RetroChallenge 2021/10 competition entry: Using the Commodore 64 to control a remote control car.

A picture containing text, indoor

Description automatically generated

Description:

The Commodore 64 was released in 1982 and quickly became the best selling computer of all time. Many users treated the system as a games controller, but it was a real computer and could also be used as a microcontroller, using it’s ports like the User Port.

In this project I wanted to use this retro computer to control a remote control (RC) car. Using an inexpensive RC car and re-fitting it with an Arduino to allow connection to a radio transceiver (NRF24L01). On the Commodore 64 side I decided to use RS232, which on the C64 needs an adapter to bring it up to the voltages needed. The adapter released by Commodore was called the VIC-1011A.

Control of the RC car can be via a joystick and recorded (to a floppy disk) for later autonomous 'play back' by the C64. The C64 will use RS232 to a microcontroller with a wireless transceiver that can connect to the RC car. Commodore 64 control on an RC car By Steve Smit Started in September 2021

The code on the Arduinos used in this project is written using the IDE in the standard C++. The code on the Commodore 64 will initially be in BASIC, but will likely be re-written in either Forth (DurexForth) or assembler.

22nd September 2021 : First piece of progress, get C64 to send bytes representing the directions of the joystick by RS232 to a transceiver to a receiver to confirm 'commands' can be sent. Code on C64 written in DurexForth: <https://youtu.be/1RyrSl5RLjI>

A picture containing text, indoor, electronics

Description automatically generated

I purchased 1/14 scale RC ‘Monster Truck’ off-road 2WD car for just under AUD$40.

A picture containing text

Description automatically generated A red race car

Description automatically generated with medium confidence

I first tried to ‘hack’ the controller to see if perhaps this might mean I could leave the electronics in the car unmodified. This initially worked!:

<https://youtu.be/TNTKXlKmA1Y>

But, for no apparent reason, the ‘steering’ motor, while making a straining noise, now doesn’t seem to have the torque to turn the car anymore. Hacking the controller wasn’t ideal in any case, as I wanted to have more control via the Commodore 64. I decided to open the car up, firstly to see if I could address the steering motor issue, but also to replace the motor controller with a l298n dual h-bridge motor controller module and hook up an Arduino to do some testing.

A picture containing indoor, computer, floor, keyboard

Description automatically generated

Even directly connecting to the ‘steering’ motor didn’t fix the lack of movement issue. I have ordered a few replacement DC motors as an option to fix the issue. An alternative is to replace the steering mechanism with a servo motor. Here is how the current DC motor controls the steering:

A close-up of a car engine

Description automatically generated with medium confidence

While I wait for replacement items, I better get onto the code on the C64. I decided that the opening screen should offer the user 4 choices:

F1 = Operate RC car without recording

F3 = Operate RC car with recording

F5 = Show saved sequences

F7 = “Play” current sequence in memory

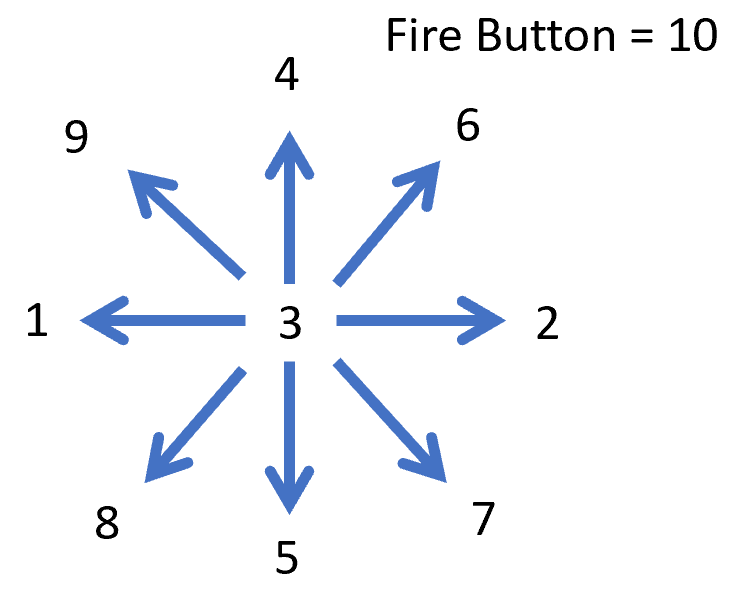
I realized that I didn’t know what byte is returned if someone presses the Function keys (F1, F3, etc.) so I wrote a quick basic program to get the values:

A blue screen with white text

Description automatically generated with low confidence

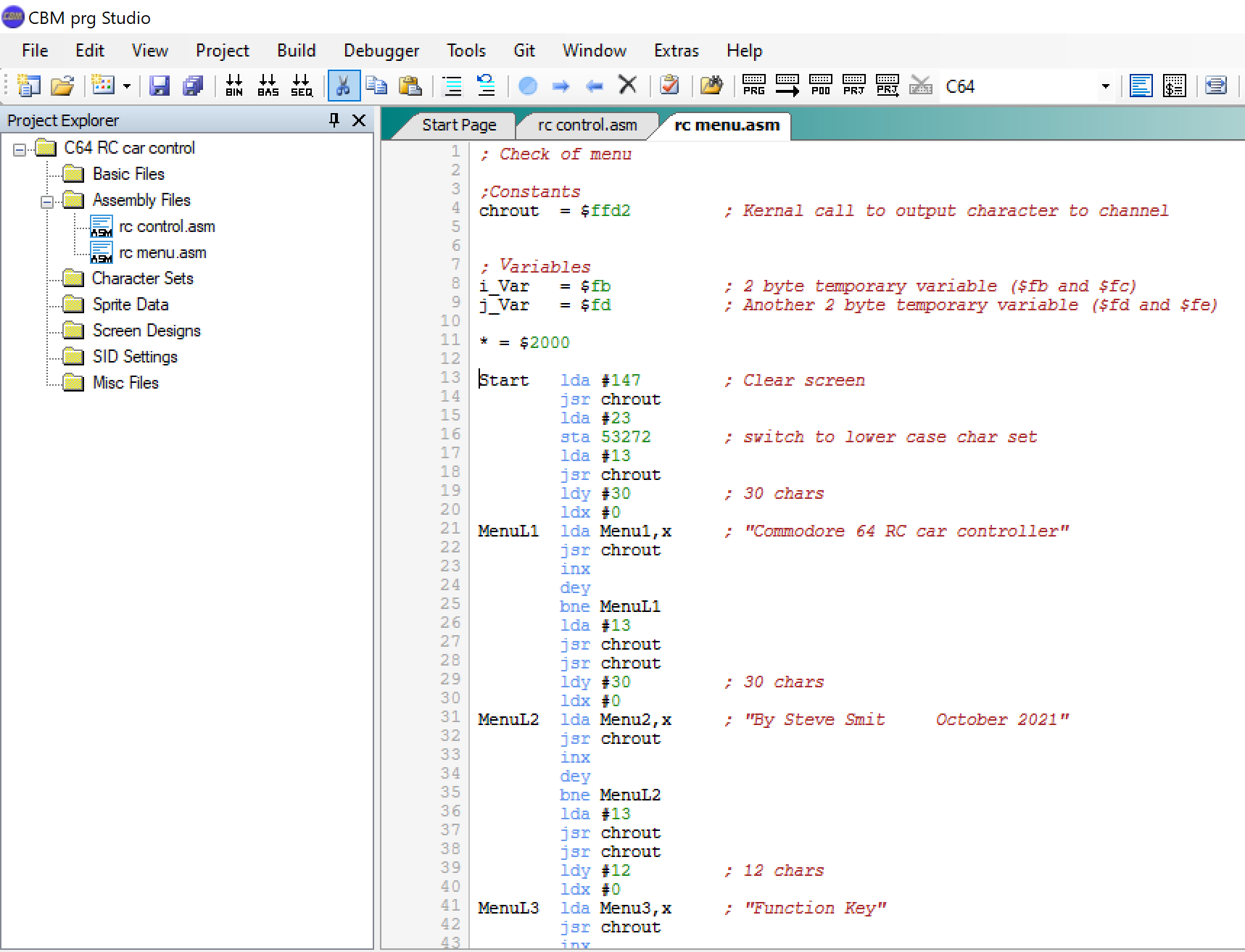
From this I see that F1 = 133, F3 = 134, F5 = 136 and F7 = 137.

