Toy Car Control

Record Drive, Rewind/Repeat and OtherPatterns

(Padmaja AVL, Anuradha Palepu, Haneen Alamat)
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
E-mails: langaluruvenkat@oakland.edu, apalepu@oakland.edu, hfalamat@oakland.edu

Abstract- **The toy car controller is a project designed using the Dragon-12 light board and an actual purchased toy car with remote controller. This project is designed to capture the drive path when user uses the remote control and then use the same data to move the car in the reverse path. Repeat the drive path from the data already saved in microcontroller or program the car to show desired patterns like left circle, right circle, zig zag etc.**

I. INTRODUCTION

The Toy car control is a system to record the drive path when user presses the buttons of car remote and then use the saved data to trigger the car in the reverse way of the path already driven. This project also can repeat the same path n number of times while user might have just pressed buttons only once.

The remote controller of the car has four keys to control the actions of the car. They are Left, Right, Forward and Backward buttons. The buttons are wired out and connected to the Digital I/O pins of HCS12 Microcontroller to read the actions. The user presses any of the four buttons to move car in desired path. The user pressed data is captured into a buffer. The buffer data is then played back/ repeated based on user selected option. Also, this project is programmed with some DEMO Patterns. The current course of action is displayed on the LCD monitor.

This project makes use of Remote control Tx2 chip that interfaces with Digital I/O pins of HCS12 microcontroller and I/O devices like DIP switches and LCD Monitor. The Timers, Buffers, Interrupts and asm & C programming would constitute and perform the full functionality.

II. BLOCKDIAGRAM

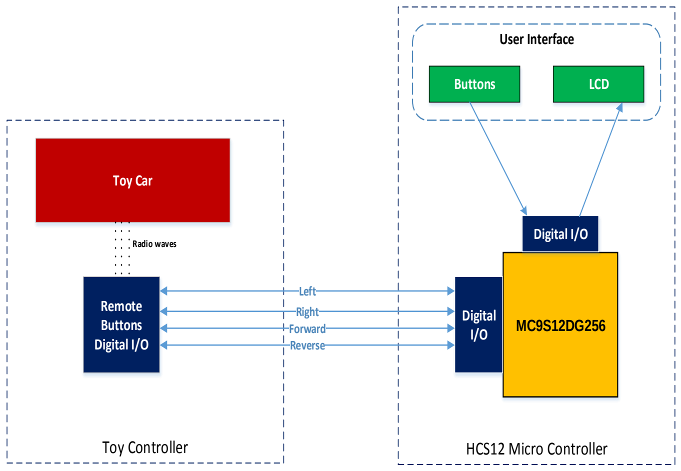


Figure 1: Block diagram

III. HARDWARE AND LOGIC CIRCUITS

A. Toy car with Remote Controller

One of the devices we used for this project is a remote controlled toy car that sends commands to the car by transmitting radio waves when a particular movement selection is chosen by the user. The vehicle itself has a full function radio transmitter that controls the Forward/Reverse and Left/Right steering which helps us build on with the rest of the code because it already has certain basic functionalities.

The Toy car remote controller has a chip which is connected to the movement selection buttons. This chip will drive the four buttons. The Tx2 chip is used as transmitter and its pin assignment is as follows:

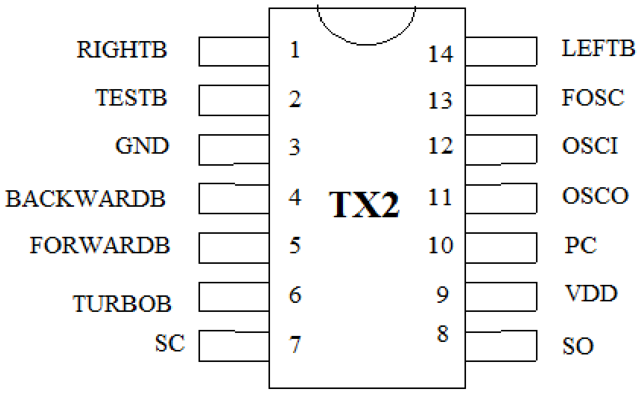
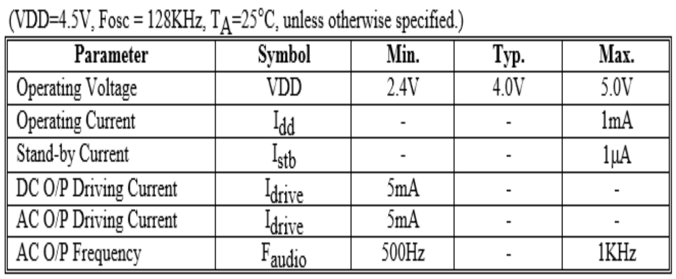


