remember SBC if there is a borrow carry is CLEARED
also SBC if the two numbers are equal you still get a negative result

MODIFICATION LIST :

Furby29/30/31/32
Final testing for shipment of code on 8/2/98.
Tables updated. Motor speed updated, wake up/name fix
sequential tables never getting first entry, fixed.
New diag5.asm, Light3.asm (if light osc stalls it won't hang
system).

Furby33
In motor brake routine, turn motors off before turning reverse
braking pulse on to save transistors.

Furby34
Cleanup start code and wake routines.
Light sensor goes max dark and stays there to raff time, than
call sleep macro and shut down.

Furby35
Adds four new easter eggs. BURP ATTACK, SAY NAME, TWINKLE SONG,
and ROOSTER LOVES YOU. Also add new names.
1. Light sensor has a hysteresis point of continually triggering sensor.
2. Light sensor decrements two instead of one on hungry counter.
3. Diagnostic de for light sensor won't trigger very easily.
4. When a fuzzy receives the I.R. sleep command he sends the same command out before going to sleep.
5. When hungry is enough to trigger sick counter, each sensor deducts two instead of one for each hit.
6. When diagnostics complete clear memory, reset hungry & sick to FF randomly choose new name & voice, then writes EEPROM before going to sleep. Also extend EEPROM diagnostic to test all locations for pass/fail of device.
7. Add new light routine
8. Change hide and seek egg to light, light, light, tummy.
9. Change sick/hungry counter so that it can only get so sick and not continue down to zero. (MAX_SICK)
10. In diagnostics, motor position test, first goes forward continuously until the front switch is pressed, then goes reverse continuously until the front switch is pressed again, and then does normal position calibration stopping at the calibration switch.
11. On power up we still use tilt and invert to generate startup random numbers, but if feed switch is pressed for cold boot, we use it to generate random numbers, because it is controlled by the user where the tilt and invert are more fickle.
12. No matter what age, 25% of time he randomly pulls speech from age to generate more Furibish at older ages.
13. Twinkle song egg
   When song is complete, if both front and back switches are pressed we goto deep sleep. That means only the invert can wake us up, not the tilt switch.
Actual numeric value for TI pitch control

- bit 7 set = subtract value from current course value
- clr = add value to current course value
- bit 6 set = select music pitch table
- clr = select normal speech pitch table
- bit 0-5 value to change course value (no change = 0)

A math routine in 'say_0' converts the value for + or -
if < 80 then subtracts from 80 to get the minus version of 00
i.e., if number is 70 then TI gets sent 10 (which is -10)
If number is 80 or > 80 then get sent literal as positive.

NOTE: MAX POSITIVE IS 8F (+16 from normal voice of 00)
MAX NEGATIVE is 2F (-47 from normal voice of 00)

This is a difference of 51h

8Fh is hi voice (8f is very squeeeeeeek)
2Fh lo voice (very low)

The math routine in 'say_0' allows a ± decimal number in the speech table.
A value of 80 = no change or 00 sent to TI
81 = +1
8f = +16
A value of 7F = -1 from normal voice
70 = -15

The voice selection should take into consideration that the hi voice
selection plus an additional offset is never greater than 8f
Or a low voice minus offset never less than 2f.

Voice1 EQU 83h ; (+3) hi voice
Voice2 EQU 7Ah ; (-6) mid voice
Voice3 EQU 71h ; (-15) low voice

we converted to a random selection table, but since all voice
tables
use the equates plus some offset, we change in the SAY_0
routine. We always assign voice 3 which is the lowest, and based on
the random power up pitch selection, the ram location 'Rvoice'
holds
the number to add to the voice+offset received from the macro
table.

Voice EQU Voice3 ;pitch (choose Voice1, Voice2,
Voice3) (voice2+norm)

Select Voice1 since it is the lowest and then add the difference to
get
Voice2 or Voice3. Here we assign that difference to an equate to be
used in the voice table that is randomly selected on power up.

Voice1 EQU 18 ;Voice3 + 18d = Voice1
Voice2 EQU 09 ;Voice3 + 09d = Voice2
S_voice3 EQU 0 ; Voice3 + 00d = Voice3

; Motor speed pulse width:
; Motor_on = power to motor, Motor_off is none.

Mpulse_on EQU 16 ;
Mpulse_off EQU 16 ;

Cal_pos_fwd EQU 134 ; calibration switch forward direction
Cal_pos_rev EQU 134 ; calibration switch forward direction

; PORTS
; SPC40A has: 16 I/O pins
; PORT_A 4 I/O pins 0-3
; PORT_C 4 I/O pins 0-3
; PORT_D 8 I/O pins 0-7

; RAM
; SPC40A has: 128 bytes of RAM
; from $80 - $FF

; ROM
; SPC40A has:
; BANK0 user ROM from $0600 - $7FFF
; BANK1 user ROM from $8000 - $FFFF

; VECTORS
; NMI vector $7FFA / $7FFB
; RESET vector $7FFC / $7FFD
; IRQ vector $7FFE / $7FFF

; PORTS
; SPC120A has: 17 I/O pins
; PORT_A 4 I/O pins 0-3
; PORT_B 4 I/O pins 0, 1, 2, 4, 5
; PORT_C 4 I/O pins 0-3 input only
; PORT_D 8 I/O pins 0-7

; RAM
; SPC120A has: 128 bytes of RAM
; from $80 - $FF

; ROM
; SPC120A has:
; VECTORS

; NMI vector $7FFA / $7FFB
; RESET vector $7FFC / $7FFD
; IRQ vector $7FFE / $7FFF

unuseable areas in rom

; SPC40A: 8000H - DFFFH should be skiped (Dummy area)
; bank 0 = 600 - 7FFA
; bank 1 = 8000 - DFFF reserved , start @ E000 - FFFA

; SPC80A: 10000H - 13FFFH should be skiped (Dummy area)
; bank 0 = 600 - 7FFA
; bank 1 = 8000 - DFFF
; bank 2 = 10000 - 13FFF reserved , start at 14000 - 17FFF

; SPC120A: SPC120A: 18000H - 19FFFH should be skiped (Dummy area)
; bank 0 = 600 - 7FFA
; bank 1 = 8000 - DFFF
; bank 2 = 10000 - 17FFF
; bank 3 = 18000 - 19FFF reserved , start at 1A000 - 1FFF

; SPC256A: ; SPC256A: Non dummy area

; SPC512A: ; SPC512A: Non dummy area

 CODE
 .SYNTAX 6502
 .LINKLIST
 .SYMBOLS

; PORT DIRECTION CONTROL REGISTER

Ports_dir EQU 00 ; (write only)

; (4 I/O pins) controlled with each bit of this register
; you can't control each pin separately, only as a nibble
; 0 = input / 1 = output

; REGISTER BITS
; D D C C B B A A (PORT)
; 7654 3210 7654 3210 7654 3210 7654 3210 (PORT BITS)

; PORT CONFIGURATION CONTROL REGISTER

;
based on if the port pin is input or output

Ports_con EQU 01 ; (write only)

; (4 I/O pins) controlled with each bit of this register
 ; 7 6 5 4 3 2 1 0

BITS)

; D  D  C  C  B  B  A  A

; 7654 3210 7654 3210 7654 3210

; port_a INPUTS can be either:
; 0 = float 1 = pulled high

; port_a OUTPUTS can be either:
; 0 = buffer 1 = upper (4) bits Open drain Pmos (source)
; lower (4) bits Open drain Nmos (sink)

; port_b INPUTS can be either:
; 0 = float 1 = pulled low

; port_b OUTPUTS can be either:
; 0 = buffer 1 = upper (4) bits Open drain Nmos (sink)
; lower (4) bits Open drain Nmos (sink)

; port_c INPUTS can be either:
; 0 = float 1 = pulled high

; port_c OUTPUTS can be either:
; 0 = buffer 1 = upper (4) bits Open drain Pmos (source)
; lower (4) bits Open drain Nmos (sink)

; port_d INPUTS can be either:
; 0 = float 1 = pulled low

; port_d OUTPUTS can be either:
; 0 = buffer 1 = Open drain Pmos (source)

; I/O PORTS

Port_A EQU 02H ; (read/write) for TI & speech recgn
CPU's

Data_D0 EQU 01H ; bit 0 data nibble port
Data_D1 EQU 02H ;
Data_D2 EQU 04H ;
Data_D3 EQU 08H ;

Port_B EQU 03H ; b0/b1 = I/O b4/b5 = irq only

T_init EQU 01H ; B0 = TI reset control
TI_CTS EQU 02H ; B1 = hand shake to TI
IR_IN EQU 10H ; B4 - I.R. Recev data
TI_RTS EQU 20H ; B5 = TI wants data

Port_C EQU 04H ; (read/write)

Motor_cal EQU 01H ; C0 - lo when mot + crosses switch
Pos_sen EQU 02H ; C1 - motor ical sensor (intt C1)
Touch_bck EQU 04H ; C2 - back touch
Touch_frt EQU 08H ; C3 - front touch
Port_D     EQU  05H ; (read/write)
Ball_side  EQU  01H ;D0 - hi when on any side (TILT)
Ball_invert EQU  02H ;D1 - hi when inverted
Light_in   EQU  04H ;D2 - hi when bright light hits sensor
Mic_in     EQU  08H ;D3 - hi pulse microphone input
Power_on   EQU  10H ;D4 - power to rest of circuit
Motor_ led  EQU  20H ;D5 - motor I.R. led driver
Motor_d  EQU  40H ;D6 - motor drive left (forward)
Motor_r  EQU  80H ;D7 - motor drive right (reverse)

;======================================= DATA LATCH PORT_D
;=======================================
Latch_D    EQU  06H ; (read) ; read to latch data from port_d, used for wake-up on pin change
;=======================================

;======================================= BANK SELECTION REGISTER
;=======================================
Bank       EQU  07H ; (read/write) x x x x x x b 
; 0 = bank 0, 1 = bank 1 ; only two banks in SPC40a
;=======================================

;======================================= WAKE UP
;=======================================
Wake_up    EQU  08H ; (read/write) x x x x x x w
; w=(0=disable, 1=enable wake-up on port_d change)
; read to see if wake-up, or normal reset
; this is the only source for a wake-up
; Always reset stack on wake-up.
;=======================================

;======================================= SLEEP
;=======================================
Sleep      EQU  09H ; (write) x x x x x x e
; e=(0=don't care, 1=stop)
; writing 1 to bit0, 1 as sleep
;=======================================

;======================================= TIMER A CONTROL REGISTER
;=======================================
TMA_CON    EQU  0BH ; (write)
; 7 6 5 4 3 2 1 0
; m x x x
; m= Timer one mode (0=Timer, 1=Counter)
Bit3: IE1 & IE1 = 0: Counter clock = external clock from IOC2
Bit2: T1 = 1, T1 = 0: counter clock = CPUCLK/8192
Bit1: IE0 A' = 1: counter clock = CPUCLK/65536
Bit0: TO = 0: Counter clock = external clock from IOC2
          = 1, TO = 0: counter clock = CPUCLK/4
          = 1: counter clock = CPUCLK/64

; INTERRUPTS
INTERRUPTS EQU ODH ; (read/write)

7 6 5 4 3 2 1 0
w m e b l 3 2 1 e

w = (0=watch dog ON, power-on default) (1=watch dog OFF)
m = (0=Timer A generates NMI INT, 1=Timer A generates IRQ INT)
a = (0=Timer A interrupt off, 1=Timer A interrupt on)
b = (0=Timer B interrupt off, 1=Timer B interrupt on)
3 = (0=CPU CLK/1024 interrupt off, 1=CPU CLK/1024 interrupt on)
2 = (0=CPU CLK/8192 interrupt off, 1=CPU CLK/8192 interrupt on)
1 = (0=CPU CLK/65536 interrupt off, 1=CPU CLK/65536 interrupt on)
e = (0=external interrupt off, 1=external interrupt on)
rising edge, from port_c bitl

; TIMERS
There are two 12bit timers.
Timer A can be either a timer or a counter, (as set by TIMER_CON)
Timer B can only be used as a timer.

Timers count-up end on overflow from 0FFF to 0000, this carry bit will
create an interrupt if the corresponding bit is set in INTERRUPTS
register.
The timer will be auto reloaded with the user setup value, end
start, count-up again.
Counter will reset by user loading #00 into register TMA_LSB and
TMA_MSB.
Counter registers can be read on-the-fly, this will not effect
register, values, or reset them.

; TIMER A (low byte)
TMA_LSB EQU 10H ; (read/write)

; all 8bits valid (lower 8bits of 12bit timer)
;----------------------------------- TIMER A (high byte) -----------------------------------

TMA_MSB EQU 11H ; (read/write)
; read
x x x x 11 10 9 8 timer upper 4bits
7 6 5 4 3 2 1 0
;
; write
x x t c 11 10 9 8 timer upper 4bits
7 6 5 4 3 2 1 0 register bit
;
; t=0=speech mode, 1=Tone mode
; this connects the AUDIO pin to either
; the DAC2, or Timer generated square wave
;
; c=0=CPU clock, 1=CPU clock/4:

;----------------------------------- TIMER B (low byte) -----------------------------------

TMB_LSB EQU 12H
; all 8bits valid (lower 8bits of 12bit timer)

;----------------------------------- TIMER B (high byte) -----------------------------------

TMB_MSB EQU 13H
; read
x x x x 11 10 9 8 timer upper 4bits
7 6 5 4 3 2 1 0
;
; write
x x t c 11 10 9 8 timer upper 4bits
7 6 5 4 3 2 1 0 register bit
;
; t=0=speech mode, 1=Tone mode
; this connects the AUDIO pin to either
; the DAC2, or Timer generated square wave
;
; c=0=CPU clock, 1=CPU clock/4:

;----------------------------------- D/A converters -----------------------------------

DAC1 EQU 14H ; (write)
DAC2 EQU 15H ; (write)

; this needs more work to understand DMH
; 16H ADCoutputPort16H:

DAC_ctrl EQU 16H
Bit7: I/O 0: Disable ADC; 1: Enable ADC
Bit6: I/O
Bit5: I/O
Bit4: I/O
Bit3: I/O
Bit2: I/O
Bit1: I/O
Bit0: I/O

---

Operating equate definition

---

To calculate samples:

CPU clk/sample rate divisor

Hi & Lo timer reg count = FFF

FFF = divisor = value of hi & lo reg.

ex: 6mHZ clk = 166nSEC

****** start Tracker

; here is some definition change of time interrupt constant ;Tracker

SystemClock: EQU 6000000 ;Select 6000000Hz it will be the same
SystemClock: EQU 3579545 ;Select 3579545Hz while we are using that crystal

TimeA_low: EQU <(4096-(SystemClock/5859)) ;put constant
TimeA_hi: EQU >(4096-(SystemClock/5859))

TimeB_low: EQU <(4096-(SystemClock/1465))
TimeB_hi: EQU >(4096-(SystemClock/1465))

;****** end Tracker

Port_def EQU A7h ;D hi=out, D lo=inp / C hi=out, C lo=inp
; B hi=inp, B lo=out / A hi=out, A lo=out

Con_def EQU 50H ; D hi=buffer, D lo=inp pull lo
; C hi=buffer, C lo=inp pull hi
; B hi=inp hi-Z, B lo=buffer
; A hi=buffer, A lo=buffer

Intt_def EQU D0h ;sets interrupt reg = no watchdog, irq
; timer B, and EXT port C bit 1 = off

;****** run EQUs

;*********************************************************************
; Send a braking pulse to stop motor drift, and this EQU is a decimal number
; that determines how many times through the 2.9 mSec loop (how many loops)
; the brake pulse is on. If attempting to make single count jumps, the brake pulse needs to be between 26 and 30. For any jump greater than 10
; braking between 22 and 80 is acceptable. (Long jumps are not critical; but short jumps will begin to oscillate if braking is too great.)

; 60 long & 20 short work at 3.6v and no pulse width
Drift_long EQU 60 ; number times thru intt before clearing pulse
Drift_short EQU 25 ;

; set this with a number from 0 - 255 to determine timeout of all sensors
; for the sequential increments. If it times out the table pointer goes back to the start, else each trigger increments through the table.

; NOTE: this time includes the motor/speech execution time !!!
Global_time EQU 16 ; 1 = 742 mSEC ;; 255 = 1.3 seconds

; This determines how long Firby waits with no sensor activity, then
; calls the Bored_table for a random speech selection.
; Use a number between 1 & 255. Should probably not be less than 10.
; SHOULD BE > 10 SEC TO ALLOW TIME FOR TRAINING OF SENSORS
Bored_time EQU 40 ; 1 = 742 mSEC ;; 255 = 189.3 seconds

; Each sensor has a sequential random sp. which must equal 16.
; Each sensor has a different assignment.
; The tables are formatted with the first X assignments random
; and the remaining as sequential.
Seq_front EQU 8
Ran_front EQU 8
Seq_back EQU 9
Ran_back EQU 7
Seq_tilt EQU 10
Ran_tilt EQU 6
Seq_invert EQU 8
Ran_invert EQU 8
Seq_sound EQU 0
Ran_sound EQU 16
; rev furb11ja
;
; Each sensor also determines how often it is random or sequential
; at 50/50 or 60/40 etc.
; These entries are subtracted from the random number generated
; and determine the split (the larger here, the more likely sequential
; pick)

Seq_light EQU 0
Ran_light EQU 16

Seq_feed EQU 8
Ran_feed EQU 8

Seq_wake EQU 0
Ran_wake EQU 16

Seq_bored EQU 7
Ran_bored EQU 9

Seq_hunger EQU 5
Ran_hunger EQU 11

Seq_sick EQU 4
Ran_sick EQU 12

;rev furb11ja

Each sensor also determines how often it is random or sequential
; at 50/50 or 60/40 etc.
; These entries are subtracted from the random number generated
; and determine the split (the larger here, the more likely sequential
; pick)

Tilt_split EQU 80h
Invert_split EQU 80h
Front_split EQU 80h
Back_split EQU 80h
Feed_split EQU 80h
Sound_split EQU 80h
Light_split EQU 80h
Bored_split EQU 80h
Hunger_split EQU 80h
Sick_split EQU 80h

; Random_age EQU 30h ; at any age, below this number when a
; random number is picked will cause him
; to pull from the age 1 table. More Furbish.

Learn_chg EQU 31 ; amount to inc or dec training of words

Food EQU 20h ; amount to increase 'Hungry' for each feeding
Need_food EQU 80h ; below this starts complaining about hunger
Sick_reff EQU 60h ; below this starts complaining about sickness
Really_sick EQU 10h ; below this only complains about sickness
Max_sick EQU 80h ; can't go below this when really sick

Hungry_dec EQU 01 ; subtract X amount for each sensor trigger
Sick_dec EQU 01 ; subtract X amount for each sensor trigger

Nt_word EQU FEH ; turn speech word active off
Nt_last EQU FBH ; bit 2 off - last word sent to TI
Nt_term EQU F7h ;bit 3 off -terminator to speech TI
Clr_spch EQU FCH ;clears spch_activ & word_activ

---------
Motor_rev EQU FDh ;clears motor fwd bit
Motor_inactv EQU FFh ;kill motor activ bit
Motor_nseek EQU FFh ;kill motor seek bit
Motor_off EQU C0h ;turns both motor lines off (hi)
Motor_revs EQU 7FH ;bit 7 lo
Motor_fwds EQU BFh ;bit 6 lo
Nt_mot_on EQU DFh ;clears motor pulse on req
NtIRQdn EQU F7h ;clear IRQ stat
Nt_Motor_led EQU DFh ;motor opto led off
Motor_ledrst EQU 100 ;X * 2.9 millSec for shut off time

---------
Nt_Init_motor EQU FBh ;cks motor speed only on wake up
Nt_Init_Mspeed EQU F7h ;clears 2nd part of motor speed test

---------
Opto_spd_reld EQU 80 ;number of IRQ to count opto pulse speed
Speed_reff EQU 30 ;value to adjust speed to

---------
Nt_macro_actv EQU 7Fh ;clears request

---------
Not_bside EQU F7h ;clear ball side done flag
Not_binvrt EQU FFh ;clear ball invert done flag
Not_tch_bk EQU BFh ;clear touch back sense done flag
Not_tch_ft EQU DFh ;clear touch front sense done flag
Not_feed EQU FDh ;clear feed sense done flag

Lights_reld EQU 05 ;X * 742 millisec time between trigger
Light_cycle_reld EQU 02 ;sound sense reference cycle timer

---------
Nt_Slot_dn EQU FEh ;clear IR slot low detected

---------
Nt_lt_reff EQU FFh ;turns reff off
Nt_light_stat EQU FEh ;clears light bright status to dim status

;;; Bright & Dim equates have been moved to the light include file.

