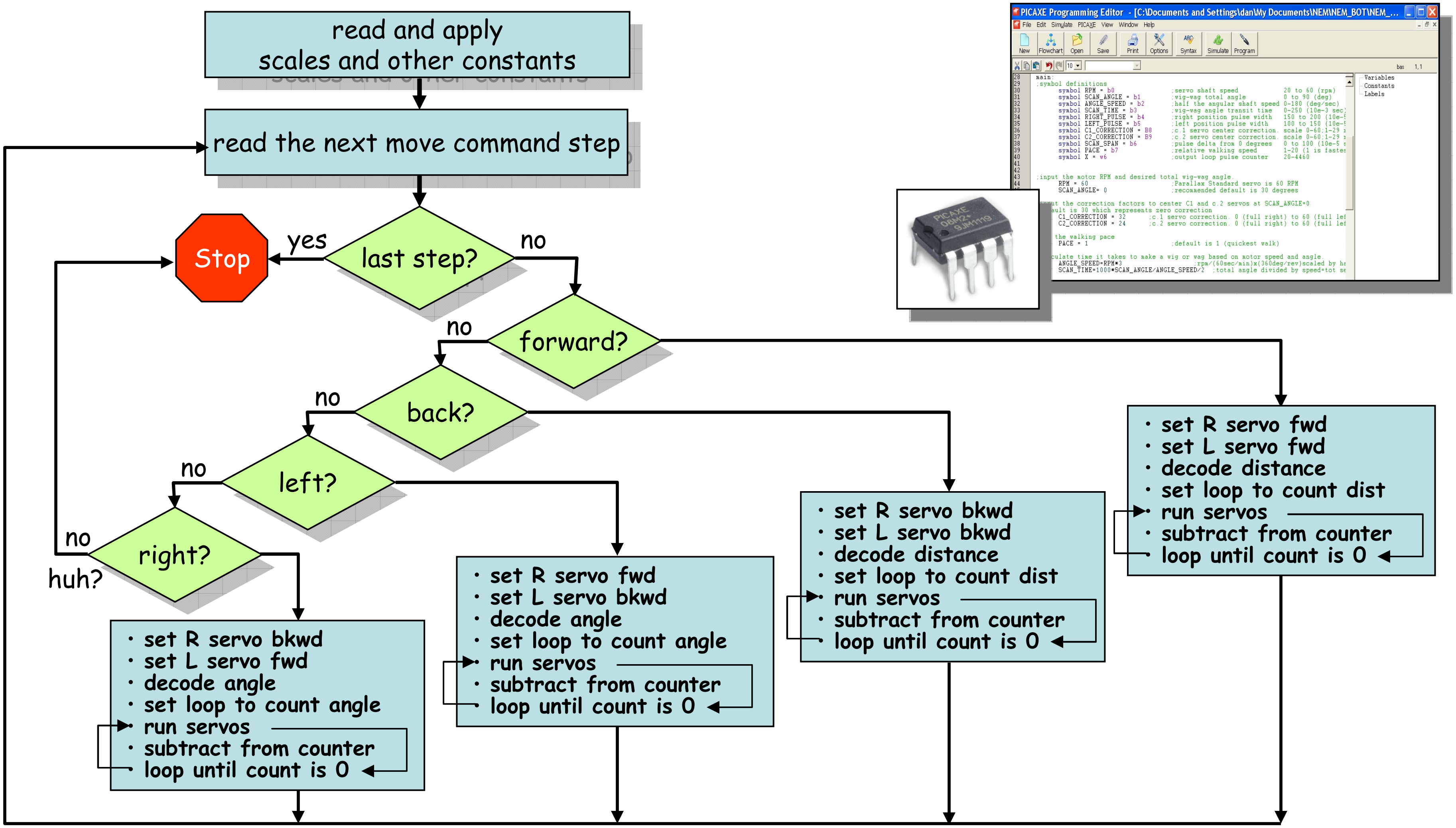
NEMBot II Program

The program decodes move commands and runs the Bot's servos



```
main:
;symbol definitions
        symbol RPM = b0                 ;servo shaft speed            20 to 60 (rpm)
        symbol SCAN_ANGLE = b1          ;wig-wag total angle          0 to 90 (deg)
        symbol ANGLE_SPEED = b2         ;half the angular shaft speed 0-180 (deg/sec)
        symbol SCAN_TIME = b3           ;wig-wag angle transit time   0-250 (10e-3 sec)
        symbol RIGHT_PULSE = b4         ;right position pulse width   150 to 200 (10e-5
        symbol LEFT_PULSE = b5          ;left position pulse width    100 to 150 (10e-5
        symbol C1_CORRECTION = B8       ;c.1 servo center correction. scale 0-60;1-29 r
        symbol C2_CORRECTION = B9       ;c.2 servo center correction. scale 0-60;1-29 r
        symbol SCAN_SPAN = b6           ;pulse delta from 0 degrees   0 to 100 (10e-5 s
        symbol PACE = b7                ;relative walking speed       1-20 (1 is fastes
        symbol X = w6                   ;output loop pulse counter    20-4460


;input the motor RPM and desired total wig-wag angle.
        RPM = 60                        ;Parallax Standard servo is 60 RPM
        SCAN_ANGLE= 0                   ;recommended default is 30 degrees

  ut the correction factors to center C1 and c.2 servos at SCAN_ANGLE=0
   ault is 30 which represents zero correction
        C1_CORRECTION = 32      ;c.1 servo correction. 0 (full right) to 60 (full lef
        C2_CORRECTION = 24      ;c.2 servo correction. 0 (full right) to 60 (full lef

    the walking pace
        PACE = 1                        ;default is 1 (quickest walk)

   culate time it takes to make a wig or wag based on motor speed and angle.
        ANGLE_SPEED=RPM*3                         ;rpm/(60sec/min)x(360deg/rev)scaled by ha
        SCAN_TIME=1000*SCAN_ANGLE/ANGLE_SPEED/2  ;total angle divided by speed=tot se
```

Alignment Check & Programming

Align your Bot so it will correctly follow your commands

□ This initial program is stored in the Bot:

- Wait 5 seconds
- Forward 36”
- Left 90°
- Right 90°
- Backward 36”
- Right 45°
- Right 45°
- Left 90°

□ Check to see if the Bot moves correctly

- Look for drifting to one side and for correct distances & angles
