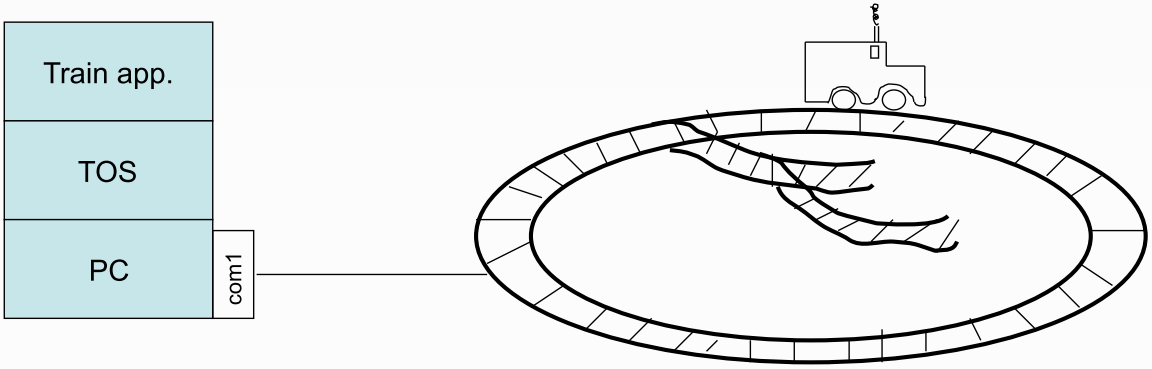
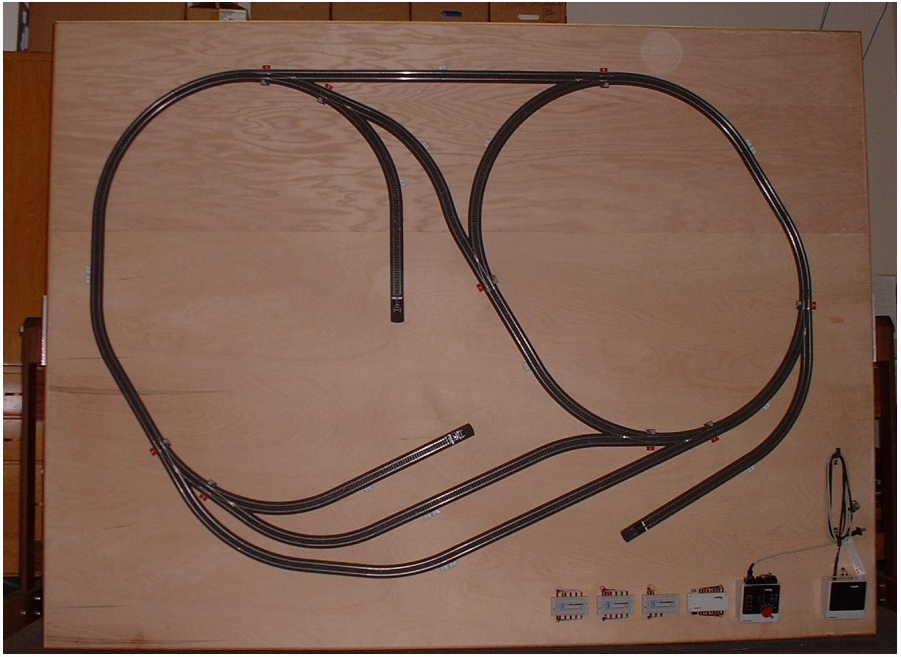
Train Application

Train Setup



- Train application runs on top of TOS
- TOS implements various operating system functions including a serial line device driver
- Serial line (com1) of the PC is connected to the train
- Commands that control the train are sent via the serial line

Model Train



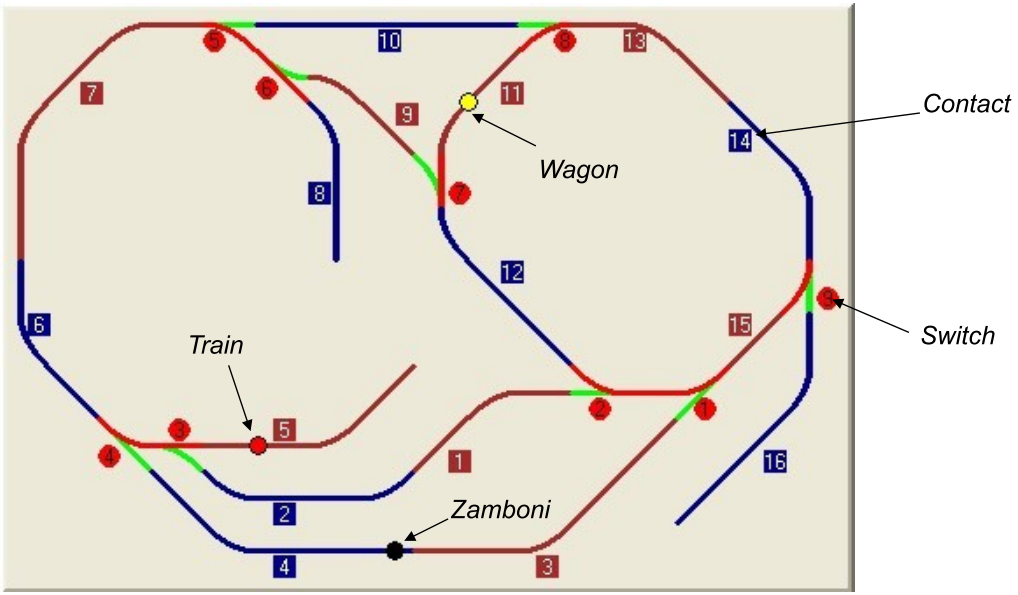
TOS Shell Commands

- The TOS Shell should understand the following commands:
  - Help function (print a list of all commands)
  - Clear the shell window (`clear_window()`)
  - Print the process table (`print_all_processes()`)
  - Make the train stop and go
  - Change the position of the switches
  - Start the PacMan game
- You can define the syntax of each of those commands.