;;; Bright EQU 05 ;light sensor trigger > reff level
;;; Dim EQU 05 ;Light sensor trigger < reff level

---------
Qik_sna_reload EQU 01 ;
Nt_snd_reff EQU DFh ;kill sound reff level bit
Nt_do_snd EQU FFh ;clears sound stats change req
Nt_snd_stat EQU FFh ;clears Sound_stat

---------
Nt_fortune EQU FFh ;kills fortune teller mode
Nt_Rap EQU FDh ;kills Rap mode
Nt_hidaseek EQU FBh ;kills Hide & seek game mode
Nt_simon EQU 7Fh ;kills simon say game mode

---------
Nt_do_tummy EQU F7h ;clears sensor changes req
Nt_do_back EQU EFh ;clears sensor change req
Nt_do_feed EQU DFh ;clears sensor change req
Nt_do_tilt EQU BFh ;clears sensor change req
Nt_do_invert EQU 7Fh ;clears sensor change req
Nt_do_lt_brt EQU DFh ;clears sensor change req
Nt_do_lt_dim EQU F8h ; clears sensor change req
Nt_temp_gam1 EQU F9h ; clears game mode bits
Nt_half_age EQU BFh ; clears req for 2 take instead of 4
Nt_randm EQU 7Fh ; clears random/sequential status

GameT_reload EQU 24 ; 1 = 742 mSEC ; 255 = 189.3 seconds

Variable definition (Ram = $00 to $FF)

DO NOT CHANGE RAM ASSIGNMENTS (X pointer used as offset)

The next group of RAM locations can be used by any sensor routine but cannot be used to save data.

TEMP ONL

TEMP0 equ 80h
TEMP1 equ 61h
TEMP2 equ 82h
TEMP3 equ 83h
TEMP4 equ 84h
IN_DAT equ 85h

END TEMP RAM

Task_ptr EQU 86h ; what function is in process
Port_A_image EQU 87h ;
Port_B_image EQU 88h ; output port image
Port_D_image EQU 89h ; output port image

Word_lo EQU 8Ah ; speech word lo adrs
Word_hi EQU 8Bh ; hi
Saysent_lo EQU 8Ch ; saysent word pointer
Saysent_hi EQU 8Dh ;
Bank_ptr EQU 8Eh ; which bank words are in
Which_word EQU 8Fh ; which word or saysent to call
Group EQU 90h ; which saysent group table
Tk_data EQU 91h ;

Which_motor EQU 92h ; holds table number of motor position
Mgroup EQU 93h ; which motor group table
Motor_lo EQU 94h ;
Motorptr_lo EQU 95h ; table pointer to get motor position
Motorptr_hi EQU 96h ;
Which_delay EQU 97h ; how much time between motor calls
Intt_Temp EQU 98h ;
Drift_fwd EQU 99h ; time motor reverses to stop drift
Drift_rev EQU 9Ah ;
Pot_timeL EQU 9Bh ; motor uses to compare against current position

Moff_len EQU 9Ch ; holds motor power off pulse time
Mon_len EQU 9Dh ; holds motor power on pulse time
Motor_pulsel EQU 9Eh ; motor pulse timer
Slot_vote EQU 9Fh ; need majority cnt to declare a valid slot
**motor_lad_timer** EQU A0h ; how long after motion done led on for IR
**Mot_speed_cnt** EQU A1h ; motor speed test
**Mot_opto_cnt** EQU A2h ;
**Cal_switch_cnt** EQU A3h ; used to eliminate noisy reads
**Motorstoped equ** A4h ; times wheel count when stopping
**Drift_counter** EQU A5h ; decide how much braking pulse to apply

<table>
<thead>
<tr>
<th>EQU</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A6h</td>
<td>Drift_counter</td>
</tr>
<tr>
<td>A7h</td>
<td>Cycle_timer</td>
</tr>
<tr>
<td>A8h</td>
<td>Sensor_timer</td>
</tr>
<tr>
<td>A9h</td>
<td>Bored_timer</td>
</tr>
<tr>
<td>AAh</td>
<td>Invrt_count</td>
</tr>
<tr>
<td>ABh</td>
<td>Tilt_count</td>
</tr>
<tr>
<td>ACb</td>
<td>Tchofnt_count</td>
</tr>
<tr>
<td>ADh</td>
<td>Tchbck_count</td>
</tr>
<tr>
<td>AEh</td>
<td>Feed_count</td>
</tr>
<tr>
<td>AFh</td>
<td>Last_IR</td>
</tr>
<tr>
<td>B0h</td>
<td>Wait_time</td>
</tr>
<tr>
<td>B1h</td>
<td>Light_timer</td>
</tr>
<tr>
<td>B2h</td>
<td>Light_count</td>
</tr>
<tr>
<td>B3h</td>
<td>Light_reff</td>
</tr>
<tr>
<td>B4h</td>
<td>Sound_timer</td>
</tr>
<tr>
<td>B5h</td>
<td>Sound_count</td>
</tr>
<tr>
<td>B6h</td>
<td>Milisec_flag</td>
</tr>
<tr>
<td>B7h</td>
<td>Macro_Lo</td>
</tr>
<tr>
<td>B8h</td>
<td>Macro_Hi</td>
</tr>
<tr>
<td>B9h</td>
<td>Egg_cnt</td>
</tr>
</tbody>
</table>

**Last IR** EQU AFh ; last IR sample data to compare to next
**Wait_time** EQU B0h ; used in IRQ to create 2.8mSec timers
**Light_timer** EQU B1h ; Light sensor routines
**Light_count** EQU B2h ; speech/motor call is next
**Light_reff** EQU B3h ; holds previous example
**Sound_timer** EQU B4h ; time to set new ref level
**Sound_count** EQU B5h ; speech/motor call is next
**Milisec_flag** EQU B6h ; set every 742 milliseconds

**Macro_Lo** EQU B7h ; table pointer
**Macro_Hi** EQU B8h ;
**Egg_cnt** EQU B9h ; easter egg table count pointer

**Macall code rev B**

**HCEL_LO** EQU B4h ;
**HCEL_HI** EQU B5h ;
**BIT_CT** EQU B6h ;

**Ligl_shift** EQU BDh ; ( was TMA_INT ) used for threshold change

**Prev_random** EQU B8h ; prevents random number twice in a row
**Bored_count** EQU B9h ; sequential selection for bored table
**TEMP5** EQU C0h ; general use also used for wake up
**Temp_ID2** EQU C1h ; use in sensor training routines
**Temp_ID** EQU C2h ; use in sensor training routines
**Learn_temp** EQU C3h ; use in sensor training routines

**Req_macro_lo** EQU C4h ; holds last call to see if sleep or IR req
**Req_macro_hi** EQU C5h ;
**Sickr_count** EQU C6h ; sequential counter for sick speech table
**Hungr_count** EQU C7h ; sequential counter for hunger speech table
```plaintext
Motor_pulse2 EQU C8h ; motor pulse timer

; "***** DO NOT CHANGE BIT ORDER *****"

<table>
<thead>
<tr>
<th>Stat_0</th>
<th>EQU C9h</th>
<th>; System status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Want_name EQU 01H</td>
<td>; bit 0 = set Forces system to say Purby's name</td>
<td></td>
</tr>
<tr>
<td>Lt_prev_dn EQU 02H</td>
<td>; bit 0 = done flag for quick light changes</td>
<td></td>
</tr>
<tr>
<td>Init_motor EQU 04H</td>
<td>; bit 1 = on wakeup do motor speed/batt test</td>
<td></td>
</tr>
<tr>
<td>Init_Mspeed EQU 08H</td>
<td>; bit 3 = 2nd part of motor speed test</td>
<td></td>
</tr>
<tr>
<td>Train_Bk_prev EQU 10H</td>
<td>; bit 4 = set when 2 back sw hit in a row</td>
<td></td>
</tr>
<tr>
<td>Save_new_name EQU 04H</td>
<td>; bit 5 = only happens on cold boot</td>
<td></td>
</tr>
<tr>
<td>REQ_dark_sleep EQU 04H</td>
<td>; bit 6 = set - dark level sends to sleep</td>
<td></td>
</tr>
<tr>
<td>Gotosleep</td>
<td>; bit 7 = if set on wake up then dont</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stat_1</th>
<th>EQU CA9h</th>
<th>; System status more</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word_activ EQU 01H</td>
<td>; bit 0 = set during any speech</td>
<td></td>
</tr>
<tr>
<td>Say_activ EQU 02H</td>
<td>; bit 1 = when saysent is in process</td>
<td></td>
</tr>
<tr>
<td>Word_end EQU 04H</td>
<td>; bit 2 = set when sending FF word end to TI</td>
<td></td>
</tr>
<tr>
<td>Word_term EQU 08H</td>
<td>; bit 3 = set to send 3 #ffh to end speech</td>
<td></td>
</tr>
<tr>
<td>Up_light EQU 10H</td>
<td>; bit 4 = set when shift is incrmtg</td>
<td></td>
</tr>
<tr>
<td>Send_ref EQU 20H</td>
<td>; bit 5 = set for new referenc cycle</td>
<td></td>
</tr>
<tr>
<td>Half_ege EQU 40H</td>
<td>; bit 6 = set for 2 tables of ege instead of 4</td>
<td></td>
</tr>
<tr>
<td>Randm_sel EQU 80H</td>
<td>; bit 7 = decides random/sequential for tables</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stat_2</th>
<th>EQU CBH</th>
<th>; System status more</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motor_activ EQU 01H</td>
<td>; bit 0 = set = motor in motion</td>
<td></td>
</tr>
<tr>
<td>Motor_fwd EQU 02H</td>
<td>; bit 1 = set=fwd clr=rev</td>
<td></td>
</tr>
<tr>
<td>Motor_seek EQU 04H</td>
<td>; bit 2 = seeking to next position</td>
<td></td>
</tr>
<tr>
<td>Side_dn EQU 08H</td>
<td>; bit 3 = set = previously flaged</td>
<td></td>
</tr>
<tr>
<td>Binvrt_dn EQU 10H</td>
<td>; bit 4 = set prev done</td>
<td></td>
</tr>
<tr>
<td>Tchf_dn EQU 20H</td>
<td>; bit 5 = -</td>
<td></td>
</tr>
<tr>
<td>Tchbk_dn EQU 40H</td>
<td>; bit 6 = -</td>
<td></td>
</tr>
<tr>
<td>Macro_activ EQU 80H</td>
<td>; bit 7 = set when macro in process</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stat_3</th>
<th>EQU CCh</th>
<th>; System status more</th>
</tr>
</thead>
<tbody>
<tr>
<td>Light_set EQU 01H</td>
<td>; bit 0 = set= bright clr = dim</td>
<td></td>
</tr>
<tr>
<td>Feed_dn EQU 02H</td>
<td>; bit 1 = set= prev done</td>
<td></td>
</tr>
<tr>
<td>Sound_set EQU 04H</td>
<td>; bit 2 = -</td>
<td></td>
</tr>
<tr>
<td>IRQ_dn EQU 08H</td>
<td>; bit 3 = set when IRQ occurs by IRQ</td>
<td></td>
</tr>
<tr>
<td>Lt_reff EQU 10H</td>
<td>; bit 4 = set for light sense reff cycle</td>
<td></td>
</tr>
<tr>
<td>Motor_on EQU 20H</td>
<td>; bit 5 = set=motor pulse power on</td>
<td></td>
</tr>
<tr>
<td>M_forward EQU 40H</td>
<td>; bit 6 = lr = move motor forward</td>
<td></td>
</tr>
<tr>
<td>M_reverse EQU 80H</td>
<td>; bit 7 = clr = move motor reverse</td>
<td></td>
</tr>
</tbody>
</table>

; ***********************************************

: Following bit maps are reserved for easter egg / games

<table>
<thead>
<tr>
<th>Stat_4</th>
<th>EQU CDH</th>
<th>; System task request state</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do_snd EQU 01H</td>
<td>; bit 0 = set when sound &gt; prev reff level</td>
<td></td>
</tr>
<tr>
<td>Do_light_brt EQU 02H</td>
<td>; bit 1 = set when light &gt; prev reff level</td>
<td></td>
</tr>
<tr>
<td>Do_light_dim EQU 04H</td>
<td>; bit 2 = set when light &lt; prev reff level</td>
<td></td>
</tr>
<tr>
<td>Do_tummy EQU 08H</td>
<td>; bit 3 = set when front touch triggered</td>
<td></td>
</tr>
<tr>
<td>Do_back EQU 10H</td>
<td>; bit 4 = set when back touch triggered</td>
<td></td>
</tr>
</tbody>
</table>
```
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do_feed</td>
<td>20H</td>
<td>bit 5 = set when feed sensor triggered</td>
</tr>
<tr>
<td>Do_tilt</td>
<td>40H</td>
<td>bit 6 = set when tilt sensor triggered</td>
</tr>
<tr>
<td>Do_invert</td>
<td>80H</td>
<td>bit 7 = set when inverted sensor triggered</td>
</tr>
<tr>
<td>Stat_5</td>
<td>CEh</td>
<td>game status</td>
</tr>
<tr>
<td>temp_gam1</td>
<td>01H</td>
<td>bit 0 = used in game play</td>
</tr>
<tr>
<td>temp_gam2</td>
<td>02H</td>
<td>bit 0 =</td>
</tr>
<tr>
<td>temp_gam3</td>
<td>04H</td>
<td>bit 1 =</td>
</tr>
<tr>
<td>temp_gam4</td>
<td>08H</td>
<td>bit 3 =</td>
</tr>
<tr>
<td>temp_gam5</td>
<td>10H</td>
<td>bit 4 =</td>
</tr>
<tr>
<td>temp_gam6</td>
<td>20H</td>
<td>bit 5 =</td>
</tr>
<tr>
<td>temp_gam7</td>
<td>40H</td>
<td>bit 6 =</td>
</tr>
<tr>
<td>temp_gam8</td>
<td>80H</td>
<td>bit 7 =</td>
</tr>
<tr>
<td>Game_1</td>
<td>CFh</td>
<td>system game status</td>
</tr>
<tr>
<td>Fortune_mode</td>
<td>01H</td>
<td>bit 0 = set = furby in fortune teller mode</td>
</tr>
<tr>
<td>Rap_mode</td>
<td>02H</td>
<td>bit 0 = set = furby in RAP SONG mode</td>
</tr>
<tr>
<td>Hide_seek_mode</td>
<td>04H</td>
<td>bit 1 = set = furby in hide &amp; seek game mode</td>
</tr>
<tr>
<td>Simon_say_mode</td>
<td>08H</td>
<td>bit 3 = set = furby in simon says game mode</td>
</tr>
<tr>
<td>Burp_mode</td>
<td>10H</td>
<td>bit 4 = set = mode</td>
</tr>
<tr>
<td>Name_mode</td>
<td>20H</td>
<td>bit 5 =</td>
</tr>
<tr>
<td>Twinkle_mode</td>
<td>40H</td>
<td>bit 6 =</td>
</tr>
<tr>
<td>Rooster_mode</td>
<td>80H</td>
<td>bit 7 =</td>
</tr>
<tr>
<td>Qualify1</td>
<td>DOh</td>
<td>easter egg disquallified when clear</td>
</tr>
<tr>
<td>DQ_fortune</td>
<td>01h</td>
<td>bit 0 = fortune teller</td>
</tr>
<tr>
<td>DQ_rap</td>
<td>02h</td>
<td>bit 1 = rap song</td>
</tr>
<tr>
<td>DQ_hide</td>
<td>04h</td>
<td>bit 2 = hide and seek</td>
</tr>
<tr>
<td>DQ_simon</td>
<td>08h</td>
<td>bit 3 = simon says</td>
</tr>
<tr>
<td>DQ_burp</td>
<td>10h</td>
<td>bit 4 = burp attack</td>
</tr>
<tr>
<td>DQ_name</td>
<td>20h</td>
<td>bit 5 = says his name</td>
</tr>
<tr>
<td>DQ_twinkle</td>
<td>40h</td>
<td>bit 6 = sings song</td>
</tr>
<tr>
<td>DQ_rooster</td>
<td>80h</td>
<td>bit 7 = rooster loves you</td>
</tr>
</tbody>
</table>

// THIS GROUP OF RAM IS SAVED IN EEPROM

// Need to read these from EEPROM and do test for false data

// "age" uses bit 7 to extend the "age_counter" to 9 bits, and this
// is saved in EEPROM also.

// "AGE" MUST BE IN D1h BECAUSE EEPROM READ & WRITE USE THE EQU FOR START RAM.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>D1h</td>
<td>age = 0-3 (4 total)</td>
</tr>
<tr>
<td>Age_counter</td>
<td>D2h</td>
<td>inc on motor action, rolls over &amp; inc age</td>
</tr>
<tr>
<td>Name</td>
<td>D3h</td>
<td>holds 1-6 pointer to firby's name</td>
</tr>
<tr>
<td>Rvoice</td>
<td>D4h</td>
<td>which one of three voices</td>
</tr>
<tr>
<td>Pot_timeL2</td>
<td>D5h</td>
<td>counter from wheel L.R. sensor</td>
</tr>
<tr>
<td>Hungry_counter</td>
<td>D6h</td>
<td>holds hungry/full counter</td>
</tr>
<tr>
<td>Sick_counter</td>
<td>D7h</td>
<td>healthy/sick counter</td>
</tr>
<tr>
<td>Seed_1</td>
<td>D8h</td>
<td>only seed 1 &amp; seed 2 are saved</td>
</tr>
<tr>
<td>Seed_2</td>
<td>D9h</td>
<td></td>
</tr>
</tbody>
</table>

// These are used for training each sensor. There is a word number which
is 1-16 for the sensor table macro list and a ram for count which
determines how often to call the learned word.

*** DO NOT CHANGE ORDER----- RAM adrs by Xreg offset

Tilt_learned   EQU DAh ;which word trained
Tilt_lrn_cnt   EQU DBh ;count determines how often called
Feed_learned   EQU DCh ;which word trained
Feed_lrn_cnt   EQU DDh ;count determines how often called
Light_learned   EQU DEh ;which word trained
Light_lrn_cnt   EQU DFh ;count determine how often called
Dark_learned   EQU E0h ;which word trained
Dark_lrn_cnt   EQU E1h ;count determines how often called
Front_learned   EQU E2h ;which word trained
Front_lrn_cnt   EQU E3h ;count determine how often called
Sound_learned   EQU E4h ;which word trained
Sound_lrn_cnt   EQU E5h ;count determines how often called
Wake_learned   EQU E6h ;which word trained
Wake_lrn_cnt   EQU E7h ;count determines how often called
Invert_learned   EQU E8h ;which word trained
Invert_lrn_cnt   EQU E9h ;count determines how often called

; next is equates defining which ram to use for each sensor
; according to the sensor ram defined above. (compare to numbers above)

Tilt_ID        EQU 00 ;defines what offset for above_ram
definitions
Feed_ID        EQU 02
Light_ID       EQU 04
Dark_ID        EQU 06
Front_ID       EQU 08
Sound_ID       EQU 10
Wake_ID        EQU 12
Invert_ID      EQU 14
Back_ID        EQU EEh ;special value triggers learn mode

** For power on test, WE only clear ram to E9h and use EAh for a
; messenger to the warm boot routine. We always clear ram and initialize
; registers on power up, but if it is a warm boot then read EEPROM
; and setup ram locations. Location EAh is set or cleared during power
; up
; and then the stack can use it during normal run.

Warm-cold      EQU EDh
Spcl_seed1     EQU EEh
Spcl_seed2     EQU EFh
Deep_sleep     EQU F0h ;0=no deep sleep 11h is. (tilt wont wakeup)

****** Need to allow stack growth down ( EAh- FFH ) ********
Stacktop EQU FFH ; Stack Top

_ORG 00H
BLKW 300H,00H ;Fill 0000 AAA 05FFH= 00

PROGRAM STARTS HERE

ORG 0600H

RESET:

Include Wake2.asm ;asm file

;********* end Tracker

; For power on test, WE only clear ram to E9h and use EAh for a
; messenger to the warm boot routine. We always clear ram and initialize
; registers on power up, but if it is a warm boot then read EPROM
; and setup ram locations. Location EAH is set or cleared during power
; up and then the stack can use it during normal run.

; Clear RAM to 00H

LDA #00H ; data for fill
LDX #E9H ; start at ram location

RAMClear:
STA 00,X ; base 00, offset x
DEX ; next ram location
CPX #7FH ; check for end
BNE RAMClear ; branch, not finished ; fill done
Main:

InitIO:
LDA 01 ; turn DAC on
STA DAC_ctrl ; DAC control
LDA #Port_def ; set direction control
STA Ports_dir ; load reg
LDA #Con_def ; set configuration
STA Ports_con ; load reg
LDA 00 ; set for bank 0
STA Bank ; set it
LDA #00H ; disable wakeup control
STA Wake_up ;
LDA #00h ; disable sleep control
STA Sleep ; set don't care
LDA #Int_dflt ; Initialize timers, etc.
STA Interrupts ; load reg
LDA #00H ; set timer mode
STA TMA_CON ; set reg
LDA #TimeA_low ; get preset timer for interrupts
STA TMA_LSB ; load
LDA #TimeA_hi ; get hi byte for preset
STA TMA_MSB ; load it
LDA #TimeB_low ; get preset timer for interrupts
STA TMB_LSB ; load
LDA #TimeB_hi ; get hi byte for preset
STA TMB_MSB ; load it
LDA #Ch ; preset status for motors off
STA Stat_3 ;
LDA #00H ; init ports
STA Port_A ; output
LDA #33H ; init ports
STA Port_B_Image ; ram image
STA Port_B ; output
LDA #01H ; init ports
STA Port_C ; output
LDA #DOH ; init ports
STA Port_D_Image ; ram image
STA Port_D ; output
LDA #FFh ; milisec timer reload value
STA Mili_sec ; also preset IRC timer
CLI ; Enable IRQ
JSR Kick_IRQ ;wait for interrupt to restart.
JSR TI_reset ;go init TI (uses 'Cycle_timer')

Preset motor speed, assuming mid battery life, we set the pulse width
so that the motor won't be running at 6 volts and burn out. We then
predict what the pulse width should be for any voltage.

| LDA  | #Mpulse_on | ;preset motor speed |
| LDA  | #11        | ;set motor on pulse timing |
| LDA  | #05        | ;set motor off pulse timing |
| STA  | Mono_len   | |
| STA  | Moff_len   | |

; \*
; Diagnostics and calibration Routine
; \*

Include Diag7.asm ;asm file

****** Only called by diagnostic speech routines *******

; Be sure to set 'MACRO_HI' and all cells are in that 128 byte block.

