Lab 10 is TRobots competition
Use Version 1.85 (make sure it is 2012 version)
All in C, Metrowerks Project

Set up TRobot

-Download **Trobots1_85.zip** and unzip
    Give it a simple path: **D:\Trobots1_85**
-Trobot.exe is the game engine (try it)
-Folder **S19** contains runtime files
    sx files: object code to run, **z3.sx c0.sx**
-**Trobot.ini** configuration
-**logFile.txt** results of simulation
-Folder **BattleS19** contains lots of battle buddies
    Move them into the **S19** folder when you want a battle
-Folder **Lab10** is THE starter project
    Do not make your own project

Run TRobot

-Open Starter project and observe **Watch** variables
    Execute Project->Make (F7)
-Open Folder **bin** and see **z3.sx** just created
-Start TExaS double-clicking **Texas.uc**
    Tile windows, import **z3.sx** and run
    Compare **Watch** variables with C code
-Copy from **bin** to **S19** folder
-Double click **Trobot.exe** to launch game
    Compare **Watch** variables with C code
    Notice the same tank motion in both TExaS and game

Configure TRobot

-Edit the **CopyMe** batch file in **bin** folder
-Edit **Trobot.ini** so your tank is the player
    This will allow your program to breakpoint
    Player=0 means A0, and Player=103 means Z3
-Configure this project for your tank (do this once)
    Edit->HCS12SerialMonitorSettings
    Click **Linker for HC12**
    Place tank name in **Application Filename**
    E.g., change **z3.abs** to **a0.abs**

TExaS Design Cycle

-Use TExaS to test PTT outputs to track motors
    No ADC, no turret,
    SCI will output, but no scoring
-Edit C code in Metrowerks Project
    Execute Project->Make (F7)
-Configure TExaS once
    Mode->OpenS19Mode add your C files
-In TExaS, import **z3.sx** and run
    Watch variables in Mem3800
    All the usual debugging

TRobot Design Cycle

-Add battle buddies into the **S19** folder

    c0.sx just sits there (you can make duplicates)
    Files from **S19TestFiles** will fire back

-Edit C code in Metrowerks Project

    Execute Project->Make (F7)
    asm stop adds a breakpoint

-Double-click **CopyMe** to **bin** folder

-Launch **Trobot.exe**

    Observe position, registers, **Watch** variables
    <esc> <Enter> <Enter> stops the game

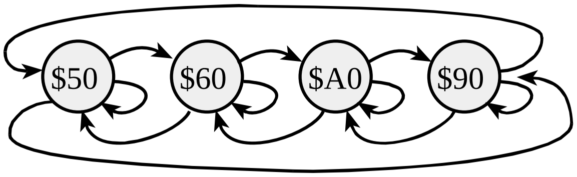
Stepper motor output sequence

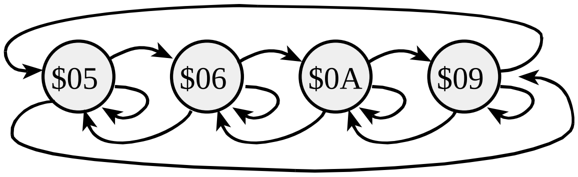
- Full-step sequence = 5,6,10,9,...
- Half-step sequence = 5,4,6,2,10,8,9,1,...
- Forward stepping causes the motor to spin forward
  - Full-step sequence = 5,6,10,9,...
  - Half-step sequence = 5,4,6,2,10,8,9,1,...
- Backward stepping causes the motor to spin backward
  - Full-step backward sequence = 9,10,6,5,...
  - Half-step backward sequence = 1,9,8,10,2,6,4,5,..

Command	Left track	Right track	Δθ (°)	Δx (m) PM7=0	Δx (m) PM7=1	Robot motion
(F,F)	full-step forward	full-step forward	0	0.5	0.75	forward
(f,f)	full-step backward	full-step backward	0	-0.5	-0.75	backward
(F,f)	full-step forward	full-step backward	-6	0	0	turn CW
(f,F)	full-step backward	full-step forward	6	0	0	CCW

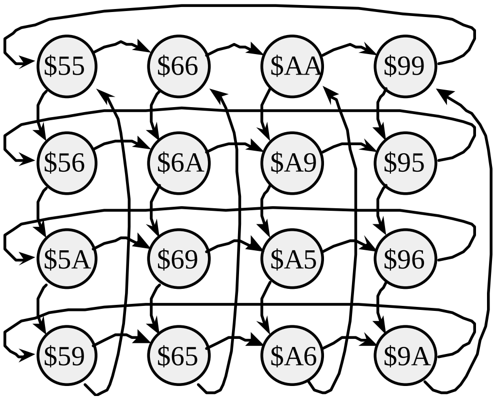
Table 3. These four full-step commands are sufficient to move the robot.