Figure 2: Tx2 Chip Pin assignment

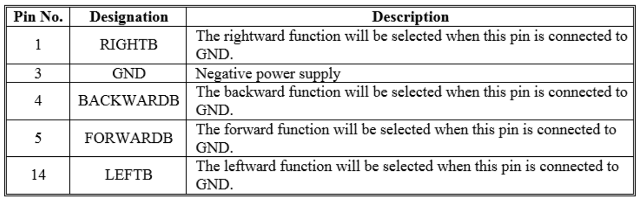
The Tx2 is a CMOS LSI chip designed for remote controlled car applications. The Tx2 electrical characteristics are as follows:


(VDD=4.5V, Fosc = 128KHz, T_A=25°C, unless otherwise specified.)

Parameter	Symbol	Min.	Typ.	Max.
Operating Voltage	VDD	2.4V	4.0V	5.0V
Operating Current	I_{dd}	-	-	1mA
Stand-by Current	I_{stb}	-	-	1μA
DC O/P Driving Current	I_{drive}	5mA	-	-
AC O/P Driving Current	I_{drive}	5mA	-	-
AC O/P Frequency	F_{audio}	500Hz	-	1KHz

Figure 3: Tx2 Chip Electrical characteristics

The pins behavior is described in the below table:



Pin No.	Designation	Description
1	RIGHTB	The rightward function will be selected when this pin is connected to GND.
3	GND	Negative power supply
4	BACKWARDB	The backward function will be selected when this pin is connected to GND.
5	FORWARDB	The forward function will be selected when this pin is connected to GND.
14	LEFTB	The leftward function will be selected when this pin is connected to GND.

Figure 4: Tx2 Chip - pin description

B. *Dragon 12 Light Board – HCS12 Microcontroller*

The Dragon12-Light board comes with the MC9S12DG256 chip and other peripherals.

The main features of the MC9S12DG256 are listed below:
· Powerful 16-bit CPU
· 256K bytes of flash memory
· 12K bytes of RAM
· 4K bytes of EEPROM
· SCI ports · SPI ports
· CAN 2.0 ports
· I2C interface, · 8-ch 16-bit timers
· 8-ch 8-bit or 4-ch 16 bit PWM
· 16-channel 10-bit A/D converter
· Fast 25 MHz bus speed via on-chip Phase Lock Loop
· BDM for in-circuit programming and debugging
· 112-pin LQFP package offers up to 91 I/O in a small footprint

The pin assignment of HCS12 is as follows:

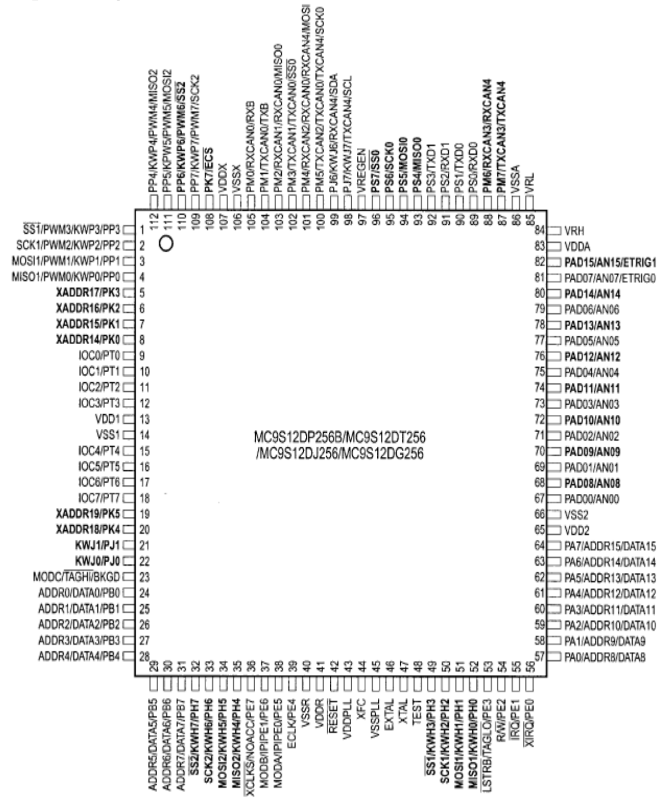


Figure 5: HCS12 Pin assignment

Many I/O pins of the MC9S12DG256 on the Dragon12-Light board are used by on-board Peripherals but not all on-board peripherals will be used by one application program. So the I/O pins on unused peripheral devices can still be used for our project.

The Ports used in this project are PortE, PortK, PortT and PortH. The general description of each port is as follows:

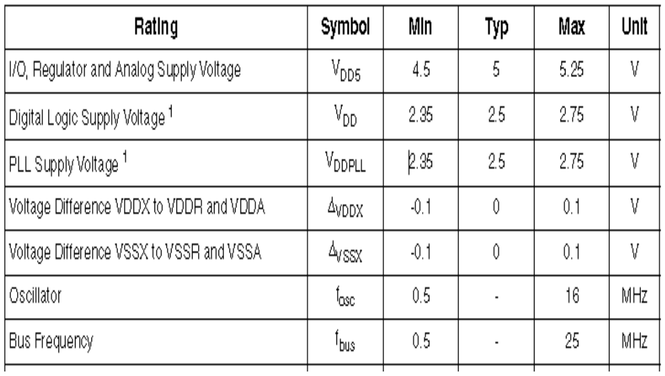
PORT E
General Purpose I/O (GPIO) Port. Data Register (PORTE): \$0008. Pin 0 is connected to /XIRQ. Pin 1 is connected to /IRQ. Data Direction Register (DDRE): \$0009. '1' means output, '0' means input. Port E Assignment Register (PEAR): \$000A. Write \$10 to configure all 8 bits as I/O pins. Dragon12-Light Board: External I/O pins.

PORT K
General Purpose I/O (GPIO) Port. Data Register (PTK): \$0032. Data Direction Register (DDRK): \$0033. '1' means output, '0' means input. Dragon12-Light Board: External I/O pins. Also, PK5-PK0 they are connected to the alphanumeric LCD module. The LCD module operates in a write-only, 4-bits mode.

PORT T
General Purpose I/O (GPIO) Port. Data Register (PTT): \$0240. Data Direction Register (DDRT): \$0242. '1' means output, '0' means input. Dragon12-Light Board: External I/O pins.

PORT H
General Purpose I/O (GPIO) Port. Data Register (PTH): \$0260. Data Direction Register (DDRH): \$0262. '1' means output, '0' means input. Dragon12-Light Board: External I/O pins. Also, they are connected to the DIP switch and Push Buttons (4 LSBs).

The electrical characteristics of HCS12 are as follows:



Rating	Symbol	Min	Typ	Max	Unit
I/O, Regulator and Analog Supply Voltage	V_{DD5}	4.5	5	5.25	V
Digital Logic Supply Voltage [1]	V_{DD}	2.35	2.5	2.75	V
PLL Supply Voltage [1]	V_{DDPLL}	2.35	2.5	2.75	V
Voltage Difference VDDX to VDDR and VDDA	Δ_{VDDX}	-0.1	0	0.1	V
Voltage Difference VSSX to VSSR and VSSA	Δ_{VSSX}	-0.1	0	0.1	V
Oscillator	f_{osc}	0.5	-	16	MHz
Bus Frequency	f_{bus}	0.5	-	25	MHz

Figure 6: HCS12 Electrical Characteristics

The VDDX, VSSX, VDDR and VSSR pairs supply the I/O pins, VDDR supplies also the internal voltage regulator. VDD1, VSS1, VDD2 and VSS2 are the supply pins for the digital logic, VDDPLL, VSSPLL supply the oscillator and the PLL. VSS1 and VSS2 are internally connected by metal. VDDA, VDDX, VDDR as well as VSSA, VSSX, VSSR are connected by anti-parallel diodes for ESD protection.