Dieg_macro:
  STA Macro_Lo ;save lo byte of Macro table entry
  LDA #0b8h ;490h
  ;=x offset to adrs.400 added
to diag call
  CLC
  ADC Macro_Lo ;add in offset
  STA Macro_Lo ;update
  LDA #01 ;get hi byte adrs 400 = 190h
  STA Macro_Hi ;save hi byte of Macro table entry
  JSR Get_macro ;go start motor/speech
  JSR Notrdy ;Do / get status for speech and motor
  RTS ;yo !

; Enter with Areg holding how many 30 milisecond delay cycles

Half_delay:
  STA TEMPI ;save timer

Half_d2:
  LDA #10 ;set 1/2 sec (y = 2.9 mSec)
  STA Cycle_timer ;set it

Half_d3:
  LDA Cycle_timer ;ck if done
  BNE Half_d3 ;loop
  DEC TEMPI ;
  BNE Half_d2 ;loop
  RTS ; done
; We assume diagnostic only runs on coldboot

LDA #FFh ;initialize word training variable
STA Temp_ID ;
LDA #FFh ;
STA Hungry_counter ; preset furby's health
STA Sick_counter

; We sit here and wait for tilt to go away, and just keep incrementing ; counter until it does. This becomes the new random generator seed.

Init_md:
INC TEMPl ; random counter
LDA Port_D ; get switches
AND #03 ; check tilt & invert sw
BNE Init_md ; loop till gone
LDA TEMPl ; get new seed
STA Spcl_seed1 ; stuff it
STA Seed_1 ; also load for cold boot

; Use feed sw to generate a better random number

JSR Get_feed ; go test sensor
LDA Stat_4 ; get system
AND #DFh ; check sw
BNE Feed_md ; if feed sw then cold boot
JMP End_coldinit ; else do warm boot

Feed_md:
INC TEMPl ; random counter
LDA Stat_4 ; get system
AND #DFh ; clear any prev feed sw senses
STA Stat_4 ; update
JSR Get_feed ; go test sensor
LDA Stat_4 ; get system
AND #DFh ; check sw
BNE Feed_md ; wait for feed to go away
LDA TEMPl ; get new seed
STA Spcl_seed1 ; stuff it
STA Seed_1 ; also load for cold boot

; If this is a cold boot, reset command then clear EEPROM and ; chose a new name and voice.

Do_cold_boot:
LDA #00
STA Warm_cold ; flag cold boot
LDA Stat_0 ;system
ORA #Sey_new_name ;make system say new name
STA Stat_0

****** NOTE ******

; VOICE AND NAME SELECTION MUST HAPPEN BEFORE EEPROM WRITE OR
; THEY WILL ALWAYS COME UP 00 because ram just got cleared!!!!!!

; Random voice selection here

LDA #80h ;get random/sequential split
STA IN_DAT ;save for random routine

LEX #00 ;make sure only gives random
LDA #10h ;get number of random selections
JSR Ran_seq ;go get random selection

TAX
LDA Voice_table,X ;get new voice
STA Rvoice ;set new voice pitch

; On power up or reset, Furby must go select a new name ... ahw how cute.

JSR Random ;
AND #1Fh ;get 32 possible
STA Name ;set new name pointer
JSR Do_EE_write ;write the EEPROM

End_coldinit:

; FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
; 'Special initialization prior to normal run mode
; Jump to Warm_boot when portD wakes us up
; FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

Warm_boot: ;noinal :start when Port_D wakes us up.

JSR $EEH:M_READ ;read data to ram

Eeprom_read_byp:

; FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
; If light osc fails, or too dark and that sends us to sleep, we
; set 'Dark_sleep_prev' and save it in EEPROM in 'Seed_2'.
; when the sleep routine executes, {00 01 based on this bit}
; When we wake up we recover this bit and it becomes the previous done
; flag back in 'Stat_0', so that if the osc is
; still dark or failed, Furby won't go back to sleep.
LDA Seed_2 ; from EEPROM
BEQ No_prevsleep ; jump if none
LDA Stat_0 ; system
ORA #Dark_sleep_prev ; prev done
STA Stat_0 ; update

No_prevsleep:

; LDA Spcl_seedl ; recover start up random number
; STA Seed_1 ; set generator

;******************************************

; Pot_timeL2 is saved in RAM through sleep mode and then reloaded.
; Pot_timeL which is the working register for the motor position.
; This allows startup routines to clear RAM without forgetting the
; last motor position.
LDA Pot_timeL2 ; get current count
STA Pot_timeL ; save in motor routine counter

;******************************************

; Get <age> and make sure it is not greater than 3 (age4)
LDA Age ; get current age
AND #83h ; preserve bit 7 which is 9th age counter bit
; ; ; ; ;
STA Age ; set system

;******************************************

LDA #Bored_reld ; reset timer
STA Bored_time ;

LDA #01 ; set timer
STA Last_IR ; timer stops IR from hearing own IR xmit

JSR Get_light ; go get light level sample
LDA TEMPl ; get new count
STA Light_reff ; update system

;******************************************

LDA Warm_cold ; decide if warm or cold boot
CMP #11h ; check for warm boot
BEQ No_zero ; jump if is
LDA $00 ; point to macro 0 (SEND TO SLEEP POSITION)
STA Macro_Lo
STA Macro_Hi
JSR Get_macro ; go start motor/speech
JSR Notrdy ; Do / get status for speech and motor

; Currently uses 4 tables, one for each age.
LDA Stat_0
ORA *Init_motor ; flag motor to do speed test
ORA *Init_Mspeed ; 2nd part of test
STA Stat_0 ; update

; Do wake up routine:
LDA #Global_time ; reset timer to trigger sensor learning
STA Sensor_timer

LDA #80h ; get random/sequential split
STA IN_DAT ; save for random routine

LDX $00h ; make sure only gives random
LDA #10h ; get number of random selections

JSR Ram_seq ; go get random selection
LDA TEMPO ; get decision

STA IN_DAT ; save decision
LDA #Wake_ID ; which ram location for learned word count

JSR Start_learn ; go record training info
LDA IN_DAT ; get back word to speak

JSR Decid_age ; do age calculation for table entry
LDX TEMP0 ; age offset
LDA Wakeup_S1,X ; get new sound/word
STA Macro_Lo ; save lo byte of Macro table entry

LEX
LDA Wakeup_S1,X ; get new sound/word
STA Macro_Hi ; save hi byte of Macro table entry
JMP Start_macro ; go start speech
Idle:

; Idle routine is the time slice task master (TSTM) ugh!
; We must call each routine and interleave with a call to speech
; to insure we never miss a TI request for data.

JSR Notready ; Do / get status for speech and motor

; This bit is set when light sensor is darker than 'Dark_sleep'

LDA Stat_0 ; system
AND #REQ_dark_sleep ; ck for req
BEQ No_dark_req ; jump if not

LDA Stat_0 ; system
AND #BFh ; kill req
STA Stat_0 ; update

LDA #A6h ; sleep macro
STA Macro_Lo
LDA #00h ; sleep macro
STA Macro_Hi

JMP Start_macro ; go say it

No_dark_req:

; When any sensor or timer calls the "start_macro" routine, the
; Macro_Lo & Macro_Hi are saved. Everyone jumps back to Idle and when
; speech/motor routines are finished, this routine will look at the
; macros that were used and execute another function if a match is
; found.

; Checks for his name first, then any IR to send, and finally, the sleep
; commands. The temp macro buffers are cleared before

Spcl_Name1:
LDX #00 ; offset

Spcl_Name2:
LDA Ck_Name_table,X ; ck lo byts
CMP #FFh ; ck for end of table (not 255 cant execute)
BEQ Spcl_IR1 ; done if is
CMP Req_macro_lo ; ck against last speech request
BNE Not_Name2 ; jump if not
INX ; to hi byte
LDA Ck_Name_table,X ; ck hi byte
CMP Req_macro_hi ; ck against last speech request
BNE Not_Name3 ; jump if not
JMP Say_Name ; speak it
Not_Name2: INX
Not_Name3: INX
JMP Spcl_Name2 ; loop till done

Say_Name:
LDA Stat_0 ; kill req for startup new name
AND #DFh ; update
STA Stat_0
LDA Name ; current setting for table offset
CLC
ROL A ; 2's comp
TAX
LDA Name_table,X ; get lo byte
STA Macro_lo ; save lo byte of Macro table entry
INX
LDA Name_table,X ; get hi byte
STA Macro_hi ; save hi byte of Macro table entry
JSR Get_macro ; go start motor/speech
JSR Notrdy ; Do / get status for speech and motor

Spcl_IR1:
LDX #00 ; offset
Spcl_IR2:
LDA IRxmit_table,X ; ck lo byte
CMP #FFh ; ck for end of table (note 255 cant execute)
BEQ Spcl_IR_dn ; done if is
CMP Req_macro_lo ; ck against last speech request
BNE Not_IRxmit2 ; jump if not
INX ; to hi byte
LDA IRxmit_table,X ; ck hi byte
CMP Req_macro_hi ; ck against last speech request
BNE Not_IRxmit3 ; jump if not
INX ; point to IR table
LDA IRxmit_table,X ;
STA TEMP2 ; xmit temp r.m.
LDA #EDh ; TI command for IR xmit
STA TEMP1 ;
JSR Xmit_TI ; go send it
LDA #Bored_reld ; reset bored timer
STA Bored_timer ;
LDA #03 ; set timer
STA Last_IR ; timer stops IR from hearing its own IR xmit

JMP Spcl_IR_dn ; done - ola ....
Not_IRxmit2: INX
Not_IRxmit3: INX
JMP Spcl_IR2 ; loop till done
; SPECIAL MACRO
LDX #00 ;offset

; SPECIAL SLEEP
LDA Sleepy_table,X ;ck lo byte
CMP #FFh ;ck for end of table (note 255 can't execute)
BEQ Ck_macro_dn ;done if is
CMP Req_macro_lo ;ck against last speech request
BNE Not_sleepy2 ;jump if not
INX ;to hi byte
LDA Sleepy_table,X ;ck hi byte
CMP Req_macro_hi ;ck against last speech request
BNE Not_sleepy3 ;jump if not
LDA #00 ;clear macro pointers for wake up
STA Req_macro_lo
STA Req_macro_hi

; SPECIAL SLEEP 2
Before going to sleep send sleep command to all others.

LDA #15 ;
STA TEMP2 ;out temp ram
LDA #$FDh ;TI command for IR xmit
STA TEMP1 ;
JSR Xmit_TI ;go send it

;need to wait >600 milisec before going to sleep because we aren't using
;busy flags from TI and need to make sure it is done transmitting the
;I.R. code, the sleep routine kills the TI and it would never send the
;command.

LDA #25 ;how many 30 milisec cycles to call
JSR Half_delay ;do 30 milisec delay cycles

;end mod
JMP GoToSleep ;night-night

; SPECIAL SLEEP 3
INX

; SPECIAL SLEEP 4
INX
JMP Spcl_sleepl ;loop til done

; SPECIAL MACRO DN
LDA #00 ;clear macro pointers for wake up
STA Req_macro_lo
STA Req_macro_hi
JMP Test_new_name ;on to task master

; ; ; ; ; ; ; SLEEP TABLE & IR TABLE ..... MOVE TO INCLUDE FILE LATER

Sleepy_table:
DW 91 ;hangout
DW 166 ;wake up
DW 167 ;wake up
DW 168 ;wake up
DW 169 ;wake up
IRxmit_table:

<table>
<thead>
<tr>
<th>DB</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>DW 258</td>
<td>Back dw</td>
</tr>
<tr>
<td>DW 259</td>
<td>Back dw</td>
</tr>
<tr>
<td>DW 260</td>
<td>Back dw</td>
</tr>
<tr>
<td>DW 403</td>
<td>;IR</td>
</tr>
<tr>
<td>DW 413</td>
<td>;IR</td>
</tr>
<tr>
<td>DW 429</td>
<td>;IR</td>
</tr>
<tr>
<td>DB FFh</td>
<td>;FF FF is table terminator</td>
</tr>
</tbody>
</table>

;trigger macro
DB 00 ;which IR command to call (0 - 0f)
DW 13 ;trigger macro
DB 00 ;which IR command to call (0 - 0f)
DW 17 ;trigger macro
DB 00 ;which IR command to call (0 - 0f)
DW 19 ;trigger macro
DB 00 ;which IR command to call (0 - 0f)
DW 26 ;trigger macro
DB 00 ;which IR command to call (0 - 0f)
DW 29 ;trigger macro
DB 00 ;which IR command to call (0 - 0f)
DW 33 ;trigger macro
DB 00 ;which IR command to call (0 - 0f)
DW 34 ;trigger macro
D2 90 ;which IR command to call (0 - 0f)
DW 44 ;trigger macro
DB 00 ;which IR command to call (0 - 0f)
DW 45 ;trigger macro
DB 00 ;which IR command to call (0 - 0f)
DW 48 ;trigger macro
D2 00 ;which IR command to call (0 - 0f)
DW 50 ;trigger macro
DB 00 ;which IR command to call (0 - 0f)
DW 55 ;trigger macro
DB 00 ;which IR command to call (0 - 0f)
DW 60 ;trigger macro
DB 00 ;which IR command to call (0 - 0f)
DW 149 ;from rooster wake up
DB 00 

;trigger macro
DB 01 ;which IR command to call (0 - 0f)
DW 363 ;trigger macro
DB 01 ;which IR command to call (0 - 0f)
DW 393 ;trigger macro
DB 01 ;which IR command to call (0 - 0f)
DW 248 ;trigger macro
DB 02 ;which IR command to call (0 - 0f)
DW 313 ;trigger macro
DB 02 ;which IR command to call (0 - 0f)
DW 86 ;trigger macro
DB 03 ;which IR command to call (0 - 0f)
DW 93 ;trigger macro
DB 03 ;which IR command to call (0 - 0f)
DW 339 ;trigger macro
DB 09 ;which IR command to call (0 - 0f)
DW 396 ;trigger macro
DB 09 ;which IR command to call (0 - 0f)
DW 409 ;trigger macro
DB 09 ;which IR command to call (0 - 0f)
DW 399 ;which IR command to call (0 - 0f)
DW 409 ;trigger macro
DB 09 ;which IR command to call (0 - 0f)
DW 399 ;trigger macro
DB 10 ;which IR command to call (0 - 0f)
DW 407 ;trigger macro
DB 10 ;which IR command to call (0 - 0f)
DW 408 ;trigger macro
DB 10 ;which IR command to call (0 - 0f)
DW 272 ;trigger macro
DB 11 ;which IR command to call (0 - 0f)
DW 273 ;trigger macro
DB 11 ;which IR command to call (0 - 0f)
DW 274 ;trigger macro
DB 11 ;which IR command to call (0 - 0f)
DW 275 ;trigger macro
DB 11 ;which IR command to call (0 - 0f)
DW 336 ;trigger macro
DB 12 ;which IR command to call (0 - 0f)
DW 342 ;trigger macro
DB 12 ;which IR command to call (0 - 0f)
DW 401 ;trigger macro
DB 12 ;which IR command to call (0 - 0f)
DW 92 ;trigger macro
DB 13 ;which IR command to call (0 - 0f)
DW 411 ;trigger macro
DB 13 ;which IR command to call (0 - 0f)
DW 419 ;trigger macro
DB 13 ;which IR command to call (0 - 0f)
DW 427 ;trigger macro
DB 13 ;which IR command to call (0 - 0f)
DW 291 ;trigger macro
DB 14 ;which IR command to call (0 - 0f)
DW 402 ;trigger macro
DB 14 ;which IR command to call (0 - 0f)
DW 412 ;trigger macro
DB 14 ;which IR command to call (0 - 0f)
DW 428 ;trigger macro
DB 14 ;which IR command to call (0 - 0f)
DW 256 ;trigger macro
DB 15 ;which IR command to call (0 - 0f)
DW 257 ;trigger macro
DB 15 ;which IR command to call (0 - 0f)
DW 420 ;trigger macro
DB 15 ; which IR command to call (0 - 0f)

; mod F-rels2 ; send sleep if recv sleep on IR

DW 403 ; trigger macro
DB 15 ; which IR command to call (0 - 0f)
DW 413 ; trigger macro
DB 15 ; which IR command to call (0 - 0f)

; end mod

DB FFh, FFh ; FF FF is table terminator

; CK_Name_table:

DW 97
DW 248
DW 393
DW 414
DW 149
DW 305
DW 404
DW 421

DB FFh, FFh ; FF FF is table terminator

;**********Say Name****************

; Say Name:

Test_new_name:

LDA Stat_0 ; system
AND #Say_new_name ; take system say new name
BEQ Nosayname ; bypass if clear
LDA Stat_0
AND #DFh ; kill req for startup new name
STA Stat_0 ; update
LDA Name ; current setting for table offset
CLC
ROL A ; 2's comp
TAX
LDA Name_table, X ; get lo byte
STA Macro_Lo ; save lo byte of Macro table entry
INC
LDA Name_table, X ; get hi byte
STA Macro_Hi ; save hi byte of Macro table entry
JSR Get_macro ; go start motor/speech
JSR Notrdy ; Do / get status for speech and motor

Nosayname:

;**********below routines run at 742 mSec loops**********

; Timer B sete 'Millisec_flag' each 742 milliseconds

A-32
Updt_timer:
LDA Millsec_flag ; if > 0 then 742 milli seconds have passed
BEQ TimerL_dn ; bypass if 0
LDA #00 ; clear it
STA Millsec_flag ; reset
LDA Sensor_timer ; get current timer * 742mSec sec
BEQ TimerL1 ; do nothing if 0
DEC Sensor_timer ;-1
TimerL1:
LDA Light_timer ; get current timer * 742mSec sec
BEQ TimerL2 ; do nothing if 0
DEC Light_timer ;-1
TimerL2:
LDA Sound_timer ; get current timer * 742mSec sec
BEQ TimerL3 ; do nothing if 0
DEC Sound_timer ;-1
TimerL3:
LDA Bored_timer ; get current timer * 742mSec
BEQ TimerL4 ; do nothing if 0
DEC Bored_timer ;-1
TimerL4:
LDA Last_IR ; get current timer * 742mSec
BEQ TimerL5 ; do nothing if 0
DEC Last_IR ;-1
TimerL5:
LDA 4 ; get it
CLC
SBC #08 ; check if off end
BCC Ck_tsk_A ; jump if < 9
LDA #01 ; reset pointer
STA Task_ptr ;

Ck_tsk_A:

; If too sick then no game play...

CLC
LDA Sick_counter ; how sick is he
SEC #Really_sick ;
BCC Ck_task_egg ; do egg if not
JMP Ck_bored ; bypass if too sick

; Scan all game mode pointers to determine if any are active.
; Continue to execute the first active game found, and that game always
; allows the task list to be scanned for sensor input. If no games are
; active, then check task 0 to determine if the correct sensor sequence
; is occurring which will initiate the next game.

