

Controlling RC Vehicles with Your Calculator

Operates Many—But Not All—4-Function, 27-MHz RC Vehicles

The RC Controller from Math Machines, Ltd. provides a simple, fun, safe and productive tool for introducing students to applications of math and science in the world of robotics. Students **see** the results of their mathematical calculations as immediate, physical actions.



Preparing Your Calculator

We recommend that you download the TI-Basic calculator programs SIGNAL, CONTROL and DCUINIT from either the Math Machines website (www.mathmachines.net) or from the CD provided with your RC Controller. To download programs, your computer must have one of the connection cables available from Texas Instruments. Appropriate computer software (TI Connect or TI Graphlink) must also be installed on your computer. See TI's website at <http://education.ti.com/us/product/accessory/connectivity/features/software.html> for additional information and for a free download of the TI Connect software.*

If you do not have access to a computer and connector cable, you can also enter the abbreviated SIG program at right directly from the calculator keypad. Like the SIGNAL program available on the CD and on the Math Machines website, this program allows you to enter a signal *S*, which must be an integer between 0 and 15. The program also allows you to enter the time, *T*, in seconds for which the signal will be transmitted.

Key strokes	Screen Shows:
[PROG]/[NEW] [ENTER] SIG [ENTER]	PROGRAM Name=SIG0
[PROG] / [I/O] 1	
2 nd [ALPHA] "S?" [ALPHA] , [ALPHA] S [ENTER]	PROGRAM:SIG :Input "S?",S :Input "T?",T :Send((1,S),2,S, 0)) :Send((3,T,2,0,0)
[PROG] / [I/O] 1	
[2ND] [ALPHA] "T?" [ALPHA] , [ALPHA] T [ENTER]	
[PROG] / [I/OZ] [ALPHA] B	
2 nd {1,3,1,2, [ALPHA] S,0 2 nd } [ENTER]	
[PROG] / [I/OZ] [ALPHA] B	
2 nd {3, [ALPHA] T,2,0,0 2 nd } [ENTER]	
2 nd [QUIT]	

* An alternative version of TI Connect is provided on the CD in this package. This beta version was distributed by TI during 2004 and includes a program editor for writing and modifying calculator programs on your computer. The version of TI Connect available on TI's website at this printing does *not* include the program editor.

Preparing and Testing the RC Controller and Vehicle

The RC Controller requires a graphing calculator from the TI-83 /84 family, plus either a CBL2 or LabPro interface. (The Controller will also function with the original CBL using a special adapter cable, available separately.)

1. Assemble the calculator and CBL2 or LabPro, just as you would for data collection. Mount the calculator onto the CBL2 or LabPro, using the cradle provided with the interface, and use the black link cable to connect the calculator and interface. Be sure the CBL2 or LabPro has reasonably fresh batteries or is connected to its plug-in power adapter.
2. Switch the RC Controller “off,” and insert a 9-V battery into the compartment on the bottom. (Alternatively, you can use a plug-in adapter to provide the DC voltage. The adapter **must** be 9V DC, center positive, with a current capacity of at least 200mA.)
3. Screw the antenna into place on top of the RC Controller.
4. Use the cable provided to connect the RC Controller to the “DIG/ SONIC” port on the CBL2 or the “DIG / SONIC 1” port on the LabPro.
5. Prepare the 27-MHz, 4-function vehicle as directed on its package. Also install batteries as directed in the vehicle and the hand controller. Use the hand controller to verify that the vehicle functions correctly. (NOTE: If the vehicle runs when you don’t expect it to, verify that the RC Controller’s power switch is off.)
6. Turn on the RC Controller. Also turn on the vehicle and place it so you can observe its wheels or tracks directly. Run the SIGNAL or SIG program on your calculator and send the signals 1, 2, 4 and 8 for 2 seconds each. These four numbers correspond to the binary values that activate the four functions of the vehicle. You should see that each of number produces a distinct response from the vehicle. The red LED on the RC Controller should light to show it is on. When you are sending signals, the yellow LED should flash faintly.

