed = TRUE;
        initialMovementBL();
        break;
```

`i` - IRCONF

Displays or calculates black line IR detection thresholds. Modifier selects configuration method:

Modifier	Function
0	Use default hardcoded values (500–600)
1	Calculate thresholds from average sensor values

```
    case 0:
        black_low = 500;
        black_high = 600;
        dispPrint("BL=500", '2');
        dispPrint("BL=600", '3');
        break;
    case 1:
        black_low = (ADC_Left_Detect+ADC_Right_Detect)/2 - 50;
        black_high = (ADC_Left_Detect+ADC_Right_Detect)/2 + 50;
        // Values are printed but not used in line logic
        break;
```

`e` - EXIT

Sends the car away from the line-following course (e.g., end of run).

- Modifier 9 is recommended to go far off the course.
- Code multiplies `setTime` by 10 for longer duration.

```
    case EXIT:
        new_forward();
        if (run_time > setTime*10){
            motor_off();
            commanding_send = WAIT;
        }
        break;
```

`s` - SLOWRIGHT / ARCH

Used for a complex motion sequence to align with the track after Pad 8. Modifier is ignored. Hardcoded durations control each movement in a state machine:

```
void arch_movement(void){
    case 1: // FORWARD #1
        if(arch_counter > 20 * 2.4){ // 2.4 sec
            Off_Case();
            archState = 0;
            nextState = 2;
        }
    // Other states follow a similar wait → move pattern
}
```

`d` - DIRECTION

Created for debug purposes. Modifier# determines movement direction, NOT movement duration. Movement Duration is always set using the `setTime` variable. This was NOT in the original D-Day code. This was added recently for debugging unexpected movmement issues.

```
case 'd':
    direction = (int)iot_TX_buf[response_parse+1] - '0';
    setTime = 10; // All movements last 10s
    commanding_send = DIRECTION;
```

Macro to Character Map

Macro	Character	Notes
FORWARDS	`F`	Forward
BACK	`B`	Backward
RIGHT	`R`	Turn right
LEFT	`L`	Turn left
INTERCEPT	`X`	Black line intercept
ARRIVED	`A`	Display arrival pad
IRCONF	`i`	IR display/test config
EXIT	`e`	Drive far away from track
SLOWRIGHT	`s`	ARCH pad maneuver (`modificaton#` is ignored)
DIRECTION	`d`	Debug custom motor path

Notes & Tips

- Always start commands with `^`
- You won't see the car move with modifier `0` on most motion commands

- Use `d` (DIRECTION) to manually verify directions and motor behavior. Only meant to be used for debugging, not D-Day
- Avoid `X5` unless IR intercept spin direction is corrected
- ARCH (`s`) and IRCONF (`i`) are timing/state based; their modifier is not critical or is ignored

---

Special Notes & Warnings

- **180 Spin in INTERCEPT (Modifier 5):** Currently spins the car in the wrong direction. Instead of aligning with the circle leg of the track, it turns into the line and follows it backwards. Use with caution, and only after modifying the spin direction logic post-black line detection.

- **DIRECTION Command:** This command was introduced solely for debugging. It is not part of the original D-Day sequence. The `modifier#` specifies which movement pattern to use, but **the movement time is always hardcoded to 10 seconds via `setTime`**. Useful for troubleshooting unexpected direction behavior.

- **IRCONF Values Ignored:** While the IRCONF command lets you set `black_low` and `black_high` thresholds (either to defaults or sampled), these values are not used in the actual IR line-following logic. The original hardcoded values performed well enough in practice.

- **EXIT Command Usage:** Although any modifier value can be used, the intention of the `EXIT` command is to move far away from the line-following area after completing a run. Therefore, high values like `^3061e9` are recommended.