Ck_task_egg:

LDA Game_1 ; get game active bits
ROR A ; move bit 0 to carry
BCC Ck_g2 ; check next if not active
JMP Game_fortune ;jump if active

Ck_g2:
ROR A ;bit 1
BCC Ck_g3 ;check next if not activ
JMP Game_Rap ;jump if active

Ck_g3:
ROR A ;bit 2
BCC Ck_g4 ;check next if not activ
JMP Game_Rap ;jump if activ

Ck_g4:
ROR A ;bit 3
BCC Ck_g5 ;check next if not activ
JMP Game_hideeek ;jump if activ

Ck_g5:
ROR A ;bit 4
BCC Ck_g6 ;check next if not activ
JMP Game_Burp ;jump if activ

Ck_g6:
ROR A ;bit 5
BCC Ck_g7 ;check next if not activ
JMP Game_simon ;jump if activ

Ck_g7:
ROR A ;bit 6
BCC Ck_g8 ;check next if not activ
JMP Game_name ;jump if activ

Ck_g8:
ROR A ;bit 7
BCC Ck_g9 ;check next if not activ
JMP Game_rooster ;jump if activ

Ck_g9:

; none active
;
;******************************

; Task 0: scans all active requests from sensors looking for a trigger.
; If any are set then scan through the game select tables for each game
; looking for a match, and increment the counter each time a successive
; match is found. If one is not in sequence, then that counter is reset
; to zero. Since all counters are independent, then the first one to
; completion
; wins and all other are zeroed.
;
; All sensor triggers are in one status byte so we can create a number
; based on who has been triggered (we ignore the I.R. sensor).
; The following bits are in Stat_4 and are set when they are triggered
; by the individual sensor routines:
;
; 00 = none
; 01 = Loud sound
; 02 = Light change brighter
; 04 = Light change darker
; 06 = Front tummy switch
; 10 = Back switch
; 20 = Feed switch
; 40 = Tilt switch
We assign a single bit per game or egg scenario. Each time a sensor is triggered, we increment the counter and test all eggs for a match. If a particular sensor doesn't match, then set its disqualified bit and move on. If at any time all bits are set, then clear counter to zero and start over. When a table gets an FF then that egg is executed.

Each time a sensor is triggered, the system timer is reset. This timer, called 'Sensor_timer' is reset with 'Global_time' equate. This timer is also used for the random sequential selection of sensor responses. If this timer goes to zero before an egg is complete, i.e., Furby has not been played with, then clear all disqualified bits and counters.

Currently there are 24 possible eggs. (3 bytes)

; Qualify 1:
; DQ_fortune EQU 01 ;bit 0 = fortune teller
; DQ_rap EQU 02 ;bit 1 = rap song
; DQ_hide EQU 04 ;bit 2 = hide and seek
; DQ_simon EQU 08 ;bit 3 = simon says
; DQ_burp EQU 10 ;bit 4 = burp attack
; DQ_name EQU 20 ;bit 5 = say name
; DQ_twinkle EQU 40 ;bit 6 = sing song
; DQ_rooster EQU 80 ;bit 7 = rooster-love you

; Qualify 2: ;removed due to lack of RAM
; bit 0 =
; bit 1 =
; bit 2 =
; bit 3 =
; bit 4 =
; bit 5 =
; bit 6 =
; bit 7 =

; Test trigere here

c_k_game:
; LDA Sensor_timer ;ck if no action for a while
; LDA Bored_timer ;ck if no action for a while
; BNE Ck_gameactiv ;jump if system active
; JSR Clear_games ;go reset all other triggers and game pointers

c_k_gameactiv:
; LDA Qualify1 ;test if all are disqualified
; CMP #FFh ;compare active bits only
; BNE Ck_anysens ;jump if some or all still active
; LDA Qualify2 ;test if all are disqualified
; CMP #00h ;compare active bits only
; BNE Ck_anysens ;jump if some or all still active
; JSR Clear_games ;go reset all other triggers and game pointers

c_k_anysens:
; LDA Stat_4 ;ck if any sensor is triggered
; BNE Ck_game ;go ck games if any set
; JMP Ck_bored ;bypass if none
Ck_gam1: ; fortune teller
    LDX  Egg_cnt   ; get current count
    LDA  Qualify1  ; update game qualification
    AND  #DQ_fortune ; check if dis-qualified bit
    BNE  Ck_gam2   ; bail out if is
    LDA  Fortune_table,X ; get current data
    AND  Stat_4   ; compare against sensor trigger
    BNE  Ck_gam1a ; if set then good compare
    LDA  Qualify1 ; update game qualification
    ORA  #DQ_fortune ; set dis-qualified bit
    STA  Qualify1  ; update system
    JMP  Ck_gam2  ; check next egg

Ck_gam1a:
    LDA  Fortune_table+1,X ; get current +1 to see if end of egg
    CMP  #FFh   ; test if end of table and start of game
    BNE  Ck_gam2   ; jump if not at end
    JSR  Clear_games ; go reset all other triggers and game pointers
    LDA  Game_1   ; get system
    ORA  #Fortune_mode ; start game mode
    STA  Game_1  ; update
    JMP  Idle   ; done

Ck_gam2: ; Rap mode
    LDA  Qualify1 ; update game qualification
    AND  #DQ_rap ; check if dis-qualified bit
    BNE  Ck_gam3   ; bail out if is
    LDA  Rap_table,X ; get current data
    AND  Stat_4   ; compare against sensor trigger
    BNE  Ck_gam2a ; if set then good compare
    LDA  Qualify1 ; update game qualification
    ORA  #DQ_rap ; set dis-qualified bit
    STA  Qualify1  ; update system
    JMP  Ck_gam3 ; check next egg

Ck_gam2a:
    LDA  Rap_table+1,X ; get current data +1 to see if end of egg
    CMP  #FFh   ; test if end of table and start of game
    BNE  Ck_gam3 ; jump if not at end
    JSR  Clear_games ; go reset all other triggers and game pointers
    LDA  Game_1 ; get system
    ORA  #Rap_mode ; start game mode
    STA  Game_1 ; update
    JMP  Idle ; done

Ck_gam3: ; Hide and seek
    LDA  Qualify1 ; update game qualification
    AND  #DQ_hide ; check if dis-qualified bit
    BNE  Ck_gam4 ; bail out if is
    LDA  Hseek_table,X ; get current data
    AND  Stat_4 ; compare against sensor trigger
    BNE  Ck_gam3a ; if set then good compare
    LDA  Qualify1 ; update game qualification
    ORA  #DQ_hide ; set dis-qualified bit
    STA  Qualify1 ; update system
    JMP  Ck_gam4 ; check next egg

Ck_gam3a:
    LDA  Hseek_table+1,X ; get current data +1 to see if end of egg
    CMP  #FFh ; test if end of table and start of game
    BNE  Ck_gam4 ; jump if not at end
    JSR  Clear_games ; go reset all other triggers and game pointers
LDA Game_l ;get system
ORA Hideseek_mode ;start game mode
STA Game_l ;update
JMP Idle ;done

C. gam4: ; Simon says
LDA Qualify1 ;update game qualification
AND #DQ_simon ;check if dis-qualified bit
BNE CK_gam5 ; bail out if is
LDA Simon_table,X ; get current data
AND Stat_4 ; compare against sensor trigger
BNE CK_gam4a ; if set then good compare
LDA Qualify1 ; update game qualification
ORA #DQ_simon ; set dis-qualified bit
STA Qualify1 ; update system
JMP CK_gam5 ; check next egg

CK_gam4a:
LDA Simon_table+1,X ; get current data +1 to see if end of egg
CMP #FFh ; test if end of table and start of game
BNE CK_gam5 ; jump if not at end
JSR Clear_games ; go reset all other triggers and game pointers
LDA Game_l ; get system
ORA #SimonSay_mode ; start game mode
STA Game_l ; update
LDA #00 ; clear all pointers
STA Stat_5 ; system
JMP Idle ; done

CK_gam5: ; Burp attack
LDA Qualify1 ; update game qualification
AND #DQ_burp ; check if dis-qualified bit
BNE CK_gam6 ; bail out if is
LDA Burp_table,X ; get current data
AND Stat_4 ; compare against sensor trigger
BNE CK_gam5a ; if set then good compare
LDA Qualify1 ; update game qualification
ORA #DQ_burp ; set dis-qualified bit
STA Qualify1 ; update system
JMP CK_gam6 ; check next egg

CK_gam5a:
LDA Burp_table+1,X ; get current data +1 to see if end of egg
CMP #FFh ; test if end of table and start of game
BNE CK_gam6 ; jump if not at end
JSR Clear_games ; go reset all other triggers and game pointers
LDA Game_l ; get system
ORA #Burp_mode ; start game mode
STA Game_l ; update
LDA #00 ; clear all pointers
STA Stat_5 ; system
JMP Idle ; done

CK_gam6: ; say name
LDA Qualify1 ; update game qualification
AND #DQ_name ; check if dis-qualified bit
BNE CK_gam7 ; bail out if is
LDA Name_egg,X ; get current data
AND Stat_4 ; compare against sensor trigger
BNE CK_gam6a ; if set then good compare
LDA Qualify1 ; update game qualification
ORA #DQ_name ; set dis-qualified bit
STA Qualifyl ;update system
JMP Ck_gam7 ;check next egg

Ck_gam6a:
LDA Name_egg+1,X ;get current data +1 to see if end of egg
CMP »FFh ;test if end of table and start of game
BNE Ck_gam7 ;jump if not at end
JSR Clear_games ;go reset all other triggers and game pointers
LDA Game_1 ;get system
ORA #Name_mode ;start game mode
STA Game_1 ;update
LDA #00 ;clear all pointers
STA Stat_5 ;system
JMP Idle ;done

Ck_gam7: ; twinkle song
LDA Qualifyl ;update game qualification
AND #DQ_twinkle ;check if dis-qualified bit
BNE Ck_gam8 ;bail out if is
LDA Twinkle_egg,X ;get current data
AND Stat_4 ;compare against sensor trigger
BNE Ck_gam7a ;if set then good compare
LDA Qualifyl ;update game qualification
ORA #DQ_twinkle ;set dis-qualified bit
STA Qualifyl ;update system
JMP Ck_gam8 ;check next egg

Ck_gam7a:
LDA Twinkle_egg+1,X ;get current data +1 to see if end of egg
CMP »FFh ;test if end of table and start of game
BNE Ck_gam8 ;jump if not at end
JSR Clear_games ;go reset all other triggers and game pointers
LDA Game_1 ;get system
ORA #Twinkle_mode ;start game mode
STA Game_1 ;update
LDA #00 ;clear all pointers
STA Stat_5 ;system
JMP Idle ;done

Ck_gam8: ; rooster loves you
LDA Qualifyl ;update game qualification
AND #DQ_rooster ;check if dis-qualified bit
BNE Ck_gam9 ;bail out if is
LDA Rooster_egg,X ;get current data
AND Stat_4 ;compare against sensor trigger
BNE Ck_gam8a ;if set then good compare
LDA Qualifyl ;update game qualification
ORA #DQ_rooster ;set dis-qualified bit
STA Qualifyl ;update system
JMP Ck_gam9 ;check next egg

Ck_gam8a:
LDA Rooster_egg+1,X ;get current data +1 to see if end of egg
CMP »FFh ;test if end of table and start of game
BNE Ck_gam9 ;jump if not at end
JSR Clear_games ;go reset all other triggers and game pointers
LDA Game_1 ;get system
ORA #Rooster_mode ;start game mode
STA Game_1 ;update
LDA #00 ;clear all pointers
STA Stat_5 ;system
JMP Idle ;done
Ck_gam9:

Ck_gameend:

INC Egg_cnt ; ince on any sensor trigger
LDA Egg_cnt ; get
CLC
SBC #10 ; limit max to 10 for error checking
BCC Cge2 ; continue if less
JSR Clear_games ; reset all

Cge2:

LDA #00 ; clear all sensor triggers this pass
STA Stat_4 ; ready for next pass of sensor triggers
JMP Ck_bored ; done with easter egg test

Clear_allgam:

LDA #00 ; clear all game enabled bits
STA Game_1
STA Game_2

Clear_games:

LDA #00 ; clear counter
STA Egg_cnt
STA Stat_4 ; clear game status
STA Stat_5 ; clear game status
STA Qualify1 ; clear all dis-qualify bits
STA Qualify2 ; clear all dis-qualify bits
RTS ; done

; 00 = none
; 01 = Loud sound
; 02 = Light change brighter
; 04 = Light change darker
; 08 = Front tummy switch
; 10 = Back switch
; 20 = Feed switch
; 40 = Tilt switch
; 80 = Invert switch

; These look up tables provide the sequence of sensor triggers required
; to enter that specific game mode. (FFh is always the last byte)

Fortune_table:

DB 04h, 04h, 10h, FFh ; light, light, back

Rap_table:

DB 01h, 01h, 01h, 01h, FFh ; snd, snd, snd, snd

Hseek_table:

DB 04h, 04h, 04h, 08h, FFh ; light, light, light, frnt

Simon_table:

DB 08h, 10h, 01h, 04h, FFh ; frnt, back, snd, light

Burp_table:
DB 20h, 20h, 20h, 10h, FFh ; feed, feed, feed, back
Name_egg:  
DB 38h, 08h, 08h, 10h, FFh ; frnt, frnt, frnt, back
Twinkle_egg:  
DB 01h, 01h, 01h, 10h, FFh ; snd, snd, snd, back
Rooster_egg:  
DB 04h, 04h, 04h, 10h, FFh ; light, light, light, back

; Normal task scan of sensors and timers.

; Currently uses 4 tables, one for each age.

; CK_bored:
LDA Bored_count ; ck if bored ... =0
BNE CK_tsk1 ; jump if not bored

LDX #Bored_split ; get random/sequential split
STA IN_DAT ; save for random routine

LDA #Seq_bored ; get number of sequential selections
LDA #Ran_bored ; get number of randoms
JSR Ran_seq ; go decide random/sequential
B.S Bored_ran ; Random mode when carry SET

LDX Bored_count ; save current
INC Bored_count ; if not then next table entry
LDA Bored_count ; get
CLC
SEC #Seq_bored-1 ; ck if > assignment
BCC Bored_side ; jump if <
LDA #00 ; reset to 1st entry of sequential
STA Bored_count ;
Bored_side:
STA ; currentCount

Bored_ran:
JSR Decid_age ; do age calculation for table entry
LDX TEMPO ; age offset
LDA Bored_S1,X ; get new sound/word
STA Macro_Lo ; save lo byte of Macro table entry

INX ;
LDA Bored_S1,X ; get new sound/word
STA Macro_Hi ; save hi byte of Macro table entry
JMP Start_macro ; go set group/table pointer for motor & speech

; CK_tsk1:
LDA Task_ptr ;
CMP #01 ; decide which
BNE CK_tsk4 ; jump if not
JMP CK_tilt ; CK ball switch side sense

; CK_tsk4:
CMP #02 ; decide which
BNE CK_tsk5 ; jump if not
This rtn tests for motor and speech activity and only services them to allow each request to finish, and then return to task routine. As long as motor is active, we continually reload the motor led timer to keep the optical counter alive and when all activity is complete, the IRQ will turn led off when timer goes to 00.

Notrdy:

JSR Task_1; go do speech
JSR Task_2; go do motor

LDA Stat_1; get system
AND #Word_activ; Test for speech word active
BNE Notrdy2; jump if not done
LDA Stat_2; update
AND #Say_activ; ck for saysent active
BNE Notrdy2

LDA Stat_2; get system
AND #Motor_esek; ck motor request
BNE Notrdy2; jump if set
LDA Stat_2; get system
AND #Motor_acvt; ck motor in motion
BNE Notrdy2

LDA Drift_fwd; motor drift counter 0 when done
BNE Notrdy2
LDA Drift_rev ;
BNE Notrdy2 ;
LDA Stat_2 ;system
AND #Macro_actv ;ck for flag request
BEQ Notrdy_dn ;bail if none
JSR Ck_MACRO ;decide if more chaining in process
JMP Notrdy2 ;continue
Notrdy_dn:
RTS ;only leave when everyone done
Notrdy2:
LDA #Motor_led_rst ;get led timer reload
STA Motor_led_timer ;how long the motor stays on
JMP Notrdy ;loop

Task_1:
LDA Stat_1 ;get system
AND #Word_activ ;Test for spch word active
BNE W_activ ;jump if not done
; More_spch:
LDA Stat_1 ;update
AND #Say_activ ;ck for say sent active
BEQ EndTask_1 ;nothing going on, ck next task
JSR Do_nextsent ;continue on with say sent
JMP EndTask_1 ;Next task
W_activ:
LDA Port_B ;get TI req/busy line
AND #TI_RTS ;ck byte
L.E EndTask_1 ;if no speech then ck motor
JSR Do_spch ;go send next byte to TI
EndTask_1:
RTS

Task_2:

; ----------- Motor Routines -----------
; get next motor data

Ck_motor:
LDA Stat_2 ;get system
AND #Motor_actv ;ck motor in motion
BEQ Ck_mot2 ;done
JMP Do_motor ;not done so check position
Ck_mot2:
LDA Stat_2 ;get system
AND #Motor_next ;ck motor request
BEQ NEXT_out ;jump if none

Next_motor:
; LDA Drift_fwd ;motor drift counter 0 when done
Sat a timer and check counter 'motorstoped' (incremented with wheel count) to see if it changed. When it stops changing then the motor has stopped.

LDA motorstoped ; check for 0
BNE NMM_out ; wait till 0
LDA TEMPl ; get last motor count
CMP Pot_timeL ; check if changed
BEO Motor_done ; jump if same (motor finally stopped)
LDA Pot_timeL ; get current
STA TEMPl ;
LDA $15 ; reset timer (8)
STA motorstoped ;
JMP NMM_out ; wait another cycle

Motor_done:
LDA Cycle_timer ; get step timer
BNE NMM_out ; wait till 0
STA Drift_counter ; use as a temp register
JSR Motor_data ; get data
LDA $00
STA TEMPl ; reset
LDA Motor_lo ; get data (use for 1 byte table DB). CMP $FFh ; is it table end (don't inc off end)
BNE Motor_pause ; more
LDA Stat_2 ; get system
AND #Motor_ntseek ; clear seek flag
STA Stat_2 ; update system

Motor_pause:
LDA Motor_lo ; check for pause request on this step ; 00
BNE More_motor ; more
JMP Motor_killed ; set cycle timer and wait for next motor step

To initialize the motor call table, the originator loads 'Which_motor' with the pointer and calls 'Decida_motor'.

Ck_Macro:
JSR Next_macro ; get data
STA Which_motor ; save motor seek pointer
JSR Next_macro ; get data
STA Mgroup ; save high byte
CMP $00h ; check for end of macro
BNE Got_macro ; do it if not 0
LDA Which_motor ; check low byte for 0
CMP $00h ; check for end of macro
END_MACRO:
LDA Stat_2 ;get system
AND #Nt_macro_actv ;clear request
STA Stat_2 ;update
LDA #Bored_reld ;reset bored timer
STA Bored_timer ;

NO_MACRO:
RTS ;done

NEXT_MACRO:
LDX #00H
LDA (Macro_Lo,X) ;get speech/motor table request
INC Macro_Lo ;next
BNE Mac_dat2 ;jmp in no roll over
INC Macro_Hi ;rolled over so hi +1
Mac_dat2:
RTS

; The speech and motor pointer table pointer from the sensor table, are
; a 1-999 decimal number. The assemble converts to two 8 bit numbers and
; this creates a one of four group of 128 byte pointers in each group.
; We also do 2's offset for table lookup.

CLC ; do motor
ROL Which_motor ;move hi bit to carry
ROL Mgroup ;move carry into one of four group ptr
LDA Which_motor ;offset
STA Which_word ;set speech group pointers
LDA Mgroup ;offset
STA Sgroup ;
JSR Decide_motor ;start motor routine
JSR Say_0 ;start speech routine
RTS ;back to task master

MORE_MOTOR:
LDA Stat_3 ;system
ORA #Motor_on ;flag on mode
STA Stat_3 ;update
LDA Mon_len ;get length of on pulae
STA Motor_pulse ;set timer
LDA Stat_2 ;get system
ORA #Motor_actv ;set motor in motion
STA Stat_2 ;update

MCALC_LO:

; When motor stope, if the IR detector is on the slot in the wheel, no
; action is needed. If passed the slot, when the next motion command
; occurs,
; if the direction is the same as the last motion, no action is needed.
; If the direction is opposite to last motion then we decrement or
LDA Motor_lo ; get data
CMP Pot_timeL ; check for same
BNE Tst_fwdmors ; jump if not 0
LDA Stat_2 ; get system
AND #Motor_inactv ; clear active flag
STA Stat_2 ; update system
JMP Endtask_2 ; bail out
Tst_fwdmors: 
CLC
SBC Pot_timeL ; get current position
BCC Go_rev ; if borrow then dec command
Go_fwd:
LDA Port_C ; get IR detector
AND #Pos_sen ;
BEQ Go_fwd2 ; bypass if sensor is over slot in wheel
LDA Stat_2 ; get system
AND #Motor_frd ; get direction motor was last headed
BNE Go_fwd2 ; if set then new direction is same as last
DEC Pot_timeL2 ; compensate for counter direction reversal
Go_fwd2: 
LDA Stat_2 ; get system
ORA #Motor_fwd ; set = motor fwd (inc)
ORA #Motor_actv ; set motor in motion
STA Stat_2 ; update system
LDA Stat_3 ; get current status
ORA #Motor_off ; turn both motors off
AND #Motor_fwds ; move motor in fwd dir
JMP End_rev ; go finish port setup
Go_rev:
LDA Port_C ; get IR detector
AND #Pos_sen ;
BEQ Go_rev2 ; bypass if sensor is over slot in wheel
LDA Stat_2 ; get system
AND #Motor_frd ; get direction motor was last headed
BEQ Go_rev2 ; if clr then new direction is same as last
INC Pot_timeL2 ; compensate for counter direction reversal
Go_rev2: 
LDA Stat_2 ; get system
AND #Motor_rsv ; clear fwd flag
ORA #Motor_actv ; set motor in motion
STA Stat_2 ; update system
LDA Stat_3 ; get current status
ORA #Motor_off ; turn both motors off
AND #Motor_rsvs ; move motor in rev dir
End_rev: 
STA Stat_3
JMP Endtask_2 ; done
Go_motor:

; increment, based on new direction, to compensate for the slot which
; will be counted twice.
; jmp Byp_motorS3

LDA Stat_0 ; system
AND #Init_Mspeed ; ck if motor to do sped test
BEQ Byp_motorS3 ; only runs on wake up
LDA Stat_0 ; system
AND #Init_motor ; ck if motor to do sped test
BEQ Byp_motorS3 ; only runs on wake up
LDA Stat_0 ; system
AND #Init_Init_motor ; done
STA Stat_0 ; update

LDA #00 ; reset opto sped counter
STA Mot_opto_cnt ; set it
LDA #Opto_spd_relid ; get timer value for sped test
STA Mot_sped_cnt ; set it

Byp_motorS2:

LDA Mot_sped_cnt ; get timer
BNE Byp_motorS3 ; do nothing if >0

LDX Mot_opto_cnt ; get wheel count during sped test
LDA Motor_speed.X ; get motor on pulse width
STA Mon_len ; on time
LDA #Mpulse_on+1 ; max cycle time on/off
CLC
SEC Mon_len ; get complmut
STA Moff_len

ECS Byp_motorS3 ; jump if not neg
LDA #00
STA Moff_len

LDA Stat_0 ; system
AND #NT_Init_Mspeed ; clear motor to do speed test
STA Stat_0 ; update

Byp_motorS3:

; On power up we preset Mon_len to 11 and Moff_len to 5. This prevents
; the motor from destroying itself when the batteries are 6.4v.
; This also gives a timed count on the speed test of -7 difference.
; So I adjusted the table to compensate for the shift.