IV. Experimental Setup

The remote control wiring diagram is as below:

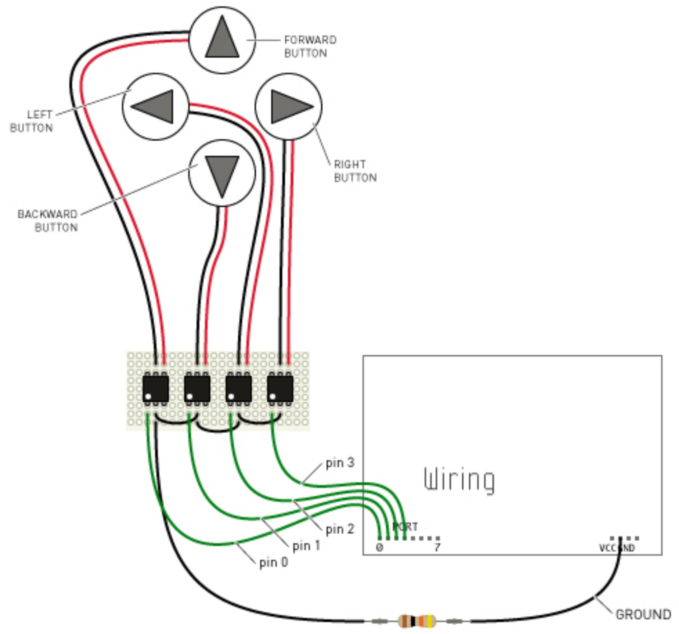


Figure 7: Wiring connections of Remote Control Keys

The four control keys of remote controller are wired out from the respective pins of Tx2 Chip. Each wire is connected to a pin of HCS12 of desired port, say PORTE and then configure the HCS12 port as output. Another set of wires from the same control keys are wired out and connected to another port say PORTH which is configured as input.

A pull up resistor of 10K is used to drive the Forward and Backward pins of HCS12 to make the input stable and when the control key is pressed only they are connected to GND. This is shown in the below diagram.

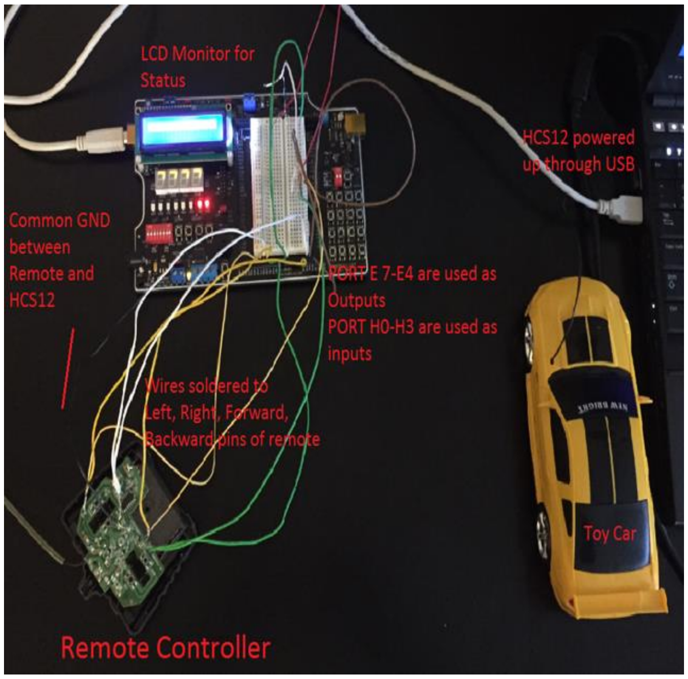

LCD Monitor for Status
HCS12 powered up through USB
Common GND between Remote and HCS12
PORT E 7-E4 are used as Outputs
PORT H0-H3 are used as inputs
Wires soldered to Left, Right, Forward, Backward pins of remote
Toy Car
Remote Controller

Figure 8: Experimental setup

V. Hardware Architecture

The complete hardware architecture of control key connections of HCS12 port is depicted as below:

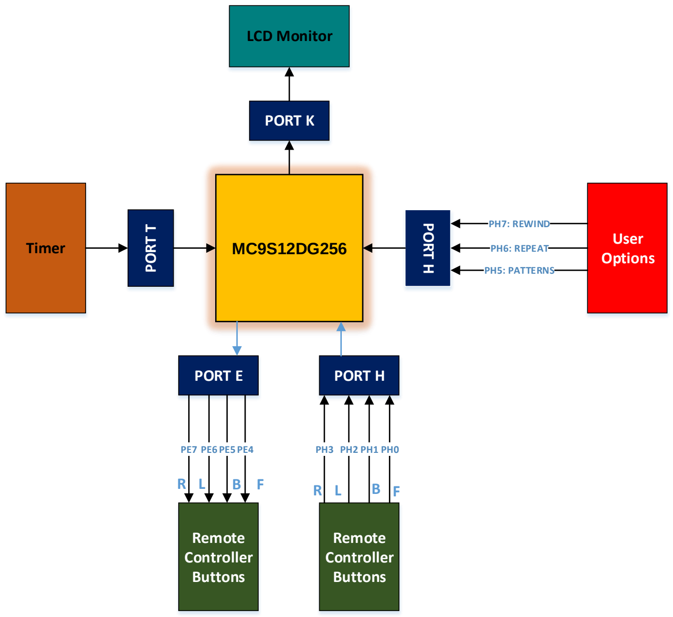


Figure 9: Hardware Architecture

Port E:

PE4 - Pin 39 - As Output from HCS12 to FORWARD key
PE5 - Pin 38 - As Output from HCS12 to BACKWARD key
PE6 - Pin 37 - As Output from HCS12 to LEFT key
PE7 - Pin 36 - As Output from HCS12 to RIGHT key

Port H:

Each pin in port H reads a switch, but it still can be used as an input for reading a TTL or CMOS output from any circuit. In our project, three buttons of DIP switch on Port H are used as inputs. These are dedicated for user to select the options like REWIND, REPEAT and DEMO PATTERNS. If REWIND option is selected, the drive path is played backwards. If REPEAT is selected, the drive path is repeated n number of times. If DEMO PATTERNS is selected, the car will move in preprogrammed patterns like circle, zig zag etc.

PH7 (user option1) - Pin 32 – DIP sw (input) for REWIND
PH6 (user option2) - Pin 33 – DIP sw (input) for REPEAT
PH5 (user option3) - Pin 34 – DIP sw (input) for PATTERNS

The other PORTH pins are used for reading the car movement when four control keys are pressed.

PH0 - Pin 52 - As input to HCS12 from FORWARD key
PH1 - Pin 51 - As input to HCS12 from BACKWARD key
PH2 - Pin 50 - As input to HCS12 from LEFT key
PH3 - Pin 49 - As input to HCS12 from RIGHT key

Port K:

Port K is used to drive the LCD Monitor. When the user presses a button either REWIND, REPEAT or DEMO PATTERNS, it is displayed in the LCD monitor during the complete course of respective action performed by the car.

PK2 - Pin 6 - DB4 of LCD module (bi-directional)
PK3 - Pin 5 - DB5 of LCD module (bi-directional)
PK4 - Pin 20 - DB6 of LCD module (bi-directional)
PK5 - Pin 19 - DB7 of LCD module (bi-directional)

VI. Operation and Programming

A. *Implementation*

The HCS12 digital ports are selected to read the input from four control keys of remote controller and when desired drive the same keys as output from the data stored earlier. When data pins are configured as input, Data read from the data register reflects the instantaneous voltage levels on the input pins.

Port H is chosen as input port. Pins PH0, PH1, PH2 and PH3 of HCS12 are connected to FORWARD, REVERSE, LEFT and RIGHT steering pins respectively.

Similarly, the output from GPIO ports of HCS12 can drive a 5V digital peripheral and hence Port E is chosen as output port. PE0, PE1, PE2 and PE4 are used to drive FORWARD, REVERSE, LEFT and RIGHT control key pins respectively. When the data register value of respective pins is high, then associated pin will deliver 5V and respective button is pressed. This way all but button presses can be programmed from the microcontroller.

A 10k ohms pull up resistor is used to make the PH0 and PH1 pins stable.