- Fix one thing at a time, starting with drift then distance and angle
- Note the error (like 3” to far out of 36”), then correct it
- A course conductor will help you make adjustments

NEMBot II Motion Programming

NATIONAL ELECTRONICS MUSEUM

Programming requires a few actions

- ❑ Decide the movements you want the Bot to make
  - ▪ Write down the steps in order
  - ▪ Convert steps into program move codes
  - ▪ Enter codes into your Picaxe® program using the PicAxe® Editor
- ❑ There are other *optional* codes too
  - ▪ An initial pause after the Bot is switched on
    - Default is 5 seconds
  - ▪ Pauses before and after all turns
    - Default is 500mS (0.5 sec)
- ❑ A course conductor will help you with programming

NEMBot II Motion Programming con’d

NATIONAL ELECTRONICS MUSEUM

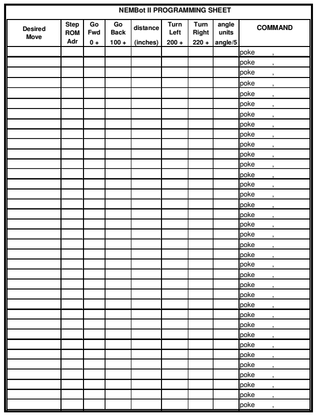
Expressing moves as numeric commands is not complicated

- ❑ Each command is a single number from 1 to 220
- ❑ Each command is the sum of a direction code and a distance
- ❑ There are four direction codes
  - move forward   direction code 0   + distance can be 1-100 inches
  - move back   direction code 100   + distance can be 1-100 inches
  - turn left   direction code 200   + distance can be 5-100 degrees*
  - turn right   direction code 220   + distance can be 5-100 degrees*
  - Just add up the direction code and desired distance to create the command

Examples

- move forward 24”   0 + 24 =   24
- move back 24”   100 + 24 = 124
- turn Left 90 degrees   200 + 18* = 218

*The angle scale is 5 deg/unit, so 90 degrees = 18 ‘distance’ units (18x5=90)



NEMBot II PROGRAMMING SHEET

Desired Move	Step ROM Adr	Go Fwd 0 +	Go Back 100 +	distance (inches)	Turn Left 200 +	Turn Right 220 +	angle units angle/5	COMMAND
								poke ,
								poke ,
								poke ,
								poke ,
								poke ,
								poke ,
								poke ,
								poke ,
								poke ,
								poke ,
								poke ,
								poke ,
								poke ,
								poke ,
								poke ,
								poke ,
								poke ,
								poke ,
								poke ,
								poke ,
								poke ,
								poke ,
								poke ,
								poke ,
								poke ,
								poke ,
								poke ,
								poke ,
								poke ,
								poke ,
								poke ,
								poke ,
								poke ,
								poke ,
								poke ,
								poke ,

