Overall Goals

• Learn fundamental MSN commands: ON, OFF, #START, STOP..., #R, ADD, SHOW, ZN, #ZN, SX

Rules: Statements

1: O ---> T

• Statements must have three components:
  – Input Section
  – Output Section
  – Transition

INPUT: OUTPUT ---> TRANSITION

• Input Section is to left of colon
• Output Section is between colon and arrow “--->”
• Transition is to right of arrow
ON Command

- **ON**
- Output command used to turn On the specified Output
- **Syntax:** `ON P1`
  - Where P1 = number, constant, variable, array element, or mathematical expression
- **Example:**
  
  After a response on Input 1, turn ON Output 1 and the Pellet Feeder, then go to State 2.

  S.8.1, 
  S1,  
  #R1: ON 1, "PelletFeeder ---> S2

OFF Command

- **OFF**
- Output command used to turn Off the specified Output
- **Syntax:** `OFF P1`
  - Where P1 = number, constant, variable, array element, or mathematical expression
- **Example:**

  After a response on Input 1, turn ON Output 1 and the Pellet Feeder, then go to State 2.
  After 4 seconds, turn OFF Output 1 and the Pellet Feeder and go back to State 1.

#R Command

- **#R**
- Input command used to measure responses on the specified Input (e.g. lever, nose poke, lickometer)
- **Syntax:**
  ```
  [Number of times] #R[Device that is collecting input]: OUTPUT ---> Transition
  ```
  - Where [Number of times] = number of times the response must happen (default = 1)
  - [Device that is collecting input] = number, constant, variable, array element, or mathematical expression
- **Examples:**

  After a response on the Left Lever, turn ON the Pellet Feeder, then go to State 2.

  #R"LeftLever": ON "PelletFeeder ---> S2

  After 3 responses on the Center Nose Poke, turn ON the Foot Shock, then go to State 4.

  #R"CenterNosePoke": ON "FootShock ---> S4

  After a response on Input 1, turn ON Output 1 and the Pellet Feeder, then go to State 2.

  S.8.1, 
  S1,  
  #R1: ON 1, "PelletFeeder ---> S2

  After 4 seconds, turn OFF Output 1 and the Pellet Feeder and go back to State 1.
**ADD Command**

- **ADD**
- Output command that increments a variable by one unit.
- **Syntax:**
  
  \[
  \text{INPUT: ADD X} \rightarrow \text{TRANSITION}
  \]
  
  Where \( X \) = the variable to which the value 1 will be added.
- **Example:**

  After one second, add 1 to the variable \( C \), then go to State 2.

**SHOW Command**

- **SHOW**
- Output command that displays information in the Show Area of the MED PC® window.
- **Syntax:**
  
  \[
  \text{INPUT: OUTPUT; SHOW P, Label, X} \rightarrow \text{TRANSITION}
  \]
  
  Where \( P \) = Position in Show Area (1 to 200)
  
  Label = User-defined name for that position, must begin with letter
  
  \( X \) = Variable value listed in Show Position.
- **Example:**

  After 1 response on Input 2, add 1 to variable \( A \), and display the value of variable \( A \) in position 1. Label this value "Center Key". Then go to State 3.

**SX (Null) Transition**

- **SX**
- Special Transition that "goes nowhere" — i.e. the State Set remains in the State it is in.
- **Syntax:**
  
  \[
  \text{INPUT: OUTPUT} \rightarrow \text{SX}
  \]
- **Example:**

  After 1 minute, add 1 to variable \( M \), display the value of \( M \) in position 3 with the label "Minutes", and go nowhere.
Introduction to MED PC Programming

#START Command

- **#START**
- Input command that holds initiation of a procedure until a Start Signal is given by the researcher.
- Syntax: 
  
  ```
  #START: OUTPUT --> TRANSITION
  ```

Example:

```plaintext
#START: ON "HouseLight" ---> S2
```

Wait for a Start Signal. When a Start Signal is issued, turn On the House Light and go to State 2.

STOPABORT Transition

- **STOPABORT**
- Special Transition that causes a program to immediately stop running. Any active Outputs are turned Off. Any data collected up to the STOPABORT command remains in memory and can be saved or printed.
- Syntax: 
  
  ```
  INPUT: OUTPUT --> STOPABORT
  ```

Example:

```plaintext
S2, 60': ---> STOPABORT
```

After 60 minutes, stop the program and keep all data in memory.

STOPKILL Transition

- **STOPKILL**
- Special Transition that causes a program to immediately stop running. Any active Outputs are turned Off. Any data collected is wiped from memory and cannot be recovered.
- Syntax: 
  
  ```
  INPUT: OUTPUT --> STOPKILL
  ```

Example:

```plaintext
S2, 1': ---> STOPKILL
```

After 1 minute, stop the program and flush data from memory.
**STOPABORTFLUSH Transition**

- **STOPABORTFLUSH**
  - Special Transition that causes a program to immediately stop running. Any active Outputs are turned off. Any data collected up to the STOPABORTFLUSH command is saved to the hard disk and then the memory is wiped clean.
  - Syntax: INPUT OUTPUT \( \rightarrow \) STOPABORTFLUSH
  - Example:

    After 90 minutes, stop the program, save the data to the hard disk, then wipe the memory clean.