Understanding the Signals

The table below shows the response typical of several 4-function, 27-MHz vehicles, including a vehicle with front-wheel steering (illustrated by New Bright’s model 2424) and one with tank-type steering (illustrated by JAKKS’s Road Champ RC vehicle). Some 27-MHz vehicles will respond to all 4 signals, but in different ways—for example, the signal that makes one vehicle go forward may make another vehicle move backwards. Depending on the specific coding used by the manufacturer, some 27-MHz vehicle may respond to only some of the 4 signals and some vehicle may not respond at all. *Have fun experimenting with different vehicles! We have found miniature radio-controlled mice, boats, submarines, flying saucers and many other low-cost “toys” that **do** respond to the RC Controller.*

Note that many movements of the vehicle require 2 simultaneous signals. To make the New Bright truck turn left, for example, you need to make it move forward while simultaneously turning the steering wheels to the left. You achieve this simply by

sending the sum of two signals. If “2” makes it move forward and “4” makes the steering wheels turn right, then signal “6” will make both happen simultaneously. To move the Road Champ forward, you need to send “5” (1+4) to make both the left and right wheels move forward.

Some signals are allowed under the rules of arithmetic but don't really make sense physically. Sending S=3, for example, tells New Bright's model 2424 truck to go forward (S=2) and backward (S=1) simultaneously. In reality, one of the two signals will probably take priority so the vehicle does respond. It is much easier to understand and control the vehicle's response, however, if you think in terms of the true binary elements—0, 1, 2, 4 and 8—which are shown in bold. If two different signals produce the same result, always use the lower value.

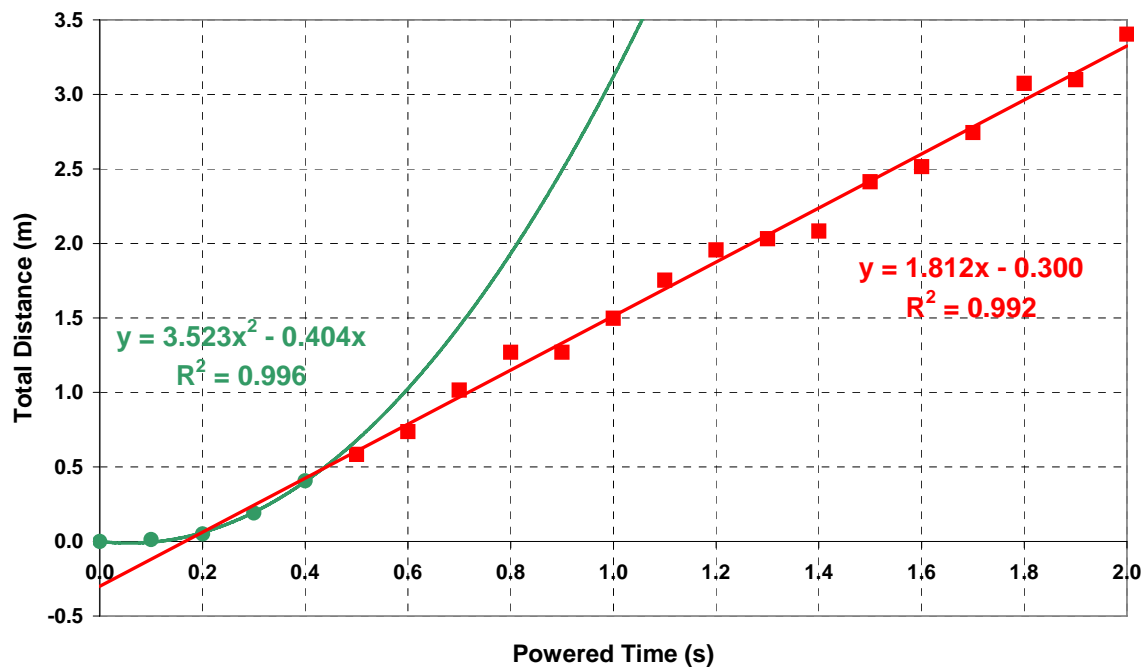
Integer (base 10)	Binary pattern	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$	New Bright Model 4242	JAKKS Road Champ
0	0000	0	0	0	0	STOP	STOP
1	0001	0	0	0	1	VEHICLE BACK	LEFT WHEEL FORWARD
2	0010	0	0	1	0	VEHICLE FORWARD	LEFT WHEEL BACK
3	0011	0	0	1	1		
4	0000	0	1	0	0	STEER RIGHT	RIGHT WHEEL FORWARD
5	0001	0	1	0	1	BACK RIGHT	VEHICLE FORWARD
6	0010	0	1	1	0	FORWARD RIGHT	PIVOT LEFT
7	0011	0	1	1	1		
8	0100	1	0	0	0	STEER LEFT	RIGHT WHEEL BACK
9	0000	1	0	0	1	BACK LEFT	PIVOT RIGHT
10	0001	1	0	1	0	FORWARD LEFT	VEHICLE BACK
11	0010	1	0	1	1		
12	0011	1	1	0	0		
13	0000	1	1	0	1		
14	0001	1	1	1	0		
15	0010	1	1	1	1		