; Compare motor position to see if at destination yet

LDA Stat_2 ; get direction
AND #Motor_fwd ; set=inc clr=dec
BEQ Motor_dec ;

; bit was set so motor in inc condition

FCalc_lo:

LDA Motor_lo ; get data
CLC ; carry=0
SBC  Pot_timeL ;table - current cap time
BCC  Motor_killfwd ;jump if result negative
JMP  Endmotor ;wait till there & pulse for speed

; Reverse direction......
Motor_dec:
    ; go reverse
    LDA  Pot_timeL ;destination
    CLC
    SBC  Motor_lo ;table position to seek to
    BCC  Motor_killrev ;jump if result negative
    JMP  Endmotor jwait till there & pulse for speed

Motor_killfwd:
    LDA  Drift_counter ;ck how far we travled
    TAX ;prep for drift table
    CLC
    SBC #20 ;ck if less than 20 steps
    BCC  M_killf2 ;jump if less
    LDA #Drift_long ;long delay if >10 steps
    JMP  M_killf3 ;go fini

M_killf2:
    LDA  Drift_table,X ;get brake pulse
    LDA #Drift_short ;short delay if < 10 steps
    JMP  M_killf3 ;go fini

M_killf3:
    STA  Drift_rev ;save
    JMP  Motor_killend ;go shut down motor

Motor_killrev:
    LDA  Drift_counter ;ck how far we travled
    TAX ;prep for drift table
    CLC
    SBC #20 ;ck if less than 20 steps
    BCC  M_killr2 ;jump if less
    LDA #Drift_long ;long delay if >10 steps
    JMP  M_killr3 ;go fini

M_killr2:
    LDA  Drift_table,X ;get brake pulse
    LDA #Drift_short ;short delay if < 10 steps
    JMP  M_killr3 ;go fini

M_killr3:
    STA  Drift_fwd ;save
    JMP  Motor_killend ;go shut down motor

Motor_killend:
    LDA  Stat_3 ;get current status
    ORA #Motor_off ;turn both motors off
    STA  Stat_3 ;update
    LDA  Stat_2 ;get system
    AND #Motor_inactv ;clear activ flag
    STA  Stat_2 ;update system
    LDA Which_delay ;time till next read
    STA  Cycle_timer ;reset it
    LDA #00
    STA  TEMP1 ;used to test motor drift between seeks
    JMP  Endtask_2

; Drift table controls the magnitude of braking pulse applied.
; If the distance just travled is less than 20 then use that number
; to point into table and get new brake pulse length.

Drift_table:
    DB  24,30,32,34,35,38,40,44,46,54,56
On wake up when the motor moves from position 0 to 134, we time it and increment a counter which is used to access this table and get the motor on pulse value.

Refer to power up preset pulse width for table pointers.

Motor_speed:

```
; DB 58,60,60,60,60,60,60,60,60,60
; DB 20,22,24,27,30,32,34,36,38
; DB 46,48,50,52,54,56,58,60,60,60
; DB 25,26,27,28,30,32,34,36,38,42,45
; DB 48,51,54,57,60,60,60,60,60,60
```

This finds the 16 bit adrs of the table and points the motor

Decide_motor:

```
LDX Which_motor ;offset ptr
LDA Mgroup ;get current group pointer
CMP #03 ;is it table group 4
BEQ Dec_mot4 ;jump if is
CMP #02 ;is it table group 3
BEQ Dec_mot3 ;jump if is
CMP #01 ;is it table group 2
BEQ Dec_mot2 ;jump if is
Dec_mot1: ;table group 1
LDA Motor_grpl,X ;get lo pointer
STA Motptr_lo ;working buffer
INX ;X+1
LDA Motor_grpl,X ;get hi pointer
JMP Dec_mot_end ;go finish load
Dec_mot2:
```

A-48
LDA Motor_grp2,X ; get lo pointer
STA Motptr_lo ; working buffer
INX ; X+1
LDA Motor_grp2,X ; get hi pointer
JMP Dec_mot_end ; go finish load

Dec_mot3: ;
LDA Motor_grp3,X ; get lo pointer
STA Motptr_lo ; working buffer
INX ; X+1
LDA Motor_grp3,X ; get hi pointer
JMP Dec_mot_end ; go finish load

Dec_mot4: ;
LDA Motor_grp4,X ; get lo pointer
STA Motptr_lo ; working buffer
INX ; X+1
LDA Motor_grp4,X ; get hi pointer

Dec_mot_end: ;
STA Motptr_hi ; working buffer
LDA Stat_2 ; system
ORA »Motor_seek ; flag system
STA Stat_2 ; update

; LDA »Motor_led_rst ; get me to led timer reload
; STA Motor_led_timer ; how long the motor IR led stays on

; Get next motor data from table according to indirect pointer.

; NOTE: we are now using DB statements in the motor table so we are back to single byte format.

More_multi_m: ;
JSR Motor_data ; last time only get 1st byte (delay)
LDA Motor_lo ; get data
STA Which_delay ; motor delay control
RTS ; done

; Test motor pulse timer and alternate on & off to keep motor speed constant through battery deterioration.

Endmotor: ;
LDA Motor_pulse ; ck pulse timer
BNE Endtask_2 ; jump if not done
LDA Stat_3 ; system
AND #Motor_on ; is it an power on pulse
BNE Motor_off ; jump if on pulse (set)
LDA Stat_3 ; system
ORA #Motor_on ; flag on mode
STA Stat_3 ; update
LDA Mon_len ; get length of on pulse
STA Motor_pulse ; set timer
; mP1s_fwd:
m LDA Stat_2 ;get system
m AND #Motor_fwd ;ck if set = motor fwd (inc)
m BEQ Pls_rev ;else go reverses
m LDA Stat_3 ;get current status
m ORA #Motor_off ;turn both motors off
m AND #Motor_fwd ;move motor in fwd dir
m JMP Pls end ;go finish port setup

;mP1s_rev:
m LDA Stat_3 ;get current status
m ORA #Motor_off ;turn both motors off
m AND #Motor_revs ;move motor in rev dir
m Pls end:
m STA Stat_3
m JMP Endtask_2 ;done

;mMotor_off: ;must be on ec turn off
m LDA Stat_3 ;system
m AND #Nmot_on ;set to power off pulse
m STA Stat_3 ;update
m LDA Off_len ;get length of off pulse
m STA Motor_pulse ;set timer
m LDA Stat_3 ;get current status
m ORA #Motor_off ;turn both motors off
m STA Stat_3 ;update
Endtask_2:
RTS ;back to Idle rtn

; --------------------------------------------------------
; Start motor/speech from macro table
;
; Because of conflicts in diagnostic routines, this routine has been
; changed to a subroutine. All normal sensors jump here, diag-s call
; direct.

Start_macro:
LDA #Bored_rel ;reset bored timer
STA Bored_timer ;

LDA Macro_lo ;save for sleepy & IR tests
STA Req_macro_lo ;
LDA Macro_hi ;save for sleepy & IR tests
STA Req_macro_hi ;

JSR Get_macro ;
JMP Idle ;done

Get_macro:

; Motor noise is triggering sound sensor hardware, so this sets the
; previously sound done flag, and the system will not respond to the
; sound sensor until the sound trigger line goes low and clears prev
; done.

LDA Stat_3 ;system
ORA #Sound_stat ;
STA Stat_3 ;sat prev dn

;------------------- end sound flag
; AGE INCRNT uses bit 7 to double age counter
LDA Age ;get bit 7 - set = counter rolled over twice
AND #80h ;get bit 7
BNE Roll_age ;bit 7 set so inc age
LDA Age
ORA #80h ;set bit 7 for next counter roll over
STA Age ;update
JMP Same_age ;done

Roll_age:
INC Age ;just grew up some
LDA Age
AND #07h ;clear bit 7
STA Age
CLC
SBC #03 ;make sure it isn't > 3 (0-3 age)
BCC Same_age ;jump if <3
LDA #03 ;max age
STA Age

Same_age:
----------------- end age

LDA Stat_2 ;system
ORA #Macro_exec ;flag request
STA Stat_2 ;update
CLC ;do speech
ROL Macro_Lo ;move hi bit to carry & get 2's offset
ROL Macro_Hi ;move carry into one of four group ptr
LDX Macro_Lo ;offset ptr
LDA Macro_Hi ;get current group pointer
CMP #03 ;is it table group 4
BEQ Dec_macro4 ;jump if is
CMP #02 ;is it table group 3
BEQ Dec_macro3 ;jump if is
CMP #01 ;is it table group 2
BEQ Dec_macro2 ;jump if is
BEQ Dec_macro_end ;go finish load

Dec_macro1: ;table group 1
LDA Macro_grpl,X ;get lo pointer
STA Macro_Lo ;working buffer
INX ;X+1
LDA Macro_grpl,X ;get hi pointer
JMP Dec_macro_end ;go finish load

Dec_macro2:
LDA Macro_grpl,X ;get lo pointer
STA Macro_Lo ;working buffer
INX ;X+1
LDA Macro_grpl,X ;get hi pointer
JMP Dec_macro_end ;go finish load

Dec_macro3:
LDA Macro_grpl,X ;get lo pointer
STA Macro_Lo ;working buffer
INX ;X+1
LDA Macro_grp3,X ;get hi pointer
JMP Dec_macro_end ;go finish load

Dec_macro4:
LDA Macro_grp4,X ;get lo pointer
STA Macro_Lo ;working buffer
INX ; X+1
LDA Macro_grp4,X ;get hi pointer
STA Macro_Hi .-working buffer
RTS

This group of speech & misc routines are used for the various game play modes, triggered by the Easter egg.

; REMEMBER TO CLEAR GAME ACTIVE STATUS WHEN DONE

; NOTE: Otomah should have a delay before the word to separate this game from the speech generated by the last sensor that triggered this game.

Otomah_lo EQU »54h .-using macro 84 for 1st word
Otomah_hi EQU *00h ,-hi byte addrs 84 = 054h

Fortdelay_lo EQU *66h .-using macro 102 for delay between speech
Fortdelay_hi EQU *00h ;hi byte addrs 102 = 066h

Game_fortune:
LDA Stat_5 ;flag used at start of game
AND #temp_gam1 ;see if prev done
BNE Gam_fort2 ;jump if done

LDA Stat_5 ;flag used at start of game
ORA #temp_gam1 ;set prev done

STA Stat_5 ;update

LDA Otomah_lo ;get macro lo byte
STA Macro_Lo ;save lo byte of Macro table entry
LDA Otomah_hi ;get macro hi byte
STA Macro_HI ;save hi byte of Macro table entry

JSR Get_macro ;go start motor/speech
JSR Notrdy ;Do / get status for speech and motor

LDA GameT_reload ;reset game timer
STA Sensor_timer ;

Gam_fort2:
JSR Test_all_sens ;go check all sensors
LDA Stat_4 ;get sensor status
AND #Do_back ;ck if back sw req
BNE Gam_fort4 ;jump if requested
LDA Stat_4 ;get sensor status
AND #Do_inv; ;ck if tilt sw req
BEQ Gam_fort3 ;jump if not requested

Gam_fort2a:
JSR Clear_allGam ;go clear all status, cancel game
JMP End_all_games ;done go say "me done"

Gam_fort3:
LDA Sensor_timer ;ck for no action timeout
BEQ Gam_fort2a ;clear all if timed out
JMP Idle ;wait for switch

Gam_fort4:
LDA Stat_4 ;get sensor status
AND #Ft_do_back ;back sw req
STA Stat_4 ;clear req
LDA #GameT_reload ;reset game timer
STA Sensor_timer ;
LDA #FortdelayLo ;get macro lo byte
STA MacroLo ;save lo byte of Macro table entry
LDA #FortdelayHi ;get macro hi byte
STA MacroHi ;save hi byte of Macro table entry
JSR Get_macro ;go start motor/speech
JSR Notrdy ;Do / get status for speech and motor

LDA Stat_1 ;get system
ORA #Half_age ;force table 1 or 2 in "Decid_age" 
STA Stat_1 ;update
LDA #80h ;get random/sequential split
STA IN_LAT ;save for random routine
LDX #00 ;make sure only gives random
LDA #10h ;get number of random selections
JSR n_seq ;go decide random/sequential

 Decid_age
JSR Decid_age ;do age calculation for table entry
LDX #Temp0 ;age offset
LDA Fortyes_S1,X ;get lo byte
STA MacroLo ;save lo byte of Macro table entry
STA Req_MACROLo ;save for game
INX ;
LDA Fortyes_S1,X ;get hi byte
STA MacroHi ;save hi byte of Macro table entry
STA Req_MACROHi ;save for game

LDX #00 ;offset

Fort_Name2:
LDA Ck_Fort_name,X ;ck lo byte
CMP #FFh ;ck for end of table (note 255 cant execute)
BEQ Fort_Name_dn ;done if is
CMP MacroLo ;ck against last speech request
Not_Fort2: ;jump if not
INX ;to hi byte
LDA Ck_Fort_name,X ;ck hi byte
CMP Macro_Hi ;ck against last speech request
BNE Not_Fort3 ;jump if not
JMP Say_Fortname ;speak it

Not_Fort3: ;jump if not
INX ;
JMP Fort_Name2 ;loop till done

Say_Fortname:
LDA Name ;current setting for table offs-
CLC
ROL A ;2's comp
TAX
LDA Name_table,X ;get lo byte
STA Macro_Lo ;save lo byte of Macro table entry
INX ;
LDA Name_table,X ;get hi byte
STA Macro_Hi ;save hi byte of Macro table entry
JSR Get_macro ;go start or speech
JSR Notrdy ;do / get status for speech and motor
LDA Req_macro_lo ;recover for game
STA Macro_Lo ;set game speech
LDA Req_macro_hi ;recover for game
STA Macro_Hi ;set game speech

Fort_Name_dr: ;go set group/table pointer for motor 4 spch
JMP Start_macro ;go set group/table pointer for motor 4 spch

; compare macro to see if we are going to call Furby's name first.

Ck_Fort_name:
DW 69
DW 77
DB FFh,FFh ;FF FF is table terminator.

; Game_Rap:
; Do_rap ;1st time thru

Grap_2:
JMP Do_rap ;1st time thru
JSR Simon_timer ;decrement bored timer
LDA Bored_timer ;system elapsed time
BEQ Rap_over ;jump if 0
JSR Test_all_sens ;go check all sensors
LDA Stat_4 ;get sensors
BEQ Grap_2 ;loop if none
AND #Do_snd ;ack for mic
BNE Do_rap ;any other sensor stops game

Rap_over:
JSR Clear_all_games ;go clear all status, cancel games
JMP End_all_games ;done go say 'me done'

A-54
Do_rapi:
LDA #00 ; clear all sensor flags
STA Stat_4
LDA #Game_reload ; get reload
STA Board_timer ; rast
LDA #80h ; get random/sequential split
STA IN_DAT ; set for random routine
LDX #00h ; make sure only gives random
LDA #10h ; get number of random selections
JSR Ran_seq ; go get random selection
LDA TEMP1 ; get decision
AND #03h ; get 1 of 4 decision
CLC
ROL A ; 2's offset
TAX
LDA Rapsong,X ; get macro lo byte
STA Macro_Lo ; save lo byte of Macro table entry
INX
LDA Rapsong,X ; get macro hi byte
STA Macro_Hi ; save hi byte of Macro table entry
JSR Get_macro ; go start motor/speech
JSR Notrdy ; Do / get status for speech and motor
JMP Great_2 ; loop

Rapsong:
DW 395 ; macro RAP song pointer
DW 396
DW 407
DW 416

HidePeek_lo EQU $DBh ; using macro 475 for startp "hide me" spch
HidePeek_hi EQU $01h ; hi byte adrs 475 = 1DBh

Hidsklost_lo EQU $DBh ; using macro 472 for "nana nana nana"
Hidsklost_hi EQU $01h ; hi byte adrs 472 = 1DBh

Hidskwon_lo EQU $B7h ; using macro 439 for "whopee"
Hidskwon_hi EQU $01h ; hi byte adrs 439 = 1B7h

Game_hideseek:
LDA #$80 ; set timer for 1 min (80 * .742)
STA HCEL_LO ; use temp ram for timer
LDA Name ; current setting for table offset
CLC
ROL A ; 2's comp
TAX
LDA Name_table.X ; get lo byte
STA Macro_Lo ; save lo byte of Macro table entry
INX
LDA Name_table.X ; get hi byte
STA Macro_Hi ; save hi byte of Macro table entry
JSR Get_macro ; go start motor/speech
JSR Notrdy ; Do / get status for speech and motor
LDA #HidePeak_lo ; get macro lo byte
STA Macro_lo ; save lo byte of Macro table entry
LDA #HidePeak_hi ; get macro hi byte
STA Macro_hi ; save hi byte of Macro table entry
JSR Get_macro ; go start motor/speech
JSR Notify ; Do / get status for speech and motor

Gam_hide2:
JSR HideS_timer ; go dec bored timer without Idle
JSR Test_all_sens ; go check all sensors
LDA Stat_4 ; get all switches
AND #Do_invert ; ck if inverted
BEQ Gam_hide2a ; jump if not inverted
JMP Gam_hide2 ; abort game and call game lost speech
JSR Clear_all_game ; go clear all status, cancels games
JMP End_all_games ; done go say "me done"

Gam_hide2a:
LDA HCEL_LO ; check for no action timeout
BNE Gam_hide2 ; wait till done to start game
LDA #00 ; clear all sensor flags
STA Stat_4 ;
LDA #242 ; set timer for 3 min (242 = .742)
STA HCEL_LO ; reset

Gam_hide4:
LDA #80h ; get random/sequential split
STA IN_DAT ; save for random routine
LDX #00 ; make sure only gives random
LDA #10h ; get number of random selections (0-0f)
JSR Ran_seq ; go decide random
AND #0Fh ; and not >16
TAX
LDA Hide_time, X ; get random timer for speech
STA Sensor_timer ;

Gam_hide5:
JSR Test_all_sens ; go check all sensors
LDA Stat_4 ; get sensor status
AND #Do_tilt ; ck if tilt sw req
BNE Gam_hide8 ; jump if requested
JSR Hides_timer ; go dec bored timer & sensor_timer
LDA HCEL_LO ; get elapsed
BEQ Gam_hide9 ; game over
LDA Sensor_timer ; get random speech timer
BNE Gam_hide5 ; loop till done

; GO SAY RANDOM WORDS TO HELP FIND HIM
LDA #80h ; get random/sequential split
STA IN_DAT ; save for random routine
LDX #00h ; make sure only gives random
LDA #10h ; get number of random selections
JSR Ran_seq ; go get random selection
LDA TEMPl ; get decision
CLC
ROL A ; 2's offset
TAX
LDA Hideseek.X ; get macro lo byte
STA Macro_Lo ; save lo byte of Macro table entry
INX
LDA Hideseek.X ; get macro hi byte
STA Macro_Hi ; save hi byte of Macro table entry
JSR Get_macro ; go start motor/speech
JSR Notrdy ; Do / get status for speech and motor
JMP Gam_hide4

Gam_hide8: ; GAME WON SPEECH
JSR Clear_all_gam ; go clear all etetue, cancel game
LDA #$idskwon_lo ; get macro lo byte
STA Macro_Lo ; save lo byte of Macro table entry
LDA #$idskwon_hi ; get macro hi byte
STA Macro_Hi ; save hi byte of Macro table entry
JMP Start_macro ; go set group/table pointer for motor & spch

Gam_hide9: ; GAME LOST SPEECH
JSR Clear_all_gam ; go clear all etetue, cancel game
LDA $03 ; number of times to call 'nana'
STA HCEL_HI

Gam_hide9e:
LDA #$idsklost_lo ; get macro lo byte
STA Macro_Lo ; save lo byte of Macro table entry
LDA #$idsklost_hi ; get macro hi byte
STA Macro_Hi ; save hi byte of Macro table entry
JSR Get_macro ; go start motor/speech
JSR Notrdy ; Do / get status for speech and motor
DEC HCEL_HI ; loop
BNE Gam_hide9a ;
JMP Idle ; done

HideS_timer:
LDA Milisec_flag ; if >0 then 742 mill seconds have passed
BEQ HideS_tdn ; bypass if 0
LDA $00 ; clear it
STA Milisec_flag
LDA HCEL_LO ; get current timer * 742mSec sec
BEQ HideS_t2 ; do nothing if 0
DEC HCEL_LO ; -1

HideS_t2:
LDA Sensor_timer ; get current timer * 742mSec sec
BEQ HideS_tdn ; do nothing if 0
DEC Sensor_timer ; -1

HideS_tdn:
RTS

Hide_time: ; for random time between calls when hiding
DB 6 ; 5 sec (x * .742)
DB 7
DB 8
DB 9
DB 10
Hide-seek: ; table of sound when Furby is hiding & waiting to be found
DW    437 ;
DW    438
DW    95
DW    96
DW    97
DW   451
DW   452
DW   437
DW   438
DW    95
DW    96
DW    97
DW   451
DW   452
DW   438

; Four byte of ram allocated for game and 5th byte is game counter.
; On start, get 4 random numbers and set the game counter to 4
; sequences.
; Furby plays the 4 sounds a waits for the sensors to respond. If it's
; wrong, then start over at beginning and if it is right then say
; whoppes
; and increment to 5 sounds,..... until all 16. If 16 correct then get
; 4 new random numbers and continue with 16 sequences.
; The invert switch bails out of the game.

