Contents

1 How to

1.1 Adding, removing and changing channel numbers

1.1.1 Channel numbers

Channel numbers are stored in `main/channels.h`. Within the `create_tv_channels()` function, a channel is specified by writing `CHANNEL("channel name", channel number);`. For example:

```
CHANNEL("BBC_ONE_HD", 101);
```

The number can be changed without much issue, and the program will automatically figure out which number keys to press on the remote when that channel is requested. The order of the channels does not matter here.

```
12  void create_tv_channels()
13  {
14      CHANNEL("BBC_ONE_HD", 101);
15
16      CHANNEL("BBC_TWO_HD", 102);
17
18      CHANNEL("BBC_THREE_HD", 107);
19
20      CHANNEL("BBC_FOUR_HD", 106);
21
22      CHANNEL("BBC_NEWS", 231);
23
24      CHANNEL("ITV1_HD", 103);
25
26      CHANNEL("ITV1", 3);
27
28      CHANNEL("ITV2", 6);
29
30      CHANNEL("ITV3", 10);
31
32      CHANNEL("ITV4", 26);
33
34      CHANNEL("CHANNEL_4_HD", 104);
35
36      CHANNEL("CHANNEL_5_HD", 105);
37
38      CHANNEL("FILM4", 14);
39
40      CHANNEL("MORE4", 18);
```

The name of the channel is important as this determines the name of the "action list" for that channel. In order to allow each remote to send the numbers for a given channel, an action list for each remote is automatically created. The names of the action lists are set by prefixing a name for the remote to the name of the channel. For example, `TV_BBC_ONE_HD` will be an action list of the TV remote's 1, 0 and 1 keys. `HDR_BBC_ONE_HD` will be another action list using the HDR remote's number keys instead. This prefixing can be seen in `main/actionlist.h` in the `Channels` section.

Channels also need to be known by the UI so they can be used when generating the channel button grid. This is done in `main/ui/custom-component-data.slint`. In the `ChannelButtonData` section, a list of channels is specified. This list is used to create the buttons in the grid. **The order of this list is the order the buttons are created in the grid, column by column.**

```
export global ChannelButtonData { //list of data to generate tv channel buttons automatically
    //each button will have the specified image shown, and run the given action list when clicked
    //buttons are generated in the grid in the order of this list, column by column

    out property <[RemoteButtonData]> data: [
        { icon: @image-url("icons/channels/BBC ONE EastHD.png"),        action_list_id: "BBC_ONE_HD",          },
        { icon: @image-url("icons/channels/BBC TWO HD.png"),           action_list_id: "BBC_TWO_HD",          },
        { icon: @image-url("icons/channels/BBC THREE HD.png"),         action_list_id: "BBC_THREE_HD",        },
        { icon: @image-url("icons/channels/BBC FOUR HD.png"),         action_list_id: "BBC_FOUR_HD",         },
        { icon: @image-url("icons/channels/BBC NEWS.png"),            action_list_id: "BBC_NEWS",            },
        { icon: @image-url("icons/channels/ITV1.png"),                action_list_id: "ITV1",                },
        { icon: @image-url("icons/channels/ITV1 HD.png"),             action_list_id: "ITV1_HD",             },
        { icon: @image-url("icons/channels/ITV2.png"),                action_list_id: "ITV2",                },
        { icon: @image-url("icons/channels/ITV3.png"),                action_list_id: "ITV3",                },
        { icon: @image-url("icons/channels/ITV4.png"),                action_list_id: "ITV4",                },
        { icon: @image-url("icons/channels/Channel 4 HD.png"),        action_list_id: "CHANNEL_4_HD",        },
        { icon: @image-url("icons/channels/Channel 5 HD.png"),        action_list_id: "CHANNEL_5_HD",        },
        { icon: @image-url("icons/channels/Film4.png"),               action_list_id: "FILM4",               },
        { icon: @image-url("icons/channels/More 4.png"),             action_list_id: "MORE4",               },
        { icon: @image-url("icons/channels/E4.png"),                 action_list_id: "E4",                  },
        { icon: @image-url("icons/channels/Dave.png"),               action_list_id: "DAVE",                },
        { icon: @image-url("icons/channels/Yesterday.png"),          action_list_id: "YESTERDAY",           },
        { icon: @image-url("icons/channels/QUEST.png"),              action_list_id: "QUEST",               },
```

1.1.2 Adding a new channel

The full process for adding a new channel is as follows:

1. Create a `CHANNEL` in `main/channels.h`
2. Place the channel logo image in `main/ui/icons/channels/`
3. Add an entry in `main/ui/custom-component-data.slint` in the `ChannelButtonData` section
   - The simplest way will be to copy an existing line (including both {} and the comma at the end)
   - The `icon` should be the directory to the channel image placed earlier. The directory is relative to the `custom-component-data.slint` file, i.e. starting from the `ui` folder.
   - The `action_list_id` should match the name used in `CHANNEL(name, number)`
   - The position of the entry will be where that channel is placed in the grid
4. The channel should now be added. Build and upload the program to the remote.

1.1.3 Removing a channel

The process for removing a channel is the reverse of adding one. The `CHANNEL` in `main/channels.h` should be removed, as well as the entry in the list in `main/ui/custom-component-data.slint`. Changing channel numbers only needs requires editing `channels.h`.