Collecting Data for the Vehicle

A very good introductory algebra activity with the RC Controller is to give student teams the task of sending the vehicle a distance, x . Tell them in advance where their vehicle will start, but make it clear they will not know the location of the finish line until a few minutes before the start. To prepare, have the students collect data for the distance traveled by the vehicle when it is powered for a variety of time intervals. If you have several different vehicles that respond to the RC Controller (perhaps brought in by the students themselves), each team can work with its own vehicle. Let the groups take turns to test their results.

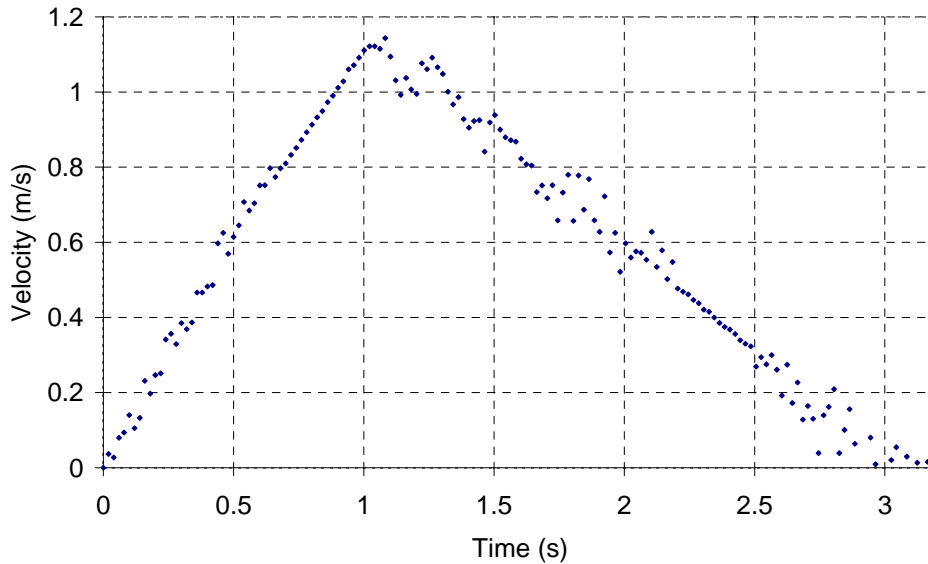
Typical data is shown below. Note that it is generally linear for times greater than about 0.5 seconds, but the y-intercept is not zero. It takes the vehicle a short time to get started. If you want your students to work with linear equations, limit them to times greater than 0.5 seconds. For more advanced students, you may want to include very short times and very short distances.