    S2: 90 : \( \rightarrow \) STOPABORTFLUSH

**Multiple Commands**

- Sometimes it is necessary to have more than one Output command strung together. This can be done using semicolons (\( ; \)) to separate individual Output commands.
- Syntax: INPUT: OUTPUT\#1; OUTPUT\#2 \( \rightarrow \) TRANSITION
- Example:

  After 10 responses on the Right Lever, Add 1 to the variable C and display the value of variable C in position 4 with the label "Rewards" then remain in the current State.

  S1L\( ^{10} \) RightLever: ADD C; SHOW 4, Rewards, C \( \rightarrow \) SX

**Z-Pulse Commands**

- **Z-Pulses**
  - Z-Pulses are special commands that can function as either Inputs or Outputs. They are used to communicate between State Sets.
  - Syntax:
    - INPUT: ZN \( \rightarrow \) TRANSITION \( \downarrow \) Z-Pulse acting as Output
    - #ZN: OUTPUT \( \rightarrow \) TRANSITION \( \downarrow \) Z-Pulse acting as Input
  - Where \( N \) = integer between 1 and 32. Must be same in both Statements.
- Example:

  Start Command in State Set 1 turn on House light and issues a Z-Pulse. This Z-Pulse then initiates a transition to State 2 of State Set 2.

  S1S1, S2, S1L #START ON "HouseLight"; S1 \( \rightarrow \) S2
  S1S2, S1, #S1 \( \rightarrow \) S2
Rules: Z-Pulses

I: ZN --&gt; T
#ZN: O --&gt; T

• Similar to State Sets and States, Z-Pulses must be numbered
  – Integer between 1 and 32
  – Numbering does not need to be sequential, they are processed in
    the order in which they occur.
• Sequential numbering is recommended to reduce confusion

Sample Program

Counting Lever Presses, Controlling Program Flow, and Displaying Information on the MED PC® Screen

• Wait for a Start Command, then turn on the House Light.
• We want an FR-5 schedule, so turn on the Pellet Feeder after 5 Left Lever Presses.
• Count the Left Lever Presses using a Variable, A, and display the counts in
  Position 1, labeled “Left Lever Presses”.
• Count Rewards using a Variable, B, and display the counts in Position 2, labeled “Rewards”.
• Turn the Pellet Feeder Off after 0.05 seconds.
• End the program after 1 minute, but leave the data in memory.

TIPS:
• Try using four State Sets with two States each.
• One State Set will be the FR-5 code, another State Set will be a
  Response Counter, another State Set will be a Reward Counter
  and Timer, and the last State Set will be a Session Timer.

Sample Program

This is a sample program
This is an FR-5 program
Filename: MEDSCN_04.mpc
Date: July 29, 2009
Author: G. Herrera
This section is for Outputs
^PelletFeeder = 3       \ My pellet feeder is wired to Output 3
^HouseLight = 7         \ My house light is wired to Output 7
This section is for Inputs
^LeftLever = 1          \ My left lever is wired to Input 1
Variables used in this program
A = Left Lever Response Counter
B = Reward Counter
S.S.1,                  \ Main Control Logic for FR
S1,                   \ Start
#START: ON ^HouseLight ---> S2
S2,                   \ 5 lever presses
5#R^LeftLever: ON ^PelletFeeder; Z1 ---> SX

Your Program should look something like this:

This is a sample program
This is an FR-5 program
Filename: MEDSCN_04.mpc
Date: July 29, 2009
Author: G. Herrera
This section is for Outputs
^PelletFeeder = 3       \ My pellet feeder is wired to Output 3
^HouseLight = 7         \ My house light is wired to Output 7
This section is for Inputs
^LeftLever = 1          \ My left lever is wired to Input 1
Variables used in this program
A = Left Lever Response Counter
B = Reward Counter
S.S.1,                  \ Main Control Logic for FR
S1,                   \ Start
#START: ON ^HouseLight ---> S2
S2,                   \ 5 lever presses
5#R^LeftLever: ADD A; SHOW 1, Left Lever Presses, A ---> SX
S.S.2,                 \ Response Counter and Display
S1,                   \ Start
#R^LeftLever: ADD A; SHOW 1, Left Lever Presses, A ---> SX
S.S.3,                 \ Reward Timer, Count, and Display
S1,                   \ Start
Z1: ADD B; SHOW 2, Rewards, B ---> S2
S2,                   \ 0.05": OFF ^PelletFeeder ---> S1
S.S.4,                  \ Session Timer
S1,                   \ Start
#START: ---> S2
S2,                   \ 1": ---> STOPABORT
- Manually Issuing a START Command
- Select Configure – Signals, then issue a Start Command

- What happens when you push the Left Lever?
- How can you tell if a Reward is delivered?
- Does the Right Lever do anything?
- What happens after 1 minute?
- What happens if you try to Close MED PC®?
Now You Know the Basics

- Try doing your own programming
- Try using State Sets like Tool Boxes:
  - Session timers, reinforcer counters, response counters
- Study the MED PC® Programmers Guide
- Utilize the On-Line Help file in MED PC®
- We’re here to help

Where to Go for Help?

- support@med-associates.com
- Ph: (802) 527-2343
- Fax: (802) 527-5095
- www.med-associates.com

Researcher’s Round Table Discussion Forum
Where to Go for Help?
Med State Notation™ Repository
http://www.med-associates.com/mednr/forum/

Thank You!
Good Luck!