Simondelay_lo  EQU  #66h ; using macro 102 for delay between speech
Simondelay_hi  EQU  #00h ; hi byte adrs 102 = 066h
Listen_me_lo   EQU  #DAh ; on start up he say "Listen Me"
Listen_me_hi   EQU  #01h ; macro 474 = 1DAh
Simon_frt_lo   EQU  #AEh ; using macro 430 for simon chooses
"tickle"
Simon_frt_hi   EQU  #01h ; hi byte adrs 430 = 1AEh
Simon_back_lo  EQU  #AFh ; using macro 431 for simon chooses "pet"
Simon_back_hi  EQU  #01h ; hi byte adrs 431 = 1AFh
Simon_snd_lo EQU #B0h ; using macro 432 for simon chooses "sound"
Simon_snd_hi EQU #01h ; hi byte addr #32 = 1B0h

Simon_light_lo EQU #B1h ; using macro 433 for simon chooses "light"
Simon_light_hi EQU #01h ; hi byte addr #33 = 1B1h

Skeyfrnt_lo EQU #0Fh ; using macro 15 for user feedback
Skeyfrnt_hi EQU #00h ; use for "front"

Skeybck_lo EQU #B2h ; using macro 434 for user feedback
Skeybck_hi EQU #01h ; use for "back"

Skeylght_lo EQU #B3h ; using macro 435 for user feedback
Skeylght_hi EQU #01h ; use for "light"

Skeysnd_lo EQU #B4h ; using macro 436 for user feedback
Skeysnd_hi EQU #01h ; use for "sound"

Simonlost_lo EQU #D8h ; lost game is macro 472
Simonlost_hi EQU #01h

Available ram not in use during this game:
HCEL_LO Counter of which sensor were on
HCEL_HI Random play ram 1
BIT_CT Random play ram 2
Task_ptr Random play ram 3
Bored_count Random play ram 4

TEMP5 Random save ram 1 ( was THA_INT ) TEMP5 used in
'RAN_SEQ'
Temp_ID2 Random save ram 2
Temp_ID Random save ram 3
Learn_temp Random save ram 4

Game_simon:
; do delay before start of game
LDA #Simondelay_lo ; get macro lo byte
STA Macro_lo ; save lo byte of Macro table entry
LDA #Simondelay_hi ; get macro lo byte
STA Macro_hi ; save hi byte of Macro table entry
JSR Notrdy ; Do / get status for speech and motor

LDA Name ; current setting for table offset
CLC
ROL A ; 2's comp
TAX
LDA Name_table,X ; get lo byte
STA Macro_lo ; save lo byte of Macro table entry
INY
LDA Name_table,X ; get hi byte
STA Macro_hi ; save hi byte of Macro table entry
JSR Notrdy ; Do / get status for speech and motor
LDA #Listen_me_lo ;get macro lo byte
STA Macro_lo ;save lo byte of Macro table entry
LDA #Listen_me_hi ;get macro lo byte
STA Macro_hi ;save hi byte of Macro table entry
JSR Get_macro ;go start motor/speech
JSR Notrdy ;Do / get status for speech and motor

LDA #Simondelay_lo ;get macro lo byte
STA Macro_lo ;save lo byte of Macro table entry
LDA #Simondelay_hi ;get macro lo byte
STA Macro_hi ;save hi byte of Macro table entry
JSR Get_macro ;go start motor/speech
JSR Notrdy ;Do / get status for speech and motor

LDA #04 ;number of sensors in 1st game
GS_rentr:
STA HCEL_LO ;load counter
STA IN_DAT ;save for later use
JSR Simon_random ;go load 2 grps of 4 ram locations

Simon1:
LDA HCEL_HI ;get 1st ram location
JSR Simon_sensor ;go to speech
JSR Rotate_play ;get next 2 bits for sensor choice
DEC IN_DAT ;-1 (number of sensors played this game)
BNE Simon1 ;loop till all speech done
JSR Recover_play ;reset random rams
LDA #GameT_reload ;reset timer
STA Bored_timer ;set
LDA #00
STA Stat_4 ;clear all sensors
LDA HCEL_LO ;get counter
STA IN_DAT ;reset it

Simon2:
JSR Test_all_sens ;go check all sensors
LDA Stat_4 ;get em
BNE Simon3 ;jump if any triggered
JSR Simon_timer ;go check for timeout
LDA Bored_timer ;
BNE Simon2 ;loop if not
JMP Simon_over ;bailout if 0

Simon3:
; do to lack of time I resort to brute force ... YUK....
LDA Stat_4 ;get which sensor
CMP #08h ;front sw
BNE Simon3a ;jump if not
LDA #Skeyfrnt_lo ;get macro lo byte
STA Macro_lo ;save lo byte of Macro table entry
LDA #Skeyfrnt_hi ;get macro hi byte
JMP Simon3dn ;go speak it

Simon3a:
CMP #10h ;back sw
BNE Simon3b ;jump if not
LDA #Skeybck_lo ;get macro lo byte
STA Macro_lo ;save lo byte of Macro table entry
LDA #Skeybck_hi ;get macro hi byte
JMP Simon3dn ;go speak it

Simon3b:
CMP #04h ;light
BNE Simon3c
LDA @Skeylght_lo ;get macro lo byte
STA Macro_Lo ;save lo byte of Macro table entry
LDA @Skeylght_hi ;get macro hi byte
JMP Simon3dn ;go speak it

Simon3c:
CMP #01h ;sound
BNE Simon3d ;jump if not
LDA @Skeyand_lo ;get macro lo byte
STA Macro_Lo ;save lo byte of Macro table entry
LDA @Skeyand_hi ;get macro hi byte
JMP Simon3dn ;go speak it

Simon3d:
CMP #Do_invert ;?
BEQ Simon3e {jump if is invert
LDA #00 ;
STA Stat_4 ;clear sensor flags
JMP Simon2 ;ignore all other sensors loop up

Simon3e:
JMP Simon_over ;bail out if is

Simon3dn:
STA Macro_Hi ;save for macro call
JSR Get_macro ;go start motor/speech
JSR Notrdy ;Do / get statue for speech and motor
LDA HCEL_HI ;get 1st ram location
AND #03 ;bit 0 & 1
TAX ;point to interpret table entry
LDA Simon_convert,X ;translat game to sensors
CMP Stat_4 ;ck for correct sensor
BNE Simon_lost ;done if wrong
LDA #00
STA Stat_4 ;clear all sensors
JSR Rotate_play ;get next 2 bits for sensor choice
DEC IN_DAT ;-1 (number of sensors played this game)
BNE Simon2 ;loop till all sensors done
JSR Simon_won ;game won
JSR Recover_play ;reset random rams
INC HCEL_LO ;increase number of sensors in next game
CLI
LDA HCEL_LO ;get current
STA IN_DAT ;reset game sensor counter
SBC #16 ;ck if max number of sensors
BCC Simon4 ;
JMP Simon1 ;loop up

Simon4:
LDA #16 ;set to max
JMP GS_reentr ;start next round

::: Simon subroutines

Simon_lost:
; LDA Stat_4 ;ck for invert sw to end game
; CMP #Do_invert ;?
; BEQ Simon_over ;bail out if is
LDA @Simonlost_lo ;get macro lo byte

A-61
STA Macro_Lo ;save lo byte of Macro table entry
LDA #Simon_lost_hi ;get macro hi byte
STA Macro_Hi ;save hi byte of Macro table entry
JSR Get_macro ;go start motor/speech
JSR Notrdy ;Do / get status for speech and motor
JMP Game_simon ;start at beginning

Simon_won:
LDA HCEL_LO ;game number (how many steps)
CLC
RCL A ; 2's offset for speech win table
TAX
LDA Simon_won_tbl X ;get lo byte
STA Macro_Lo ;save lo byte of Macro table entry
INX
LDA Simon_won_tbl X ;get hi byte
STA Macro_Hi ;save hi byte of Macro table entry
JSR Get_macro ;go start motor/speech
JSR Notrdy ;Do / get status for speech and motor
RTS

Rotate_play:
ROR Bored_count ;shfl to carry
ROR Task_ptr ;carry & shfl to carry
ROR BIT_CT ;carry & shfl to carry
ROR HCEL_HI ;carry & shfl to carry throw away lo bit
ROR Bored_count ;shfl to carry
ROR Task_ptr ;carry & shfl to carry
ROR BIT_CT ;carry & shfl to carry
ROR HCEL_HI ;carry & shfl to carry throw away lo bit
RTS

Recover_play:
LDA TEMP5 ;recover random data
STA HCEL_HI
LDA Temp_ID2
STA BIT_CT
LDA Temp_ID
STA Task_ptr
LDA Learn_temp
STA Bored_count
RTS

Simon_over:
JSR Clear_all_game ;go clear all status, cancel game
LDA #00
STA Task_ptr ;reset for normal use
JMP End_all_games ;done go say "me done"

Simon_sensor:
AND #03h ;get sensor
CLC
RCL A ; 2s offset
TAX ; offset
LDA Psimon_table,X
STA Macro_Lo
INX
LDA Psimon_table,X
STA Macro_Hi ; save hi byte of Macro table entry
JSR Get_macro ;go start motor/speech
JSR Notrdy ;Do / get status for speech and motor
RTS

Simon_delay:
LDA #Simondelay_lo ;get macro lo byte
STA Macro_Lo ;save lo bytes of Macro table entry
LDA #Simondelay_hi ;get macro hi byte
STA Macro_Hi ;save hi byte of Macro table entry
JSR Get_macro ;go start motor/speech
JSR Notrdy ;Do / get status for speech and motor
RTS

Simon_random:
JSR Random ;get random number (0-255)
STA TEMPS
STA HCEL_HI
JSR Random ;get random number (0-255)
STA Temp_ID2
STA BIT_CT
JSR Random ;get random number (0-255)
STA Temp_ID
STA Task_ptr
JSR Random ;get random number (0-255)
STA Learn_temp
STA Bored_count
RTS

Simon_timmer:
LDA Milisec_flag ;if >0 then 742mili seconds have passed
BEQ Simon_tdn ;bypass it if 0
LDA #00 ;clear it
STA Milisec_flag
LDA Bored_timer ;get current timer * 742mSec sec
BEQ Simon_tdn ;do nothing if 0
DEC Bored_timer

Simon_tdn:
RTS

Psimon_table:
DW 430 ;front switch (00)
DW 431 ;back switch (01)
DW 433 ;sound sensor (11) (lt & snd swapped in table)
DW 432 ;light sensor (10)

Simon_convert: ;converts game table to sensor table
DB 08h ;front sw
DB 10h ;back sw
DB 04h ;light
DB 01h ;sound

Simon_won_tbl: ;for each game won there ia a macro (or re-use them)
DW 72 ;0 (not used,..., place holder)
DW 72 ;1 (not used,..., place holder)
DW 72 ;2 (not used,..., place holder)
DW 72 ;3 (not used,..., place holder)
DW 72 ;4 (1st game has 4 sensors, each game adds one)
DW 72 ;5
DW 72 ; 6
DW 72 ; 7
DW 380 ; 8
DW 380 ; 9
DW 380 ; 10
DW 380 ; 11
DW 471 ; 12
DW 471 ; 13
DW 471 ; 14
DW 471 ; 15
DW 439 ; 16

End_all_games: ; when any game ends, they jump here and say done

Saygamdn_lo EQU #D9h ; using macro 473 for game over speech
Saygamdn_hi EQU #01h ;

LDA #Bored_reld ; reset bored timer
STA Bored_timer ;

LDA #Saygamdn_lo ; get macro lo byte
STA Macro_Lo ; save lo byte of Macro table entry
LDA #Saygamdn_hi ; get macro hi byte
STA Macro_Hi ; save hi byte of Macro table entry
JMP Start_macro ; go set group/table pointer for motor & spch

; Burp attack egg

Burpsnd_lo EQU #D6h ; using macro 470 for user feedback
Burpsnd_hi EQU #01h ;

Game_Burp:

JSR Clear_all_games

LDA #Bored_reld ; reset bored timer
STA Bored_timer ;

LDA #Burpsnd_lo ; get macro lo byte
STA Macro_Lo ; save lo byte of Macro table entry
LDA #Burpsnd_hi ; get macro hi byte
STA Macro_Hi ; save hi byte of Macro table entry
JMP Start_macro ; go set group/table pointer for motor & spch

;-----------------------------------------------

; Easter egg says NAME

Game_name:
JSR Clear_all_gam
LDA #Bored_reld ;reset bored timer
STA Bored_timer ;

LDA Name ;current setting for table offset
CLC
ROL A ;2's comp
TAX
LDA Name_table,X ;get lo byte
STA Macro_Lo ;save lo byte of Macro table entry
INX ;
LDA Name_table,X ;get hi byte
STA Macro_hi ;save hi byte of Macro table entry
JMP Start_macro ;go set group/table pointer for motor & speech

;Twinkle song egg

; When song is complete, if both front and back switches are pressed
; we goto deep sleep. That means only the invert can wake us up, not
; the invert, etc.

Twinklsnd_lo EQU #D5h ;using macro 469
Twinklsnd_hi EQU #01h ;
Sleep_lo EQU #A6h ;using macro 166 (before going to sleep)
Sleep_hi EQU #00h ;

Game_twinkle:

JSR Clear_all_gam
LDA #03 ;song counter
STA HCEL_LO ;set
Gtwnk:

DEC HCEL_LO ;-1
LDA Stat_2 ;Get system clear done flag
AND #Not_tch_ft ;clear previously inverted flag
AND #Not_tch_bk ;clear previously inverted flag
STA Stat_2 ;update

LDA #Bored_reld ;reset bored timer
STA Bored_timer ;

LDA #Twinkleend_lo ;get macro lo byte
STA Macro_Lo ;save lo byte of Macro table entry
LDA #Twinkleend_hi ;get macro hi byte
STA Macro_hi ;save hi byte of Macro table entry
JSR Get_macro ;go start motor/speech
JSR Notrdy ;Do / get status for speech and motor
JSR Test_all_sene ;get status
JSR Test_all_sene ;get status 2nd time for debounce
LDA Stat_4 ;switch status
AND #16h ;isolate front and back switches
CMP #12h ;
BNE Gtwnk ;if both switches pressed, goto sleep
LDA HCEL_LO ;get song loop counter
BNE Gtwnk ;loop
JMP Idle ;not so egg complete

Start_sleep:
LDA #Sleep_lo ;get macro lo byte
STA Macro_Lo ;save lo byte of Macro table entry
LDA #Sleep_hi ;get macro hi byte
STA Macro_Hi ;save hi byte of Macro table entry
JSR Get_macro ;go start motor/speech
JSR Notrdy ;Do / get status for speech and motor
LDA #11h ;set deep sleep mode
STA Deep_sleep :
JMP GoToSleep ;nity-night

;Rooster loves you egg
Roostersnd_lo EQU #D4h ;using macro 468
Roostersnd_hi EQU #01h

Game_rooster:
JSR Clear_all_game
LDA #Bored_reld ;reset bored timer
STA Bored_timer ;
LDA #Roostersnd_lo ;get macro lo byte
STA Macro_Lo ;save lo byte of Macro table entry
LDA #Roostersnd_hi ;get macro hi byte
STA Macro_Hi ;save hi byte of Macro table entry
JMP Start_macro ;go set group/table pointer for motor & spch

;unittest

;If a game requires sensor input without triggering the normal
;sensor cycle for speech, then this rtn will check all sensors for
;change and the calling game can check for the appropriate trigger
;DO NOT USE I.R. SENSOR SINCE ITS RAM LOCATIONS ARE USED IN GAMES

Test_all_sens:
JSR Get_back ;
JSR Get_Tilt ;
JSR Get_invert ;
JSR Get_front ;
JSR Get_light ;
JSR Get_sound ;
JSR Get_feed ;
RTS ;back to game

;*** Side all switch triggers when ball falls off center and I/O goes
hi.

CK_tilt: ;tilt sensor
    JSR Get_Tilt ;go ck for sensor trigger
    BCS Normal_tilt ;go fini normal spch/motor table
    JMP Idle ;no request

Get_Tilt: ;this is the subroutine entry point.
    LDA Port_D ;get I/O
    AND #Ball_side ;ck if we tilted on side
    BNE Do_bside ;jump if hi
    LDA Stat_2 ;Get system
    AND #Not_bside ;clear previously on side flag
    STA Stat_2 ;update

Side_out:
    CLC ;clear indicates no request
    RTS

Do_bside:
    LDA Stat_2 ;system
    AND #Bside_dn ;ck if previously done
    BNE Side_out ;jump if was
    LDA Stat_2 ;get system
    ORA #Bside_dn ;flag set ,only execute once
    STA Stat_2 ;update system

Normal_tilt: ;Idle rtA jumps here to complete speech/motor table

;;...
also for testing, when tilt is triggered, it resets all;
esta egg routines to allow easy entry of eggs.