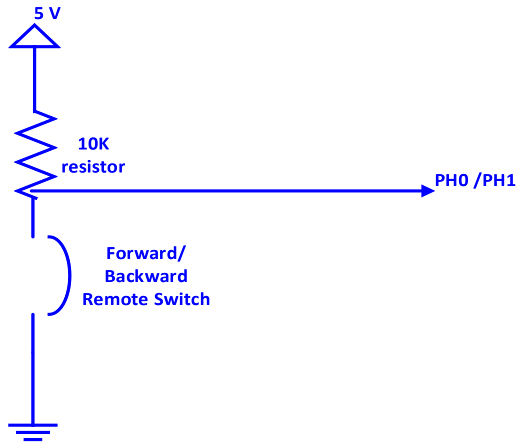


Figure 10: Pull Up resistor to control Port H pins

Port H is used to offer three services/options to the user. A REWIND, REPEAT and DEMO PATTERNS button is configured and when the user presses the desired button, one corresponding action will be taken. For instance, if user presses the REWIND button, the drive path stored in the memory of microcontroller feeds the values in the reverse order to the data register of Port E. When respective pins get appropriate voltages, car travels in the expected direction.

While any of the two actions are happening, LCD monitor driven by Port K will display the current action during the complete course of car movement.

B. *Software Control Flow*

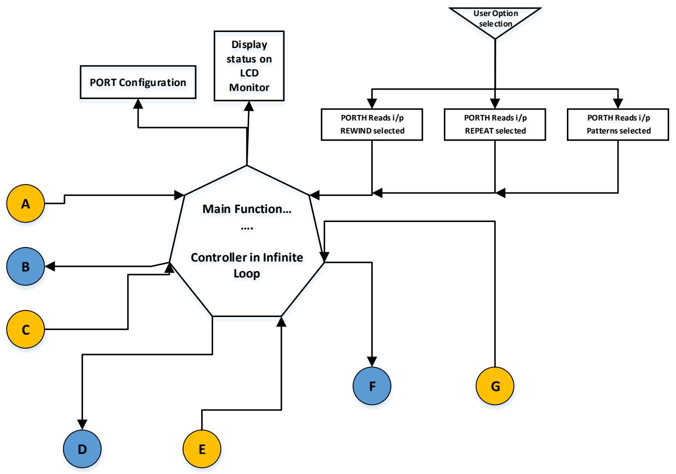

User Option selection
PORT Configuration
Display status on LCD Monitor
PORTH Reads i/p REWIND selected
PORTH Reads i/p REPEAT selected
PORTH Reads i/p Patterns selected
Main Function... .... Controller in Infinite Loop
A
B
C
D
E
F
G

Figure 11a: Software main function

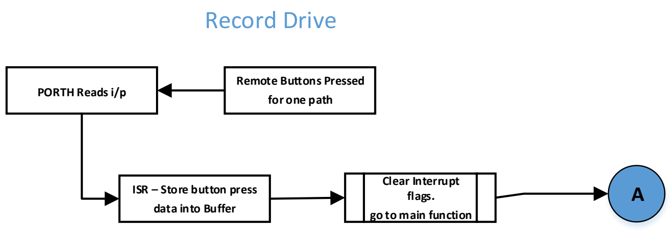

Record Drive
PORTH Reads i/p
Remote Buttons Pressed for one path
ISR – Store button press data into Buffer
Clear Interrupt flags. go to main function
A

Figure 11b: Software Record Drive

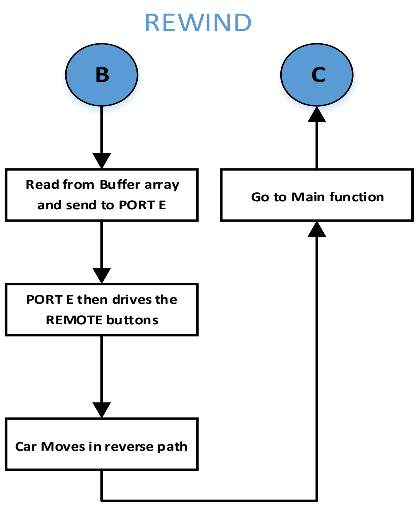

REWIND
B
C
Read from Buffer array and send to PORT E
Go to Main function
PORT E then drives the REMOTE buttons
Car Moves in reverse path