Total Distance vs. Powered Time

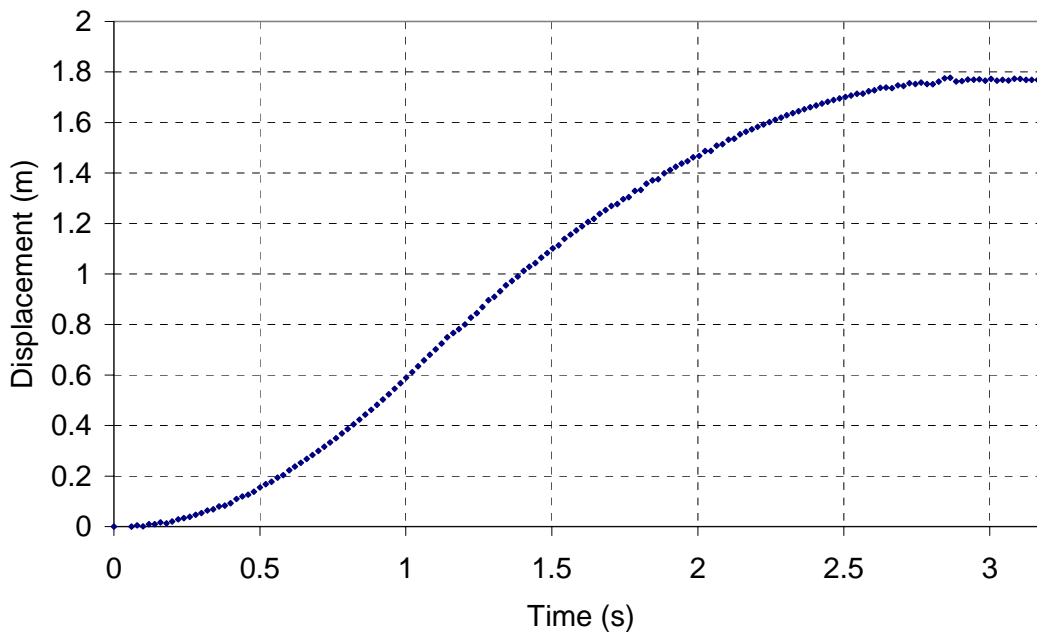


The two graphs below illustrate data collected using Vernier's motion detector to show instantaneous velocity and displacement of a vehicle similar (but not identical) to the one shown above. Note that the previous graph shows total POWERED time—the time during which power is delivered to the vehicle's motor—and total distance traveled. All vehicles have an additional deceleration time during which they continue to move after the power stops.

Velocity vs Time 1.000 s of Power



Displacement vs Time 1.000 s of Power



Signal

We often expect to see the output of a computer or calculator as a display on a screen or perhaps as a printed document. In a great many cases, however, the practical output of a computation is used directly to control something physical. The *Signal* program demonstrates how the TI-83-84 family of calculators can work with the CBL2 or LabPro to convert an integer into an electrical signal. This signal can then be passed to the RC Controller or used to activate a variety of other devices. The basic mathematical principles also apply to computers, to robotics, to manufacturing equipment and to many other practical situations.

To run the program:

- press the [PRGM] key,
- use the down arrow key to highlight the number of the SIGNAL program from the on-screen menu,
- press the [ENTER] key twice to start.

Select the option to "SET S AND T," and follow the instructions on the calculator screen. Experiment with various integers to see what happens. The process follows the standard binary system of mathematics and electronics, as indicated below.

Experiment with alternative ways of entering the integer and time. You can, for example, enter a sum of numbers rather than a single number. You can also use algebraic values, for example a constant such as K that was stored previously.

Also try the "KEY PRESS" option. Pressing any number key from 0 through 9 sends that signal until you press another key. Using standard "hexidecimal" notation, the letter keys A though F activate the integers 10 through 15. (It is not necessary to press the alpha key in this option.)

```

** SIGNAL **
SENDS SIGNAL, S
CBL2/LABPRO
<ENTER> TO START

SIGNAL
1:SET S AND T
2:KEY PRESS
3:COUNT 0 TO 15
4:COUNT 15 TO 0
5:QUIT

SIGNAL? 9
TIME(SEC)? 1.62

INPUT TIME, T,
IN SECONDS
BETWEEN 0 AND 20

OUTPUT:
S = 9
T = 1.62 SEC
<ENTER> TO START

9
PRESS KEY 0-9
OR A-F
<ENTER> FOR MENU

```

Integer (base 10)	Binary pattern	Data line 3 $2^3 = 8$	Data line 2 $2^2 = 4$	Data line 1 $2^1 = 2$	Data line 0 $2^0 = 1$
0	0000	Low	Low	Low	Low
1	0001	Low	Low	Low	High
2	0010	Low	Low	High	Low
3	0011	Low	Low	High	High
4	0100	Low	High	Low	Low
...	...				
15	1111	High	High	High	High

SIGNALF

The *SIGNALF* program (“Signal For”) has the same basic function as the *SIGNAL* program, but is intended for use as a module or subroutine within programs that you or your students write. They can use it, for example, to write a program that will activate the RC Controller to send it along a simple path.

```
PROGRAM: EXA
: 8→S
: 3→T
: PRGM SIGNALF
: 4→S
: J(3)→T
: PRGM SIGNALF
: █
```

To enter and run the EXA program:

- Press the [PRGM] key, use the arrow keys to select “NEW,” then press [ENTER].
- Type in the name of the program you want, for example EXA for “Example A.” (Note that the alpha lock is already active as you type the program name.)
- Enter “8→S” (8 STO [ALPHA] S) just as you normally would to store a value. When you run the program, this will designate the binary output signal. When you are ready to move to the next program line, press [ENTER].
- Enter “3→T” to designate the time.
- Press the [PRGM] key, use the right arrow key to highlight “EXEC,” and use the down arrow key to highlight the SIGNALF program, then press [ENTER].
- Repeat the steps to enter the other three lines—plus any others you want to try.
- When you have finished entering the program, press [2nd] [QUIT] to exit programming mode.
- You can now run your program by pressing [PRGM] and selecting the name you entered.