Programming your Bot

The programming sheet will help you write the motion code

- ❑ Remember that each move command is a single number
- ❑ Move commands are entered into sequential memory locations
  - The first memory location is always 30
  - “poke” is microcontroller code for “put into memory”
  - A program move command step looks like this (the *syntax*):

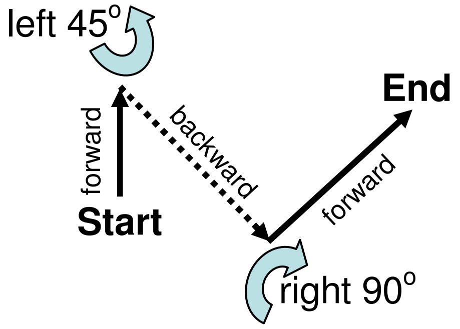
```
poke [memory location], [command]
        poke 34, 124      (move back 24 inches)
```

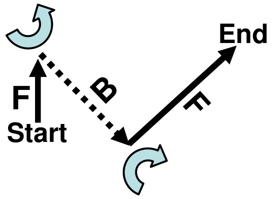
- ❑ The programming sheet will help make programming easier

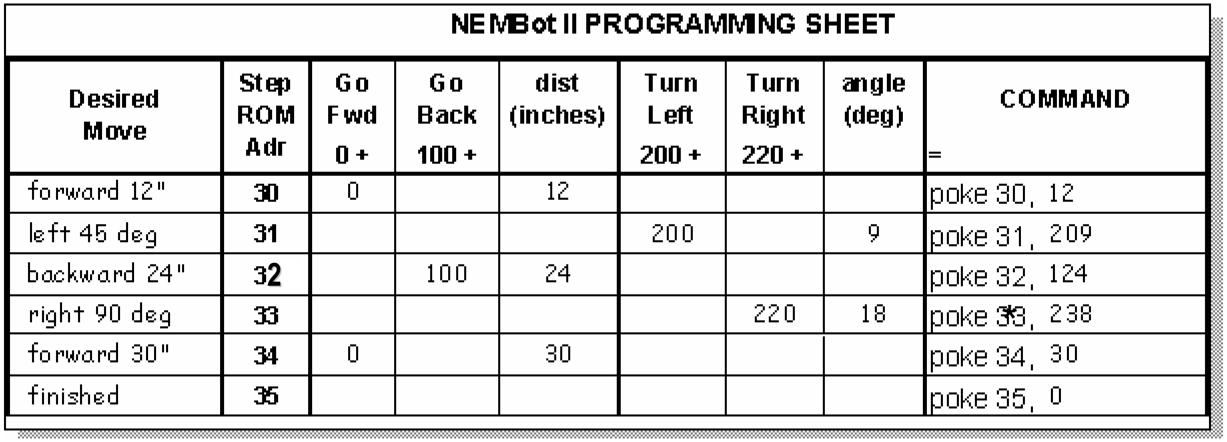
Programming your Bot - continued

The programming sheet will help you write the motion code

Example “dance”







NEMBot II PROGRAMMING SHEET

Desired Move	Step ROM Adr	Go Fwd 0 +	Go Back 100 +	dist (inches)	Turn Left 200 +	Turn Right 220 +	angle (deg)	COMMAND =
forward 12"	30	0		12				poke 30, 12
left 45 deg	31				200		9	poke 31, 209
backward 24"	32		100	24				poke 32, 124
right 90 deg	33					220	18	poke 33, 238
forward 30"	34	0		30				poke 34, 30
finished	35							poke 35, 0

More . . .

Workshop materials form a basis for more projects

- Expand NEMBot II with added sensors
  - Bump, SONAR, IR, Sound . . .
- New robot projects
- Resources
  - www.picaxe.com – Program Editor, cable driver, and programming info
  - www.picaxeforum.co.uk/forum - ideas and help
  - www.parallax.com , www.futabarc.com , www.hitecrcd.com
    - manufacturer sites – servos and accessories
  - www.sparkfun.com – *servos, accessories, and robot parts
  - www.servocity.com – servos, accessories, and robot parts
  - www.pololu.com – servos, accessories, and robot parts
  - www.digikey.com , www. jameco.com – servos and electronics parts
  - www.instructables.com - ideas and general info
    - projects using other microcontrollers (PICs) can be adapted
- Don’t forget Robofest in two weeks on April 27 *!*

*good source for AXE027 programming cable

. . . Perhaps attend another NEM workshop . . .