# **CTS Relay Scheduler Design Document**

**Date Requested:** 04/14/2021

**Requested By:** Component Tinning Services - Ryan

**Approved By:** Ryan & Mike

**Team:** Tyler

**Project Description:**

CTS would like to introduce the ability for customers to setup custom schedules for a 4-way USB Relay device. They would like a custom and standalone desktop application that would allow for scheduling of each individual relay and to automatically turn the relays on and off at the appropriate times.

The schedules for each relay need to be customizable and apply to each relay independently. Some additional features they would like to see would include having the application automatically locate the relay device since there will always only be one on a machine, the ability to force separate relays on and off, the relay names should be editable, and it should remember its last state on a system reboot.

**Requirements**

* On software load, the device address for the USB device is located automatically
* Each relay has a configurable schedule for each day of the week and at all hours
* Relays automatically turn on and off based on the schedule
* Users can force a relay on or off and it will ignore the schedule
* An indicator is present on the main form to reflect current relay status for each relay
* Relay names can be edited by users
* Last relay states are remembered when computer reboots
* Software supports generic FTDI devices and Tctec Relay devices

Some additional desired features would include:

* Configuration import/export

**Design & Implementation**

This program will consist of two forms, a primary form for the device and a form for configuring the schedules of the relays. The program will provide users with the ability to force individual relays on, off, or to follow the schedule, as well as have feedback that indicates each relay’s current state.

All USB relay devices will inherit the RelayDeviceBase class. The base class provides virtual functions for getting and setting relay state as well as getting a serial number. All child classes need to provide implementations of these three functions. The base class also handles all of the relay scheduling for the relay board.

The software will support two implementations of USB relay devices. It will support any 4 relay device with an FTDI chipset and the Tctec 4 way USB relays. Each device will have its own class and will focus on providing the interface between the software and the device itself. Initializing the device, getting and setting relays, opening and closing the port, and getting the serial number will all be contained within the device classes.

The primary form will display the device address, the four relays for the device, and the state of each. It will also have options for forcing each relay on and off as well as a button to edit the schedule. The relay names will be editable through the Edit menu button and textboxes that will change the name when they lose focus. Users can also export and import settings files from other machines through the File menu. This will port all settings including schedules, currently forced states, and relay names. This form will look like the image below:

Graphical user interface, application

Description automatically generated

The Device Address field populates automatically on software start if a USB Relay device is located. If a device is not located then the form will still load but a message will appear indicating that a device was not found. Once the device is connected, the “Refresh” button can be used to reconnect to the device and it will populate the Device Address field when connected.

Each relay can be forced on or off or allowed to follow the schedule through the Force On/Off checkbox in-line with the respective relay. This is a three state checkbox so if checked then it will force the relay on, intermediate will force it off, and unchecked will have it follow the schedule. If a user attempts to force a relay on or off but it is unable to set it to the appropriate state, it will revert to using the schedule. The current state of the checkboxes is saved in the program settings under RelayForcedStates so if a user closes the software with a relay forced on, it will be forced on again once the software is launched again. When the relays are forced, they will ignore the schedules assigned until they are unchecked.

The Status indicator for each relay is refreshed every 5 seconds and relies on the RelayDeviceBase class for maintaining the correct state. To limit the chances of a status being incorrect, particularly on startup, the software sends a command to turn each relay on or off based on the force relay states saved in settings. If it was forced on, then it will try to turn it on and vice versa for relays that are forced off or set to use the schedule. This allows the software to ensure it starts with the correct state being known. The Relay devices do not provide the ability to query the current state so maintaining state in the device class was the best option.

The schedule button for each relay will open the RelaySetupForm which displays the current relay name and schedule in editable fields. This form is pictured below:

Graphical user interface, table

Description automatically generated with medium confidence

The relay name is populated based on which button is clicked and will indicate the relay that the schedule is assigned to. Each schedule is independent to the relay it was saved for. However, the “Apply to All Schedules” checkbox will save the current schedule over each other schedule when saved. This can be used to populate a base schedule for all of the relays if there is a standard time they will all be running.

The Enable checkboxes, Start Time, and End Time fields are all populated from the RelaySchedules settings. This setting contains a list of schedules with each index representing an individual relay (Relay 1 will be index 0). Each RelaySchedule contains seven Schedules to indicate each day. The Schedules are stores within a list in each RelaySchedule class and the indices are based off the DayOfWeek enum so Sunday will be 0 and Saturday will be 6.

Users can select which days will be enabled and the start and end time of each day. A schedule will not run for a day unless it is enabled. The start times must also be before the end time or an error will appear when the user attempts to save the schedule. Once the “Save” button is clicked, the form will validate each input to ensure it is correct and then save it to the appropriate relay(s). Once closed, the device will immediately begin following the new schedule unless forced on or off.

The schedule is checked approximately every five seconds through the relayScheduleTimer. The device iterates through each schedule and checks the current day of the week against the corresponding day in the Schedules list then checks the TimeOfDay for the current time, start time, and end time. If the current time falls within the current day’s start and end times then it will tell the device to turn on if it is not already on. If it falls out of the schedule then it will tell it to turn off if not already off.

**Validation & Testing**

The test cases below should cover each requirement and any additional testing requirements to ensure the software meets at least the minimum viable product standard.

**Software Start**

* Start the software with a device connected
  + Ensure the software loads and the device address is automatically populated
* Start the software with a device disconnected
  + Ensure a message appears indicating no device was found and that the device address field is not populated
  + Ensure that relays will not turn on

**MainForm**

* Start the software without a device connected
  + Connect the device after the software is loaded
  + Click “Refresh”
  + Ensure the device connects and the device address populates with the correct address
* Click Force On for each relay
  + Ensure each relay turns on
* Click Force Off for each relay
  + Ensure each relay turns off
* Click Force On/Off so it is unchecked for each relay
  + Ensure each relay begins to follow the correct schedule
* Toggle each relay and ensure the Status indicator remains correct
* Click the schedule button for each relay
  + Ensure the RelaySetupForm appears with the correct schedule name and correct schedule
* Check each relay to be forced on
  + Reload the program, ensure the relays are still forced on
  + Repeat with each forced off
  + Repeat with each following the schedule
  + Repeat with a mix

**RelaySetupForm**

* Open the schedule for a relay
  + Ensure the Schedule form displays the correct relay name
  + Ensure the schedule displayed is correct if one was set
* Enable each day and set a start and end time
* Repeat with end time before the start time
  + Ensure the form does not save
* Correct with start time before end time
  + Ensure the form saves
* Reload the form and ensure the new schedule loads correctly
* Close the form, ensure the schedule starts and stops at the correct times
* Repeat for each relay
* Repeat but check “Apply to All Schedules”
  + Ensure each relay receives the new schedule

**Importing/Exporting**

* Export the current schedule
* Change the current schedule and save
* Import the previously exported schedule
* Ensure the settings were imported correctly
* Last relay states are remembered when computer reboots