    JSR Clear_all_gam ;

;；；；

JSR Life ;go tweak health/hungry counters
    BCS More_tilt ;if clear then do sensor else bail
    JMP Idle ;done

More_tilt:

LDA #Tilt_split ;get random/sequential split
    STA IN_DAT ;save for random routine

LDX #Seq_tilt ;get how many sequential selections
    LDA #Ran_tilt ;get number of random selections
    JSR Ran_seq ;go decide random/sequential
LDX Sensor_timer ;get current for training subroutine

BCS Tilt_ran ;Random mode when carry SET

LDA Sensor_timer ;ck if timed out since last action

BEQ Tilt_reset ;yep

LDA Tilt_count ;save current

STA BIT_CT ;temp store

INC Tilt_count ;if not then next table entry

LDA Tilt_count ;get

CLC

SBC #Seq_tilt-l ;ck if > assignment

BCC Tilt_side ;jump if <

LDA #Seq_tilt-l ;dont inc off end

STA Tilt_count ;

JMP Tilt_side ;do it

Tilt_reset:

LDA #00 ;reset to 1st entry of sequential

STA BIT_CT ;temp store

STA Tilt_count ;

Tilt_side:

LDA #Global_time ;get timer reset value

STA Tilt_count ;reset it

LDA BIT_CT ;Arc holds value for subroutine

Tilt_ran:

STA IN_DAT ;save decision

LDA #Tilt_ID ;which ram location for learned word count

JSR Start_learn ;go record training info

LDA IN_DAT ;get decision

JSR Decid_age ;do age calculation for table entry

LDX TEMPO ;age offset

LDA Tilt_S1,X ;get lo byte

STA Macro_Lo ;save lo byte of Macro table entry

INX

LDA Tilt_S1,X ;get hi byte

STA Macro_Hi ;save hi byte of Macro table entry

JMP Start_macro ;go set group/table pointer for motor & spch

****** Inverted ball switch triggers when ball touches top and I/O goes hi.

Ck_invert: ;upside down sense

JSR Get_invert ;go ck for sensor trigger

BCS Normal_invert ;go fini normal spch/motor table

JMP Idle ;no request

Get_invert: ;this is the subroutine entry point.
LDA Port_D ;get I/O
AND #Ball_invert ;ck if we upside down
BNE Do_binvrt ;jump if inverted (hi)

LDA Stat_2 ;Get system
AND #Not_binvrt ;clear previously inverted flag
STA Stat_2 ;update

Invrt_out:
  CLC ;clear carry indicates no sensor change
  RTS

Do_binvrt:
LDA Stat_2 ;get system
AND #Binvrt_dn ;ck if prev done
BNE Invrt_out ;jump if was
LDA Stat_2 ;get system
ORA #Binvrt_dn ;flag set, only execute once
STA Stat_2 ;update system

LDA Stat_4 ;game mode status
ORA #Do_invert ;flag sensor is active
STA Stat_4 ;update
SEC ;set indicates sensor is triggered
RTS

Normal_invert:

;******************************************************

JSR Life ;go tweak health/hungry counters
BCS More_invert ;if clear then do sensor else bail
JMP Idle ;done

More_invert:

;******************************************************

LDA #Invert_split ;get random/sequential split
STA IN_DAT ;save for random routine

LDX #Seq_invert ;get how many sequential selections
LDA #Ran_invert ;get number of random selections
JSR Ran_seq ;go decide random/sequential

LDX Sensor_timer ;get current for draining subroutines
BCS Invrt_rnd ;Random mode when carry SET

LDA Sensor_timer ;ck if timed out since last action
BEQ Invrt_reset ;yes

LDA Invrt_count ;saves current
STA BIT_CT ;temp store
INC Invrt_count ;if not then next table entry
LDA Invrt_count ;get
CLC
SBC #Seq_invert-1 ;ck if > assignment
BCC Invrt_set ;jump if <
LDA #Seq_invert-1 ;dont inc off end
STA Invrt_count ;
JMP Invrt_set ;do it

Invrt_reset:
LDA #00 ;reset to 1st entry of sequential
STA BIT_CT ;temp store
STA Invrt_count ;

Invrt_set:
LDA #Global_time ;get timer reset value
STA Sensor_timer ;reset it
LDA BIT_CT ;speech to call

Invrt_end:
STA IN_DAT ;save decision
LDA #Invrt_ID ;which ram location for learned word count
JSR Start_learn ;go record training info
LDA IN_DAT ;get back word to speak
JSR Decid_age ;do age calculation for table entry
LDX TEMP0 ;age offset
LDA Invrt_S1,X ;get lo byte
STA Macro_Lo ;save lo byte of Macro table entry
INX ;
LDA Invrt_S1,X ;get hi byte
STA Macro_HI ;save hi byte of Macro table entry
JMP Start_macro ;go set group/table pointer for motor & spch

Ck_back: ;Back touch sensor
JSR Get_back ;go ck for sensor trigger
BCS Normal_back ;go fini normal spch/motor table
JMP Idle ;no request

Get_back: ;this is the subroutine entry point.
LDA Port_C ;get I/O
AND #Touch_bck ;ck if Kirby's back is rubbed
BEQ Do_tch_bk ;jump if lo
LDA Stat_2 ;Get system
AND #Not_tch_bk ;clear previously inverted flag
STA Stat_2 ;update

Tchl_out:
CLC ;clear carry for no sensor request
RTS

Do_tch_bk:
LDA Stat_2 ;get system
AND #Tchbk_dn ;ck if prev done
BNE Tchl_out ;jump if was
LDA Stat_2 ;get system
ORA #Tchbk_dn ;flag set, only execute once
STA Stat_2 ;update system
LDA Stat_4 ;game mode status
ORA #Do_back ;flag sensor is active
STA Stat_4 ;update
SEC ;set indicates sensor is triggered
RTS

Normal_back: ;enter here to complete sensor speech/motor

;***********************************************************************
JSR Life ;go to health/hungry counters
JMP Idle ;done
More_back:

;***********************************************************************
LDA #Back_split ;get random/sequential split
STA IN_DAT ;save for random routine
LDX #Seq_back ;get how many sequential selections
LDA #Ran_back ;get number of random elections
JSR Ran_seq ;go decide random/sequential
LDX Sensor_timer ;get current for training subroutine
BCC Back_rnd ;Random mode when carry SET

LDA Sensor_timer ;ck if timed out since last action
BEQ Back_reset ;yep
LDA Tchbck_count ;save current
STA BIT_CT ;temp store
INC Tchbck_count ;if not then next table entry
LDA Tchbck_count ;get
CLC
SBC #Seq_back-1 ;ck if > assignment
BCC Back_set ;jump if <
LDA #Seq_back-1 ;don't inc off end
STA Tchbck_count ;
JMP Back_set ;do it
Back_reset:
LDA #00 ;reset to 1st entry of sequential
STA BIT_CT ;temp store
STA Tchbck_count ;

Back_set:
LDA #Global_time ;get timer reset value
STA Sensor_timer ;reset it
LDA BIT_CT ;get current pointer to tables

Back_rnd:

STA IN_DAT ;save decision
LDA #Back_ID ;which ram location for learned word count
The IR routine turns interrupts off for 100 Msec, which stops the timing chain (multiplies time by 100). This front end leaves interrupts on and sits in a loop for 5 mser to determine if IR is active and if so, executes normal IR. routine, else exits.

The way to include the IR program, I list as the following:

It shows the program prargraph from Ck_IR: to Ck_front:
of course, It also attach the IR.asm file
the IR.asm file I just make a little bit change, to make they work at any system clock assume by constant SystemClock:
please advise... >
STA TEMP4 ;
Got_IR2:
    JSR D_IR_test ;used as a subroutine for diags
    BCS New_IR ;jump if found data
    DEC TEMP4
    BNE Got_IR2 ;loop
    JMP Idle ;bail out if not
New_IR:
    JMP Normal_IR

;**************
; Begin Koball's code
;**************

D_IR_test:
    SEI
    JSR BYTE ;Tracker
    LDA #Intt_dflt ;Initialize timers, etc.
    STA Interrupts ;load reg
    ;Tracker
    LDA INDAT ;;load result to ACC
    CLS
    RTS

Normal_IR:
; There are 4 I.R. table arranged as all other tables, one for each age.
; But here we get a random number which determines which one of the
; four tables we point to and the actual number received is the one of
; sixteen selection.
LDA IN_DAT ;Tracker add
    AND #OFh ;kill hi nibble (compliment of lo nibble)
    STA IN_DAT ;saves
    CMP #08 ;test for special sneeze command
    BNE No_sneeze ;jump if not
    LDA #Really_sick-30 ;force Furby to get sick
    STA Sick_counter ;update
No_sneeze:
    LDA Bored_timer ;get current count
    STA TEMP1 ;save
Gst_IR_rd:
    JSR Random ;get something
    DEC TEMP1 ;-1
    BNE Get_IR_rd ;loop getting random numbers
    LDA #seed_1 ;get new random pointer
    AND #OFh ;kill hi nibbles
    STA TEMP1 ;save
    CLC
    SEC #11 ;ck if > 11
    BCC NormIR_2 ;jump if not
    LDA #96 ;point to table 4
    JMP Got_normIR
NormIR_2:
    LDA TEMP1 ;recover random number
    CLC

A-73
SBC #C7 ;ck if > 7
BCC NormIR_3 ;jump if not
LDA #64 ;point to table 3
JMP Got_normIR ;
NormIR_3:
LDA TEMPl ;recover random number
CLC
SBC #03 ;ck if > 03
BCC NormIR_4 ;jump if not
LDA #32 ;point to table 2
JMP Got_normIR ;
NormIR_4:
LDA #00 ;force table 1
Got_normIR:

CLC
ROL IN_DAT ;16 bit offset for speech
CLC
ADC IN_DAT ;create speech field offset pointer
TAX ;set offset
LDA IR_S1 ,X ;get lo byte
STA Macro_Lo ;save lo byte of Macro table entry
INX
LDA IR_S1 ,X ;get hi byte
STA Macro_Hi ;save hi byte of Macro table entry
JMP Start_macro ;go set group/table pointer for motor &

spch

Include IR2.Asm ;asm file

******** end Tracker

*********
************
************
************
************

Ck_front: ;touch front (tummy)
JSR Get_front ;go ck for sensor trigger
BCE Normal_front ;go fini normal spch/motor table
JMP Idle ;no request

Get_front: ;this is the subroutine entry point.
LDA Port_C ;get I/O
AND #Touch_front ;ck if Firby's chest is rubbed
BEC Do_tch_ft ;jump if lo
LDA Stat_2 ;Get system
AND #Not_tch_ft ;clear previously inverted flag
STA Stat_2 ;updat

Touch_end:
CLC ;clear indicates no sensor request
RTS ;

Do_tch_ft:
LDA Stat_2 ;get system
AND #Tchft_dn ;ck if prev done
BNE Touch_end ;jump if was
LDA Stat_2 ; get system
ORA #Tchfnt_dn ; flag set, only executes once
STA Stat_2 ; update system
LDA Stat_4 ; game mode status
ORA #Do_tummy ; flag sensor is active
STA Stat_4 ; update
SEC ; set indicates sensor is triggered
RTS

Normal_front: ; enter here to complete sensor speech/motor

;*******************************************************
JSR Life ; go tweak health/hungry counters
BCC More_front ; if clear then do sensor else bail
JMP Idle ; done

More_front:

;*******************************************************
LDA #Front_split ; get random/sequential split
STA IN_DAT ; save for random routine
LDX #Seq_front ; get how many sequential selections
LDA #Ran_front ; get sequential split
JSR Ran_seq ; go decide random/sequential
LDX Sensor_timer ; get current for training subroutine
BCS Front_rnd ; Random mode when carry set

LDA Sensor_timer ; ck if timed out since last action
BEQ Front_rst ; yep

LDA Tchfrnt_count ; save current
STA BIT_CT ; temp store
INC Tchfrnt_count ; if not then next table entry
LDA Tchfrnt_count ; get
CLC
SBC #Seq_front-1 ; ck if > assignment
BCC Front_set ; jump if <
LDA #Seq_front-1 ; dont inc off end
STA Tchfrnt_count ;
JMP Front_set ; do it

Front_rst:
LDA #00 ; reset to 1st entry of sequential
STA BIT_CT ; temp store
STA Tchfrnt_count ;

Front_set:
LDA #Global_time ; get timer reset value
STA Sensor_timer ; reset it
LDA BIT_CT ; get current pointer to table

Front_rnd:
STA IN_DAT ; save decision
LDA #Front_ID ;which ram location for learned word count
(Joffset)
JSR Start_learn ;go record training info
LDA IN_DAT ;get back word to speak
JSR Decid_age ;do age calculation for table entry
LDX TEMPO ;age offset
LDA Tfrnt_S1.X ;get lo byte
STA Macro_Lo ;save lo byte of Macro table entry
INX ;
LDA Tfrnt_S1.X ;get hi byte
STA Macro_Hi ;save hi byte of Macro table entry
JMP Start_macro ;go set group/table pointer for motor & spch

;---------------------------------------------------------------
;---------------------------------------------------------------
;---------------------------------------------------------------
;---------------------------------------------------------------
;
;
Ck_feed: ; food sensor

JSR Get_feed ;go ck for sensor trigger
BCS Normal_feed ;go fini normal spch/motor table
JMP Idle ;no request

Get_feed: ;this is the subroutine entry point.

; Each trigger increments the health status at a greater rate

; Special enable routine to share port pin D1 with invert switch.
; Feed switch is pulled hi by the DAC1 (aud-a) output only after
; we test the invert line. If invert is not hi, then turn on
; DAC1 and ck feed line on same port D1.

LDA Port_D ;get I/O
AND #Ball_invert ;ck if we are inverted
BEC St_feed ;jump if not inverted (lo=not inverted)
CLC ;indicates no request
RTS ;if inverted then bypass

St_feed:
LDA #FFh ;turn DAC2 on to enable feed switch
STA DAC2 ;out
LDA Port_D ;get I/O
AND #Ball_invert ;ck if feed switch closed
BNE Start_feed ;jump if hi
LDA #00
STA DAC2 ;clear feed sw enable
LDA Stat_3 ;Get system
AND #Not_feed ;clear previously inversed flag
STA Stat_3 ;update

Feed_out:
CLC ;clear indicates no request
RTS ;go test next

Start_feed:
LDA #00

A-76
STA DAC2 ;clear feed sw enable

LDA Stat_3 ;get system
AND #Feed_dn ;ck if prev done
BNE Feed_out ;jump if was
LDA Stat_3 ;get system
ORA #Feed_dn ;flag set ,only execute once
STA Stat_3 ;update system

LDA Stat_4 ;game mode status
ORA #Do_feed ;flag sensor is active
STA Stat_4 ;update
SEC ;set when sensor is triggered
RTS ;

Normal_feed: ;enter here to complete speech/motor

; health table calls here and decision for which speech pattern
LDA #Food ;each feeding increments hunger counter
CLC
ADC Hungry_counter ;feed him!
BCC Feeding_dn ;jump if no roll over
LDA #FFh ;max count

Feeding_dn:
STA Hungry_counter ;update

;;; JSR Life ;go finish sick/hungry speech

;---------------------------------------------------------------
LDA #Feed_split ;get random/sequential split
STA IN_DAT ;save for random routine

LDX #Seq_feed ;get how many sequential selections
LDA #Ran_feed ;get random assignment
JSR Ran_seq ;go decide random/sequential

LDX Sensor_timer ;get current for training subroutine
BCS Feedrand ;Random mode when carry set

LDA Sensor_timer ;ck if timed out since last action
BEQ Feed_reset ;yep

LDA Feed_count ;save current
STA BIT_CT ;temp store

INC Feed_count ;if not then next table entry
LDA Feed_count ;get

CLC
SBC #Seq_feed-1 ;ck if > assignment
BCC Feed_out ;jump if <
LDA #Seq_feed-1 ;don't inc off end
STA Feed_count ;
JMP Feed_set ;do it

Feed_reset:
LDA #00 ; reset to 1st entry of sequential
STA BIT_CT ; temp store
STA Feed_count

Feed_set:
LDA #Global_time ; get timer reset value
STA Sensor_timer ; reset it
LDA BIT_CT ; get current pointer to tables

Feedrand:
STA IN_DAT ; save decision
LDA #Feed_ID ; which ram location for learned word count

JSR Start_learn ; go record training info
LDA IN_DAT ; get back word to speak

JSR Decid_age ; do age calculation for table entry
LDX TEMP0 ; age offset
LDA Feed_S1,X ; get lo byte
STA Macro_Lo ; save lo byte of Macro table entry
INX
LDA Feed_S1,X ; get hi byte
STA Macro_Hi ; save hi byte of Macro table entry
JMP Start_macro ; go set group/table pointer for motor & speech

Ck_light: ; Bright light sensor

JSR Get_light ; now handled as a subroutine
BCC Ck_light2 ; jump if new level > reff
JMP Idle ; nothing to do

Ck_light2:
JMP Normal_light ; jump if new level > reff

Include Light5.asm ; asm file

Normal_light:
; below routines are jumped to by light exec if > reff

JSR Life ; go tweak health/hungry counters
BCS More_light ; if clear then do sensor else bail
JMP Idle ; done

More_light:

LDA #Light_split ; get random/sequential split
STA IN_DAT ; save for random routine

A-78
LDX #Seq_light ; get how many sequential selections
LDA #Run_light ; get sensor split table
JSR Run_seq ; go decide random/sequential
LDX Sensor_timer ; get current for training subroutine
BCS Lghtrand ; Random mode when carry set
LDA Sensor_timer ; ck if timed out since last action
BEQ Light_reset ; yep
LDA Light_count ; save current
STA BIT_CT ; temp store
INC Light_count ; if not then next table entry
LDA Light_count ; get
CLC
SBC #Seq_light-1 ; ck if > assignment
BCC Light_set ; jump if <
LDA #Seq_light-1 ; dont inc off end
STA Light_count ;
JMP Light_set ; do it

Light_reset:
LDA #00 ; reset to 1st entry of sequential
STA BIT_CT ; save temp store
STA Light_count ;

Light_set:
LDA #Global_time ; get timer reset value
STA Sensor_timer ; reset it
LDA BIT_CT ; get current pointer to tables

Lghtrand:
STA TEMP4 ; save seq/rand pointer
LDA Stat_3 ; system
AND #Light_stat ; ck bit for light/dark table
BEQ Do_dark ; jump if clear
LDA TEMP4 ; get pointer
STA IN_DAT ; save decision
LDA #Light_ID ; which ram location for learned word count
(offset)
JSR Start_learn ; go record training info
LDA IN_DAT ; get back word to speak
JSR Decid_age ; do age calculation for table entry
LDX TEMP0 ; age offset
LDA Light_Sl,X ; get lo byte
STA Macro_Lo ; save lo byte of Macro table entry
INC
LDA Light_Sl,X ; get hi byte
STA Macro_Hi ; save hi byte of Macro table entry
JMP Start_macro ; go set group/table pointer for motor & spch

Do_dark:
LDA TEMP4 ; get pointer
STA IN_DAT ; save decision
LDA #Dark_ID ; which ram location for learned word count
JSR Start_learn ; go record training info
LDA IN_DAT ; get back word to speak
Jsr Decid_age ; do age calculation for table entry
LDX TEMPO ; age offset
LDA Dark_Sl,X ; get lo byte
STA Macro_Lo ; save lo byte of Macro table entry
INX ;
LDA Dark_Sl,X ; get hi byte
STA Macro_Hi ; save hi byte of Macro table entry
JMP Start_macro ; go set group/table pointer for motor & speech

---------------------------------
;j which ram location for learned word count
;go record training info
;do age calculation for table entry
;get back word to speak
;do age calculation for table entry
;set group/table pointer for motor & speech

Ck_sound: ; Audio sensor
Jsr Get_sound ; now handled as a subroutine
Bcs Ck_sound2 ; jump if new level > reff
Jmp Idle ; nothing to do
Ck_sound2:
Jmp Normal_sound ; jump if new level > reff

Get_sound: ; alt entry for diagnostics

; The microphone interface generates a square wave of 2k to 100k.
; We can loop on the sense line and count time for the
; hi period to determine if sound has changed and compare it to previous
; samples.

SEI ; disable interrupts
LDX #00 ; clear
STX TEMP1 ; clear buffer
LDX #FFh ; load loop timer
STX TEMP2 ;
Ck_snd2:
DEC TEMP2 ;
Beq Ck_snd4 ; jump if timed out
lda Port_D ; get I/O
And #Mic_in ; ck sound clk is hi
Beq Ck_snd2 ; wait for it to go hi
LDX #FFh ; load loop timer
STX TEMP2 ;
Ck_snd3:
Inc TEMP1 ; count during lo clk +5
Beq Snd_over ; jump if rolled over +3
lda Port_D ; get I/O
And #Mic_in ; ck if still hi +2
Bne Ck_snd3 ; loop till lo +3
(15*166ns=2.49us)
Jmp Ck_snd4 ; done
Snd_over:
we should never get here so bail back to idle and this will
also prevent system lockup when no clk

LDA #250 ;never allow roll over
STA TEMPI

Ck_snd4: ;re-enable interrupt
CLI
JSR Kick_IRQ ;wait for motor R/C to start working again
LDA TEMPI ;get count
CLC ;clear
SEC #05 ;is diff > 5
BCC No_snd ;bail out if not

LDA Stat_3 ;system
AND #Sound_stat ;ck for prev done
BNE No_snd2 ;wait till quiet

LDA Stat_3 ;system
ORA #Sound_stat ;
STA Stat_3 ;set prev dn

LDA Stat_4 ;set indicating change > reff level
ORA #Do_snd
STA Stat_4

SEC ;carry set indicates no change
RTS

No_snd:
LDA Stat_3 ;get system
AND #Nt_snd_stat ;clear prev dn
STA Stat_3 ;update

No_snd2:
CLC ;carry clear indicates no sound
RTS ;done

Normal_sound:

; below routines are jumped to if sound pulse detected

------------------------------
JSR Life ;go tweek health/hungry counters
BCS More_sound ;if clear then do sensor else bail
JMP Idle ;done

More_sound:

------------------------------
LDA #Sound_split ;get random/sequential split
STA IN_DAT ;save for random/itine

LDX #Seq_sound ;get how many sequential selections
LDA #Ran_sound ;number of random selections
JSR Ran_seq ;go decide random/sequential
LDX Sensor_timer ; get current for training subroutine
BCE Sndrand ; Random mode when carry set
LDA Sensor_timer ; ck if timed out since last action
BEQ Snd_reset ; yep
LDA Sound_count ; save current
STA BIT_CT ; temp store
INC Sound_count ; if not then next table entry
LDA Sound_count ; get
CLC
SBC #Seq_sound-1 ; ck if > max assignment
BCC Snd_set ; jump if <
LDA #Seq_sound-1 ; dont inc off end
STA Sound_count ;
JMP Snd_set ; dc it
Snd_reset:
LDA #00 ; reset to 1st entry of sequential
STA BIT_CT ; temp store
STA Sound_count ;
Snd_set:
LDA #Global_time ; get timer reset value
STA Sensor_timer ; reset it
LDA BIT_CT ; get current pointer to tables
Sndrand:
STA IN_DAT ; save decision
LDA #Sound_ID ; which ram location for learned word count
JSR Start_learn ; go record training info
LDA IN_DAT ; get back word to speak
JSR Decid_age ; do age calculation for table entry
LDX TEMPO ; age offset
LDA Sound_SL,X ; get lo byte
STA Macro_Lo ; save lo byte of Macro table entry
INX ;
LDA Sound_SL,X ; get hi byte
STA Macro_Hi ; save hi byte of Macro table entry
JMP Start_macro ; go set group/table pointer for motor & spch

; Misc Subroutines

; SENSOR TRAINING

; Training for each sensor is set up here and the decision if the
learned
; word should be played or not.
; Temp_ID hold the ram offset for the last sensor of the learned word.
; Temp_ID2 hold the ram offset for the current sensor of the learned
; word.
; IN_DAT holds the current word the sensor chooses, and will be loaded
; with
; the learned word instead if the sensor count > the random number that
; was
; just sampled, i.e., forces learned word to play.
;
; **

; If the sensor timer is at 0 when entering here, then the LEARN_TEMP
; ram location is cleared, else the current learned word is loaded. If
; the learned word is 0 then all entries are cleared.