1.2 Adding a new remote

The learnt signals are stored in `remotes.h` and organised per-remote. To add a new remote, first add a new namespace in the file:

```
namespace NewRemoteName
{

}
```

The codes for the remote will be placed in here. They are created by writing `ACTION(name, "code")`. For example:

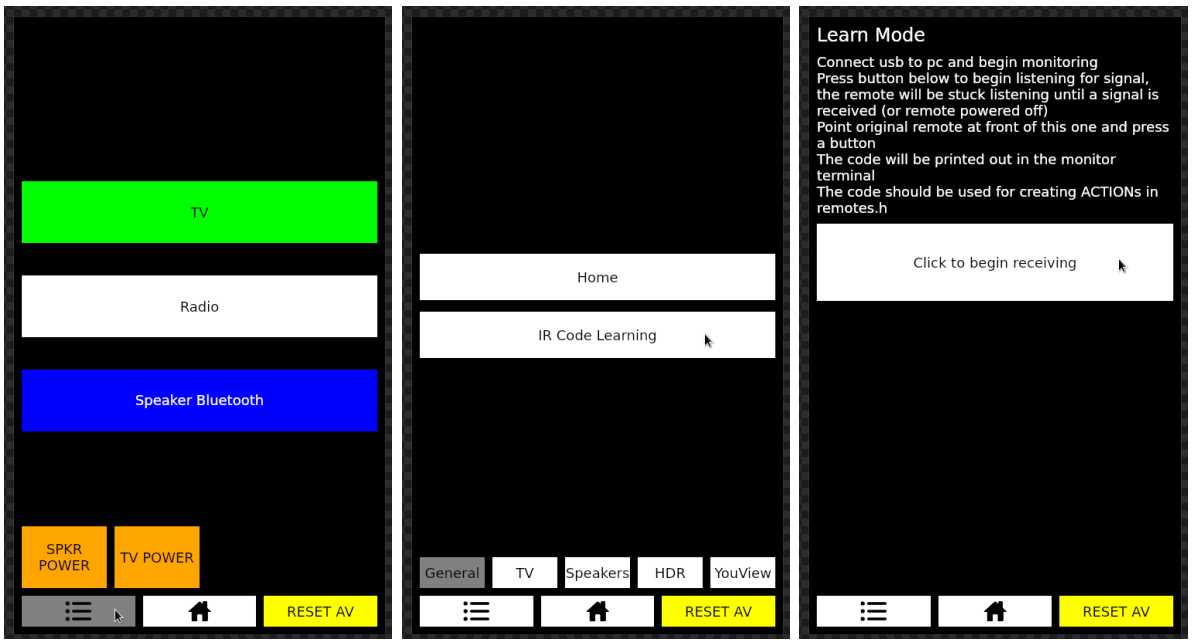
```
ACTION(POWER, "06e08da601b681b0052281b401bb81b101d4819...");
```

The `name` will be used later when referencing this `action` in `action lists`. The `code` in the example has been shortened, the actual code will be quite long.

1.2.1 Learning signals

To add codes to the new remote, first take the ESP remote and the original which is being learnt from. Connect the ESP remote to the computer with USB. TODO: decide how to handle building/uploading/monitoring. most likely use WSL.

Navigate to the `IR Recieve` screen in the ESP remote through the screen list:



Click the button to begin listening for a signal, then press a button on the remote being learnt. A code should appear in the monitor terminal, highlight it and copy it.

Now go back to `remotes.h` and create a new `ACTION` with a suitable name, and paste the code between quote marks. See the existing codes for examples. Repeat this for all the buttons on the remote.

1.2.2 Creating action lists

In order for a remote's signal to be used, it needs to be part of an `action list`. An action list is a sequence of one or more remote buttons which should be pressed automatically. These are

created in main/actionlist.h. The action list file is organised with general macros at the top, followed by code for automatically generating channel action lists, then by each remote's individual buttons. To simplify the program, all individual buttons still need to be part of an action list.

To create an action list it needs to be added to the list of all action lists, appropriately named ALL_ACTION_LISTS. Within the function create_action_lists, an action list is added to ALL_ACTION_LISTS with:

```
ALL_ACTION_LISTS["action list name"] = {
    RemoteName::ActionName,
    RemoteName::ActionName,
    ...
}
```

RemoteName is the namespace which the remote's ACTION is in.

1.3 Building and uploading the program to the remote

2 Documentation

2.1 Actions and action lists

Based on the Phillips Pronto remotes, sending one or more IR signals is modeled with the concept of an Action which is a single IR signal akin to pressing a button on a remote, and an Action List which is a sequence of Actions.

For extra functionality, there are currently 3 types of actions defined in main/action.h:

- ActionRemoteSignal - a single signal from one learnt remote button press
- ActionDelayMilliseconds - a delay of a given amount of time before the next action is run
- ActionRepeatIRForMilliseconds - a specified ActionRemoteSignal is repeated quickly to mimic the button being held down for a given time period

2.1.1 Creating actions

The majority of actions are of type ActionRemoteSignal learnt from a remote. These can be learnt through the remote using the IR Receive screen from the screens list in the general tab.

The learnt signals are stored in main/remotes.h. To easily support a variety of remotes, the signals are stored as the series of high and low pair times which the ESP RMT peripheral understands. This means the codes are very long, but no information needs to be stored about how the signal should be encoded, and signals from all remotes can be treated the same way when learnt and sent.

A namespace is created to separate each remote's actions, and each action is created with a macro for simplicity:

```
ACTION(name, code)
```

The name argument is the variable name for the action in the remote namespace. The code is a string of the learnt timing data for the signal.