Figure 11c: Software Rewind

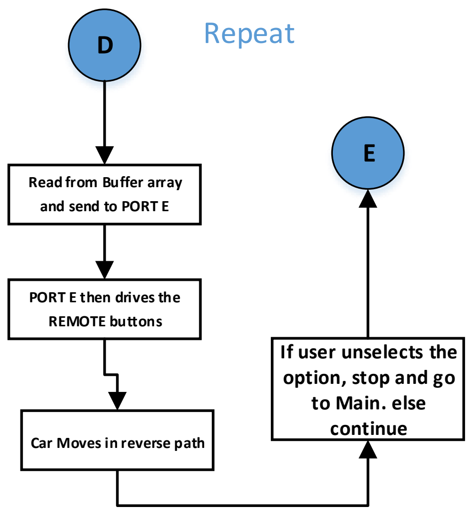

D
Repeat
Read from Buffer array and send to PORT E
PORT E then drives the REMOTE buttons
Car Moves in reverse path
E
If user unselects the option, stop and go to Main. else continue

Figure 11d: Software Repeat

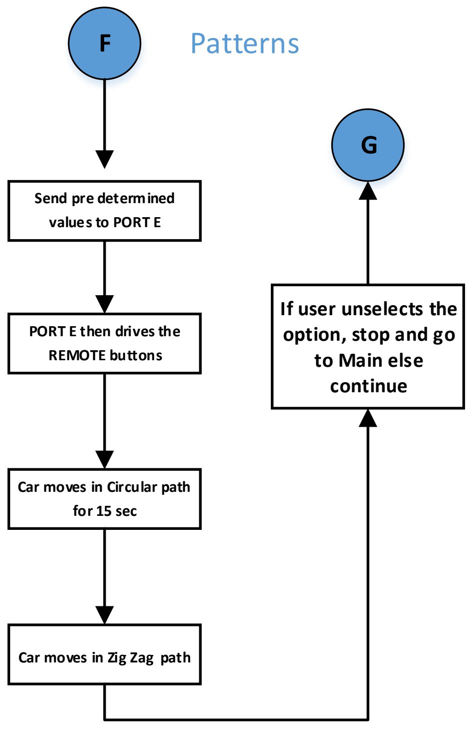

F
Patterns
Send pre determined values to PORT E
PORT E then drives the REMOTE buttons
Car moves in Circular path for 15 sec
Car moves in Zig Zag path
G
If user unselects the option, stop and go to Main else continue

Figure 11e: Software Patterns

VII. Results

We have successfully read the input actions of four control keys of the remote and captured the data into a buffer. When user selected any of the three options from REWIND, REPEAT, PATTERNS using DIP switch input, the car changed its state and acted accordingly. The Patterns we demonstrated like left circle for 15 secs, right circle for 15 sec, zig zag path forward and zig zag path backward were more spectacular and funny.

VIII. Conclusions

Overall, this project is more of a modified version of remote control cars that are already in the market. This particular vehicle can record movements, rewind and repeat along with just move it in one of the basic user-selected directions: FORWARD, BACKWARD, RIGHT and LEFT. Additionally to the four basic directions, this car can move in circles, zig-zag, a square and many other directions. Our project mainly demonstrates one of the many powerful features of HCS12 from using GPIO pins to using other peripherals in tandem without affecting the performance. Our project can be enhanced to store 10 mins of data based on the HCS12 memory availability and then feedback to the car to show the path it has taken in reverse way. Extended to this idea, if a PWN channel input is used, the car maneuvers can be more interesting.

IX. References

[1] MC9S12DT256 Device user guide v03.07, Freescale Semiconductor Inc., 24Mar 2003.

[2] HCS12 CPU Reference Manual_S12CPUV2, Rev. 4.0, Freescale Semiconductor Inc., Mar 2006.

[3] Dragon12_light_hcs12_manual_A, Ver 1.00, www.evbplus.com

[4] Han-Way Huang, The HCS12 - 9S12 - An Introduction to Software and Hardware Interfacing, 2d ed., Delmar Cengage Learning, 2010.