To edit your program, simply press [PRGM], select “EDIT” with the right arrow key, and use the down arrow key to find your program name.

CONTROL

Much educational effort in mathematics and science aims to help students understand functional relationships. By combining the many input probes available for the CBL2 or LabPro with the many pieces of equipment that can be operated with the digital output, it is possible to teach about functions in a new and powerful way. This technique is effective educationally, and it is also consistent with the needs of advanced technical jobs, such as those in modern manufacturing, computer programming, robotics, and other fields. Students can learn how automated control systems operate in real-world work environments, while they also learn rigorous mathematics and science.

Connect the temperature or light probe to the CBL2's CH1 analog input port. Connect the output board to the DIG/SONIC port.

To run the program:

- press the [PRGM] key,
- use the arrow keys to select the CONTROL program from the on-screen menu,
- press the [ENTER] key twice to start.

To begin, select the option for "INPUT ONLY" and hold the temperature or light probe in different ways to see how reading changes. When you understand the range of values to expect from the probe, press any key to stop the cycle and select QUIT from the menu to exit the program.

To enter or edit the function which calculates the value at the DIG OUT port, press the [Y=] key (while the program is **not** running) and scroll down to the Y_9 function. You can enter either an equation or a logical test, such as the " $E > 25$ ". A logical test yields a value of 1 (which activates the red LED) when the test is true, and a value of 0 (turning off all the LEDs) when the test is false.

You can also enter more complex functions using "Boolean algebra." The function shown at right, for example, sometimes yields the value 2, sometimes 8 and sometimes 0—depending upon the input value from the probe in CH1. Rather than the cutoff values shown in the examples, substitute different values appropriate to the temperature or light values you found earlier.

Press [2nd][Quit] to exit the function screen and run the CONTROL program again. This time, select the "INPUT + OUTPUT" option, and explore how the probe controls your system's output. If you have time, edit the function in different ways. You might also try using multiple output devices. You could use both a fan and a bulb, for example, to design a temperature control system that both heats and cools.

```

*** CONTROL ***
READS CH1 AS E
OUTPUT IS Y9
CBL2 / LABPRO
<ENTER> TO START
CONTROL
1: INPUT ONLY
2: INPUT + OUTPUT
3: QUIT

INPUT FROM CH1:
E = 3.06

ANY KEY TO BREAK

Plot1 Plot2 Plot3
\Y5=
\Y6=
\Y7=
\Y8=
\Y9=(E<.1)*2+(E
.15)*8
\Y0=

INPUT FROM CH1:
E = 3.074

SIGNAL OUT:
Y9 = 8

ANY KEY TO BREAK

```

SIGNALW

The *SIGNALW* program, “Signal While,” is another subroutine intended for in your programs and your students’ programs. It combines key functionality of the *SIGNAL* and *CONTROL* programs above but also adds the ability to look at two input variables.

Before running *SIGNALW*, there must be an integer stored as “S,” and there must be a function stored as function Y₉. Both “S” and the Y₉ function can be entered directly from the key pad before running *SIGNALW* (just as with the *CONTROL* program). Alternatively, both can be entered within your program, as illustrated at right.

```

PROGRAM:EXB
:2→S
:String→E←u("E<2
0",Y9)
:PRGM SIGNALW
:█

** SIGNALW **
SIG S WHILE Y9≠0
= 2
(CH1)= 2.0821
(CH2)= 1.6227
T = 6.6 SEC
Y9 = 1
CBL2

```

SIGNALW looks at probes connected to both CH1 and CH2, although it is not necessary to have probes in both channels. The reading from CH1 is stored automatically as “E,” and the reading from CH2 is stored as “F.” Function Y₉ can consider either or both of these readings.

As long as Y₉ is true (not zero), the program will output signal S. When the inputs change to make Y₉ false (equal to zero) the outputs are turned off and the module releases control to the next step in your program. Note this program is similar to *SIGNAL* in that it outputs the value “S”. Y₉ is used to determine whether or not to continue sending S, not to set the value of S itself.

The *SIGNALW* screen displays the inputs, the output signal and the time that has elapsed since the program began. The final value for each of these variables remains in the calculator for you to use later in your program.

Like *SIGNALF* and most other program modules, *SIGNALW* has the option of storing the value 4 to W as a “flag” that suppresses the screen output. This can be useful, for example, if you prefer to make students responsible for measuring time themselves or if you write a program in which you want to have full control over all the screen displays.