Possible data structures





Only some arrows shown



Firing cannon

- The offensive weapon is the *cannon*, which is mounted on a rotating turret. The cannon has a range of 255 meters.
- Missiles are fired in the direction of the turret. Sending a frame out the serial port fires a missile. The data value sent determines the firing range in meters.
- There are an unlimited number of missiles that can be fired, but because of the serial baud rate, there is a maximum rate at which the cannon can be fired. *(do not spin on TDRE!!)*
- Since the turret can rotate independently from the robot direction, it can fire any direction, regardless of robot heading.

The *scanner* is a ranging device in three sectors

PM5	PM4	Resolution	Left Scanner	Front Scanner	Right Scanner
0	0	5^o	+7.5^o to +2.5^o	+2.5^o to -2.5^o	-7.5^o to -2.5^o
0	1	10^o	+15^o to +5^o	+5^o to -5^o	-5^o to -15^o
1	0	30^o	+45^o to +15^o	+15^o to -15^o	-15^o to -45^o
1	1	120^o	+180^o to +60^o	+60^o to -60^o	-60^o to -180^o

Table 10.1. You can set dynamically set the scanner sensing resolution.

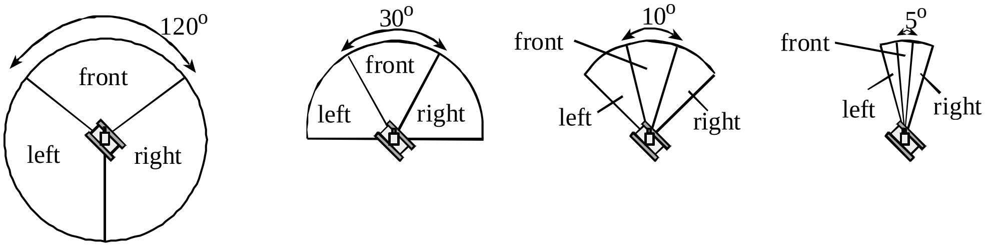


Figure 10.4. There are four possible sensing resolution for the scanner. The front direction of the scanner is set by PTP. The scanner direction is relative to the tank (not the turret)

Robot motion

- There are three stepper motors that control the robot. One stepper motor controls the left track, and a second stepper motor controls the right track. A third stepper motor rotates the gun turret.
- The smallest distance that the robot can be moved is 1 meter. The smallest angle that the robot can rotate is 1.5^o
- It takes 36 **(F,f)** commands to rotate the turret a complete 360^o

Summary of Outputs

- **Two Stepper Motors Control Robot Motion**
  - PT7-PT4 Right Track Stepper Motor
  - PT3-PT0 Left Track Stepper Motor
  - PM7 Track Stepper Motor gearbox

    (1 for fast, 0 for slow).

    You may not fire the missile in fast mode

It takes 10 bit times between a write to SCI0DRL and the eventual firing. PM7 must be low for this entire time.

- **Sensor Resolution**
  - PM5-PM4 00,01,10,11 is 5, 10, 30, 120 degrees respectively

Sensor Angle

PP7-PP0 Relative angle of the sensor to the tank, 0 to 255

0 means sensor is pointing in direction of tank (not turret)
64 means sensor pointing to tank's left
128 means sensor pointing behind the tank
192 means sensor pointing to tank's right

```
               96   64  32
                  \  |  /
                   \ | /
             128 --- x --- 0   Tank pointing this way →
                   / | \
                  /  |  \
              160   192   224
```

Figure 10.11. Sensor directions are defined relative to tank .