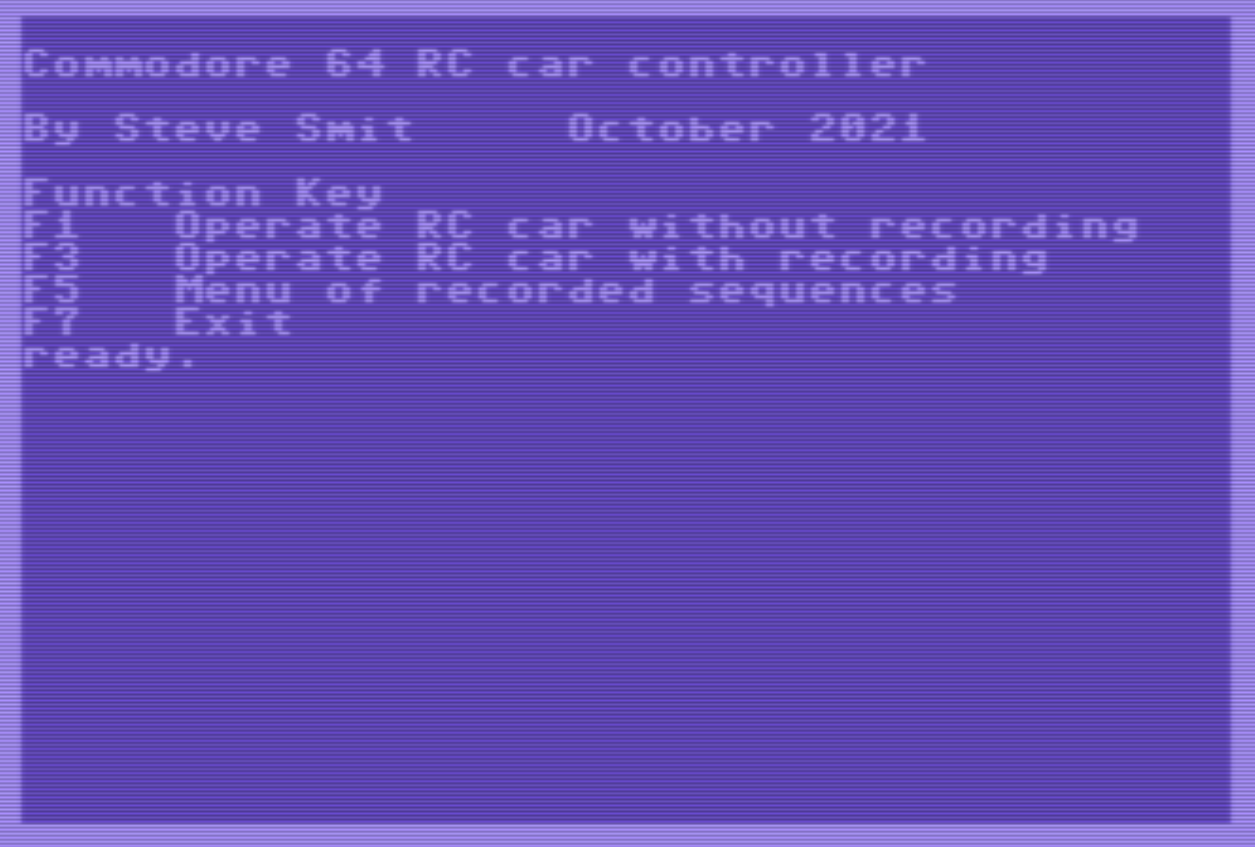
For the joystick movements, I decided that each direction and the fire button be given a value that would be sent as a command to the RC car.



27 September 2021 – Assembly programming

I’m still not sure what language to use on the Commodore 64, but I know that Assembly has the maximum level of control and speed, so l will make a start and see how I go. I’m using an IDE called CBM Prg Studio to enter the assembly that compiles the machine code:





Well, not the flashiest of menus, but it’s a start.