Train Application

- There are four different configurations.
- For each configuration, Zamboni can be enabled or disabled.
- Therefore, there are 8 different permutations for the initial setup.
- Your train app has to find out at runtime, which configuration was selected (hint: use the „C“-command to probe a track segment.
- The goal is to retrieve the abandoned wagon: the train has to rendezvous with the wagon and return to the home base while avoiding Zamboni

Labels and Switch Settings



- Numbers in square are contact IDs
- Numbers in circle are switch IDs.
- Color green stands for ‘G’ setting of a switch.
  Color red stands for ‘R’ setting of a switch.

Train Commands (1)

Command	Function	Examples	Note
L#D{CR} #: vehicle ID	Change the direction of a vehicle. Train ID is 20.	To change the direction of the train: "L20D\015"	You cannot change the direction of any vehicle other than the train.
L#S#{CR} 1st #: vehicle ID 2nd #: speed, 0-5	Change the speed of a vehicle.	To change the speed of the train to 4: "L20S4\015" To stop the train: "L20S0\015"	1. You cannot change the speed of any vehicle other than the train. 2. On the real system, speeds 1-3 do not work well.

Notation:

- #: a number.
- {CR}: carriage return. (see Note 1.)

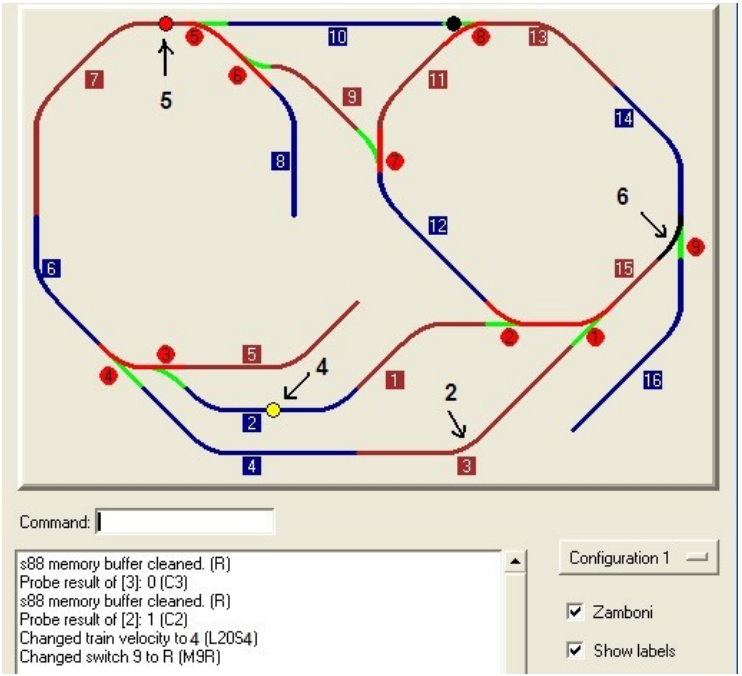
Train Commands (2)

Command	Function	Examples	Note
M#x{CR} #: switch ID X: 'R' or 'G'	Change a switch to 'G' or 'R'. 'G' and 'R' are the two possible settings of a switch.	To set switch 5 to "R": 'M5R\015'	The initial setting of a switch can be either 'G' or 'R'
R{CR}	Clear the s88 memory buffer. s88 is the device that controls the contacts.	"R\015"	This command is required for every "C" command.
C#{CR} #: contact ID	Get the status of a contact. "*1\015" is returned if there is a vehicle on the contact. Otherwise, "*0\015" is returned. (see Note 2)	To know if any vehicle is on contact 3: "C3\015"	1. A contact is a track segment with a sensor. 2. Must be preceeded by a "R" command.

An Example

Commands sent:

1. *"R\015"*
   Clear memory buffer
2. *"C3\015"*
   Probe contact 3.
   Result is 0.
3. *"R\015"*
   Clear memory buffer
4. *"C3\015"*
   Probe contact 2.
   Result is 1.
5. *"L20S4\015"*
   Set train speed to 4
6. *"M9R\015"*
   Set switch 9 to red.



Note –About the Commands

1. {CR} -- carriage return
   - The ASCII of carriage return is 13. In C, you can use ‘\015’.
   - In the simulator, if a command is from the keyboard, ‘\015’ is not needed, i.e. you only type “L20S5” to set train speed. But if a command is sent from TOS, “\015” is needed, i.e. you must send “L20S5\015” (6 characters)
2. “C” command

   The result is a string: “*0\015” or “*1\015”, where the first character is ‘*’ and the third character is a carriage return. The information you want is the second character: ‘0’ for empty contact and ‘1’ for occupied contact.
3. Pause between commands

   A pause is required between commands. You can do so by sleeping, for example, 15 ticks. (Hint, define a variable for the number of ticks to sleep. You may have to change it when you try to run on the real system)