- **Gun Control**
  - PM3-PM0 Gun Turret Direction Stepper Motor
  - PS1 serial port, send a serial output frame to shoot a missile
  - baud rate, 1 start, 8-bit data, 1 stop frame protocol
  - 8-bit data specifies the range 0 to 255, resolution to-be-determined

Health damage

- 1% - collision caused by another robot running into you.
- 5% - collision into another robot or into a wall.
- 5% - a missile hitting your robot.
- 10% - a software bug or illegal stepper output.

Scoring

- 20 point bonus when one of your missiles hits other robot
- 1 point penalty for launching a missile
- 5 point penalty when your robot is hit by a missile
- 5 point penalty when your robot has a collision.
- A robot-robot collision causes both robots to loose points, but the robot initiating contact also looses health

TRobot.ini you can change

```
Player=2;        // Tank C0
StartX=128;      // starting X position 0 to 1023
StartY=256;      // starting Y position 0 to 1023
StartDir=512;    // starting tank direction,   256=90deg
StartTurret=256; // starting turret direction, 256=90deg
Camera=4; // 5=panoramic, 4=manual, 2=orbit, 0=1st person
ShowX=1;
ShowY=1;
ShowDir=1;
ShowTurret=1;
ShowLeft=0;
ShowCenter=1;
ShowRight=0;
ShowHealth=1;
ShowPC=1;
ShowRegX=1;
ShowRegY=1;
Show3800=1;
Show3802=1;
Show3804=1;
Show3806=1;
ShowScore=1;
BigFont=0;           // 0 is little, 1 for large
PauseBetweenGames=0; // execute without pausing
StopWillPause=1;     // stop will pause for the Player Tank
CreateLogFile=1;     // save results in logFile.txt
```

TRobot.ini you cannot change

```
WallBounce=0;        // tanks do not bounce off walls
MissileSpeed=90;     // velocity of missiles (5 to 100)
MinStepTime=400;     // min bus cycles between PTT outputs
BaudRate=2400;       // SCI0 rate in bits/sec
KillBonus=50;        // bonus points for fatal shot
LiveBonus=100;       // bonus points for staying alive
```

Hot keys for TRobot simulation

Esc	Menu
Space	Pause/Resume
Arrow keys	Camera Position
PageUp PageDown	Camera Angle
F1	Help
F5	Change Screen Resolution
F6	Screenshot
F7	Toggle camera mode
F8	Switch to next player
F12	Toggle sound/music
F11	Abort this run, and start another run
F10	Exit

The grading scale for the TRobot Lab

110	>75% scoring rank and software has good structure and style.
100	50-75% scoring rank and software has good structure and style.
90	50-100% scoring rank and software has poor style.
85	25-50% scoring rank and software has good structure and style.
75	25-50% scoring rank and software has poor style.
75	0-25% scoring rank and software has good structure and style.
50	0-25% scoring rank and software has poor style.

***If you don't have significant/good code then irrespective of your rank you may get as low as 0 points for lab 10**

Schedule of Events, Spring 2012

- **Program submission.** By Wednesday 5/2 2pm: Assume you are team X1. You must name your project X1 so it creates **X1.sx** files when it compiles. You must create one zip file with the entire Metrowerks project, with a name that includes both EIDs. E.g., **ABC123_DEF456.zip**, where ABC123 and DEF456 are the UT EIDs of two students. Upload ABC123_DEF456.zip to the Blackboard account of both students. Test the process by emailing the zip file to your partner, unzipping it on a different computer, compiling it, and looking to see if it creates a new **X1.sx** file (if your partner can't compile it, neither can I). I will post the list of tank names I successfully compiled on BB by 11am Thursday 5/3. If you do not see your tank, bring a flash drive to class on Thursday. Great confusion and sadness will occur if you upload two copies of your program.
- **Final Contest** 5/3 in class: I will host the final TRobot competition. Note: there will be about the 30 robots in the battlefield. I will run two batches. The top team, as scored by the total points of all the runs in the final competition, will be get a prize.

Layered solution

Low level

Move forward, Move back, Turn 90 degrees CW
Turn 90 degrees CCW, Turn to North

High level

Game algorithm

Search for opponents, Target and fire
Defense, Lifepacks

Feedback

Read sensors

(x,y) position, ADC channels 0,1
Heading angle, ADC channel 2