; When entering, check sensor timer and bail if 0. Then test if this is
; the back switch and if so then move the current sensor to previous
; sensor
; ram and increment the counter.
; If this is not the back switch, then get previous sensor ram counter
; and
; decrement it. Then move all current sensor information to previous and
; return to caller.

; Because of training difficulties, we now need two back touches to
; increment training counters. If only one occurs then the normal
decrement
; happens. This double back touch helps to prevent accidentally training
; with a new macro by hitting the back sw when it is not the macro you
; have been working with.

start_learn:
STA Temp_ID2 ; sensor ram location of counter (current sensor)
LDA Temp_ID2 ; get current sensor ID
CMP $EEh ; EF = this is the back switch (special)
BNE Not_BCK ; jump if not
CPX $00 ; check if sensor timer timed out
BNE Learn_update ; jump if is back switch and not timed out

Not_BCK:
LDA Temp_ID ; get previous sensor ram offset
CMP $EEh ; check if last was back sw
BEQ Not_learned ; jump if no sensor prev

LDX Temp_ID ; get previous sensor ram offset
LDA Tilt_learned.X ; get learned word counter from ram
CMP Learn_temp ; compare with last word
BNE Do_lrn2 ; bail out if different
LDA Tilt_lrn_cnt,X ; prev sensor counter + offset to current

sensor
CLC
SBC $00 ; learn change
STA Tilt_lrn_cnt.X ; update
BCE Do_lrn2 ; jump if > $learn_change
BPL Do_lrn2 ; jump if not negative (rolled over)
LDA $00
STA Tilt_lrn_cnt.X ; set to zero, no roll over
Do_lrn2:
LDX Temp_ID ;get sensor learn ram offset
JSR Random ;get a number
CLC
LDA Tilt_lrn_cnt.X ;get count
CMP #FFh ;check for max
BEQ Do_lrn2a ;bypass random
CLC
SBC Seed_l ;random minus learned word counter
BCC Not_learned ;if less than random then bail out
Do_lrn2a:
LDA Tilt_lrn_cnt.X ;get learned word counter from ram
AND #0Fh ;make sure never off end of table
STA Tilt_lrn_cnt,X ;also in ram
STA IN_DAT ;force learned word for sensor
Not_learned:
LDA IN_DAT ;get current sensor word
STA Learn_temp ;SAVE FOR NEXT PASS
LDA Temp_ID2 ;get current sensor
STA Temp_ID ;save in previous sensor ram
LDA Stat_0 ;system
AND #EFh ;"Train_Bk_prev" clear 2nd time thru flag
STA Stat_0 ;update
RTS ;done-ola
Learn_update:
LDA Temp_ID ;sensor ram location for last trigger
CMP #EEh ;EE= this is the back switch (special)
BEQ Not_learned ;bail out if last trigger was also back sw
CMP #FFh ;only happens on power up
BEQ Not_learned ;false call
LDA .Stat_0 ;system
AND #Train_Bk_prev ;is this the 1st or 2nd time thru
BNE Lrn_updl ;jump if 2nd back sw hit
LDA Stat_0 ;system
ORA #Train_Bk_prev ;this is 1st time
STA Stat_0 ;update
RTS ;my job is done here!
Lrn_updl:
LDA Stat_0 ;system
AND #EFh ;"Train_Bk_prev" clear 2nd time thru flag
STA Stat_0 ;update
LDX Temp_ID ;sensor ram location for last trigger
LDA Tilt_lrn_cnt.X ;get learned word counter from ram
CMP Learn_temp ;check for training of same word
BEQ Lrn_updl2 ;jump if is
LDA Learn_temp ;get new word trainer wants to use
STA Tilt_lrn_cnt,X ;update new word
LDA #00 ;reset to 0 for new word to train
STA Tilt_lrn_cnt.X
JMP Not_learned ;done for now
Lrn_updl2:
CLC
LDA Tilt_lrn_cnt.X ;get learned word counter from ram
; on 1st cycle of new learn, we set counter 1/2 way .... (chicken)

BNE Lrn_upd2a ; jump if not 0
LDA #80h ; 1/2 way point
STA Tilt_lrn_cnt,X ; update sensor counter
JMP Clear_learn ; go finish

Lrn_upd2a:
;---------- end 1st cycle preload

ADC #Learn_chg ; add increment value
BCE Learn_overflw ; jump if rolled over
STA Tilt_lrn_cnt,X ; update sensor counter
JMP Clear_learn ; go finish

Learn_overflw:
LDA #FFh ; set to max
STA Tilt_lrn_cnt,X ; save it

Clear_learn:
JMP Do_lrn2 ; done

; When IRQ gets turned off, and then restarted, we wait two complete
; cycle to insure the motor R/C pulses are back in sync.

Kick_IRQ:
LDA Stat_3 ; get system
AND #Nt_IRQdn ; clear IRQ occurred status
STA Stat_3 ; update system
LDX #03 ; loop counter

Kick2:
LDA Stat_3 ; system
AND #IRQ_dn ; check if IRQ occurred
BEQ Kick2 ; wait till IRQ happens
LDA Stat_3 ; get system
AND #Nt_IRQdn ; clear IRQ occurred status
STA Stat_3 ; update system
DEX ; -1
BNE Kick2 ; loop till done
RTS ; is done

;公司在A-85
; Enter with 'TEMPO' holding adrs of 0-63. Areg holds lo byte and
; Xreg holds hi byte. If carry is clear then it was succesfull, if
; carry is set the write failed.

; MODIFIED eeprom, load lo byte in temp1 and hi byte in temp2
; and call EEWRT2.

LDA #00  ;use DAC output to put TI in reset
STA DAC1
SEI      ;turn IRQ off

LDA #00  ;EEPROM adrs to write data to
STA Sgroup ;save adrs
LDA #13  ;number of ram adrs to transfer [x/2]
STA Which_delay ;save
LDA #00  ;Xreg offset
STA Which_motor ;save

; Need one read cycle before a write to wake up EEPROM

LDX Which_motor ;eeprom address to read from
JSR EEREAD     ;get data (wakes up eeprom)

Write_loop:

LDA Sgroup     ;get next EEPROM adrs
STA TEMP0      ;buffer
LDX Which_motor ;ram source
LDA Age,X      ;lo byte (data byte #1)
STA TEMP1      ;save data bytes
INC Which_motor
INX
LDA Age,X      
STA TEMP2      ;hi byte (data byte #2)
JSR EEWRT2    ;send em
BCS EEfail     ;jump if bad
INC Sgroup     ;0-63 EEPROM adrs next
INC Sgroup     ;0-63 EEPROM adrs next (eeprom writes 2
bytes)
INC Which_motor ;next adrs
DEC Which_delay ;how many to send
BNE Write_loop ;send some more
RTS

; READ EEPROM HERE AND SETUP RAM

S_EEPROM_READ:

; Xreg is the adrs 0-63, system returns lo byte in Areg & hi byte in
; Xreg.

; on call: X = EEPROM data address (0-63)
; on return: ACC = EEPROM data (low byte) [also in TEMP0]
; X = EEPROM data (high byte) [also in TEMP1]
LDA  #00 ; use DAC output to put TI in reset
STA  DAC1 
SEI  ; turn IRQ off
LDX  #00 ; EEPROM address to read from
JSR  EEREAD ; get data (one read to init system)

LDA  #00 ; EEPROM adrs to read
STA  Sgroup ; save adrs
LDA  #13 ; number of ram adrs to transfer (x/2)
STA  Which_delay ; save
LDA  #00 ; Xreg offset to write ram data
STA  Which_motor ; save

Read_loop:

LDX  Sgroup ; EEPROM adrs
JSR  EEREAD ; get data

LDX  Which_motor ; ram destination
LDA  TEMPO ; get data
STA  Age,X ; lo byte (data byte #1)
INC  Which_motor ;
INX
INC  Sgroup ; 0-63 EEPROM adrs next
LDA  TEMPO1 ; get data
STA  Age,X ; lo byte (data byte #2)
INC  Which_motor ; next adrs
INC  Sgroup ; 0-63 EEPROM adrs next
DEC  Which_delay ; how many to get
BNE  Read_loop ; send some more

LDA  #00 ; clear rams used
STA  Sgroup 
STA  Which_motor ;
STA  Which_delay ;

CLI  ; Enable IRQ
JSR  Kick_IRQ ; wait for interrupt to restart
JSR  TI_reset ; go init TI (uses 'Cycle_timer')

; Begin Koball’s code

; Enable or Disable EEPROM by setting/clearing CS
; (CS = B.0)
; on call: --
; on return: --
; stack usage: 0
; RAM usage: B_IMG

A-87
**EEENA:**

LDA Port_B_image ; get prev state of port B,
ORA #001H ; turn on B.0
JMP EEE02 ;

**EEDIS:**

LDA Port_B_image ; get prev state of port 3,
AND #0FEH ; turn off B.0

**EEE02:**

STA Port_B ; output to port
STA Port_B_image ; and save port image
RTS ;

**OUTBIT:**

BCS OUTB02 ; branch if output bit = 1

LDA Port_A_image ; get prev state of port A,
AND #0FDH ; turn off A.1.
JMP OUTB04 ;

**OUTB02:**

LDA Port_A_image ; get prev state of port A,
ORA #002H ; turn on A.1.

**OUTB04:**

STA Port_A ; output bit to port
STA Port_A_image ; and save image

; toggle EEPROM clock

**TOGCLK:**

LDA Port_A_image ; get prev state of A
ORA #001H ; turn on A.0,
STA Port_A ; output to port
NOP ; delay
NOP ;
NOP ;
AND #0FEH ; turn off A.0
STA Port_A ; output to port
; Read data 16-bit data word from EEPROM at specified address
; on call: X = EEPROM data address (0-63)
; on return: ACC = EEPROM data (low byte)
; X = EEPROM data (high byte)
; stack usage: 2
; RAM usage: TEMPO

EEEREAD:

STX TEMPO ; store data addr
JSR EEENAA ; turn on CS
SEC
JSR OUTBIT
SEC ; send READ opcode (10)
JSR OUTBIT
CLC
JSR OUTBIT

LDX #6 ; init addr bit count
ROL TEMPO ; align MS addr bit in bit 7
ROL TEMPO

EERD02:

ROML TEMPO ; shift address bit into carry
JSR OUTBIT ; send it to EEPROM
DEX ; bump bit counter
BNE EERD02 ; and repeat until done

LDX #16 ; init data bit count
LDA #0
STA TEMPO ; init data bit accumulators
STA TEMPI

EERD04:

JSR TOOCLK ; toggle clock for next bit
LDA $020H ; test data bit (B.5) from EEPROM
BIT Port_5
BNE EERD08

CLC
JMP EERD10 ; EEPROM data bit = 0

EERD08:

SEC ; EEPROM data bit = 1

EERD10:

ROL TEMPO ; rotate data bit into 16-bit
ROL TEMPI ; accumulator
DEX ; bump bit counter
BNE EERD04 ; and repeat until done

JSR EEDIS ; turn off CS and return
LDA TEMPO ; set w/data byte in ACC
LDX TEMPI ; and X regs
RTS

; Issue ERASE/WRITE ENABLE or DISABLE instruction to EEPROM
; (instruction = 1001100000)
; on call: --
on return: --
stack usage: 2
RAM usage: TEMP3

EEWEN:
  LDA #OFFH ; set up enable inst
  JMP EEWE02

EEWDS:
  LDA #000H ; set up disable inst

EEWE02:
  STA TEMP3 ; save instruction
  JSR EEENA ; turn on CS
  SEC ; send start bit
  JSR OUTBIT
  CLC ; send ENA/DIS opcode (00)
  JSR OUTBIT
  CLC
  JSR OUTBIT
  LDX #6 ; init instr bit count

EEWE04:
  RCL TEMP3 ; shift instruction bit into carry
  JSR OUTBIT ; send it to EEPROM
  DEX ; bump bit counter
  BNE EEWE04 ; and repeat until done
  RTS

Write data byte to EEPROM at specified address
; on call: TEMPO = EEPROM data address (0-63)
; ACC = data to be written (low byte)
; X = data to be written (high byte)
; on return: C = 0 on successful write cycle
; C = 1 on write cycle time out
; stack usage: 4
RAM usage: TEMP0, TEMP1, TEMP2

;EEWRIT:
STA TEMP1 ;save data bytes
STX TEMP2 ;

EEWRIT2:

; JSR EEWEN ;send write enable inst to EEPROM
JSR EEDIS ;set 'low
JSR EEENA ;then high again

SEC ;send start bit
JSR OUTBIT ;
CLC ;send WRITE opcode (01)
JSR OUTBIT ;
SEC ;
JSR OUTBIT ;

LDX #6 ;init addr bit count
ROL TEMP0 ;align MS addr bit in bit 7
ROL TEMP0

;EEWR02:
ROL TEMP0 ;shift address bit into carry
JSR OUTBIT ;send it to EEPROM
DEX BNE EEWR02 ;bump bit counter
and repeat until done

LDX #16 ;init data bit count

;EEWR06:
ROL TEMP1 ;shift data bit into carry
ROL TEMP2 ;
JSR OUTBIT ;send it to EEPROM
DEX BNE EEWR06 ;bump bit counter
and repeat until done

JSR EEDIS ;cycle CS low
JSR EEENA ;then high again

LDA #0 ;init write cycle
STA TEMPG ;time out counter
STA TEMP1 ;

;EEWR08:
LDA #020H ;test READY/BUSY bit (B.5)
BIT Port_B ;from EEPROM
BNE EEWR10 ;wait for write cycle to finish

DEC TEMPG ;write cycle time out counter
BNE EEWR08
DEC TEMP1 ;
BNE EEWR08

JSR EEWR10 ;time out, disable EEPROM and
SEC ;set carry to signal error
Subroutine creates sensor table entry for the selected age.
One table for each age.
Enter with Acc holding the 1-16 table selection.
Exit with Acc & Tempo holding the offset 0-FF of the 1-4 age entry.

Special condition where we have only two tables instead of 4
(where each table is called based on age), if the "half_age" bit is
set then ages 1 & 2 call table 1 and ages 3 & 4 call table 2.

Decid_age:
STA TEMPO ;save 0-FF selection
LDA Stat_l ;system
AND #Half_age ;test if this is a special 2 table select
BEQ Decid_normal ;jump if not
LDA Stat_l ;
AND #Nc_half_age ;clear req
STA Stat_l ;update system

LDA Age ;
AND #03h ;get rid of bit 7 (9th counter bit)

CLC
SBC #01 ;actual age is 0-3, test if <2
BCC Dec_age1 ;choose age 1 (actually 0 here)
JMP Spec_age2 ;choose age 2 (actually 1 here)

Decid_normal:

;mod TestR3a.... 25% of time choose age1 to add more furbish after
he is age 4.

JSR Random ;get a number
CLC
SBC #Random_age ;below this level select age 1
BCS Noapcl_age ;jump if >
LDA #00 ;set age 1
JMP Do_age ;go do it

;end mod

Noapcl_age:
LDA Age ;get current
AND #03h ;get rid of bit 7 (9th counter bit)
CMP #03 ;is it age 4
BNE Dec_age ;jump if not
LDA #96 ;point to 4th field
JMP Do_age ;finish load from table
Dec_age3:
EMP $02 ; is it age 3
BNE Dec_age2 ; jump if not
LDA $54 ; point to 3rd field
JMP Do_age ; finish load from table
Dec_age2:
EMP $01 ; is it age 2
BNE Dec_age1 ; jump if not
Spc1_age2:
LDA $32 ; point to 2nd field
JMP Do_age ; finish load from table
Dec_age1:
LDA $00 ; point to 1st field
Do_age:
STA TEMP2 ; save age offset for speech
CLC
ROR TEMP0 ; 16 bit offset for speech
LDA TEMP2 ; which table entry
ADC TEMP0 ; create speech field offset pointer
STA TEMP0 ; save
RTS

; ---------------------------------
; Random/sequential decision control for all sensors.
; Enter with Acc holding the number of random selections for sensor.
; Enter with Xreg holding number of sequential selections
; It returns with Acc holding the random selection and the carry will be cleared for a sequential mode and set for a random mode.
; NOTE: if the caller has no random selections then carry will be cleared.

Ran_eq:
STA TEMP1 ; save random max
STX TEMP5 ; save number of sequentials
LDA TEMP1 ; force cpu status CK
BEQ Seq_decison ; jump if no randoms
DEC TEMP1 ; make offset from 0
Ran_loop:
JSR Random ; get n
ROR A ; move hi nibs to lo
ROR A
ROR A
ROR A
AND $0Fh ; get lo nibble
STA TEMP2 ; save
CLC
SBC TEMP1 ; get max random number from sensor
BES Ran_loop ; loop until <= max value
LDA TEMP2 ; get new number
CMP Prev_random ; check if duplicate from last attempt
BEQ Ran_loop ; loop if success
STA Prev_random ; update for next pass
STA TEMP1 ; new
LDA TEMP5 ; check if no sequentials
BEQ Ran_decisn ;force random if none
JSR Random ;get random/sequential decision
CMP IN_DAT ;random/sequential split
;;;
CMP #80h ;>80=random else sequential
BCC Seq_decisn ;jump if less

Ran_decisn:
LDA TEMP5 ;get number of sequential for this pass
CLC
ADC TEMP1 ;add to random for correct table start point
STA TEMP1 ;update
SEC ;set carry to indicate random
RTS ;done (Acc holds answer)

Seq_decisn:
CLC ;clear carry to indicate sequential
RTS ;done (Acc holds answer)

; Random number generator.
; SEED_1 & SEED_2 are always saved through power down
; TEMP3 & TEMP4 are random temporary files.
; Acc returns with random number, Seed_l also holds random number.

Random:
LDA Seed_1 ;
STA TEMP3 ;
LDA Seed_2 ;
STA TEMP4 ;
CLC
ROL A
ROL Seed_1
CLC
ROL A
ROL seed_1
CLC
ADC TEMP4
STA Seed_2
LEA #00
ALC Seed_1
CLC
ADC TEMP3
STA Seed_1
LDA #00
INC Seed_2
ADC Seed_1
STA Seed_1
RTS ;return with random number in Acc & seed_l

;########################################################

;########################################################
Life:

; Each FEED trigger increments the HUNGRY counter by (EQU = FOOD).

; Hungry >80 (Need_food) + Sick >C0 (Really_sick) = normal sensor
; Hungry >80 (Need_food) + Sick <C0 (Really_sick) = random SICK/SENSOR
; Hungry <80 (Need_food) + Sick >C0 (Really_sick) = random HUNGRY/SENSOR
; Hungry <60 (Sick_ref) + Sick <C0 (Really_sick) = random HUNGRY/SICK

; Hungry >60 then each sensor motion increments Sick
; Hungry <60 then each sensor motion decrements Sick

; When the system does a cold boot, we set HUNGRY & SICK to FFh......

; When returning from here, carry is set if sensor should execute
; normal routine, and cleared if sensor should do nothing.

; REFF only -----
; Hungry_counter
; Sick_counter

; Food EQU 20h ; amount to increase 'Hungry' for each feeding
; Need_food EQU 80h ; below this starts complaining about hunger
; Sick_ref EQU 60h ; below this starts complaining about sickness
; Really_sick EQU COh ; below this only complains about sickness

; Hungry_dec EQU 01 ; subtract X amount for each sensor trigger
; Sick_dec EQU 01 ; subtract X amount for each sensor trigger
; Max_sick EQU see EQU

LDA Hungry_counter ; current

; mod F-rels2 :
  CLC
  SEC
  end mod

  SBC #Hungry_dec ; -X for each trigger
  BCS first_life ; jump if not neg
  LDA #00 ; reset
first_life:
  STA Hungry_counter ; get count
  CLC
  SBC #Sick_ref ; sick if getting sick
  BCS Sick_inc ; jump if not sick
  LDA Sick_counter ; current

; mod F-rels2 :
  CLC
  SEC
  end mod

; mod testr3a
  SBC #Sick_dec ; -X for each trigger
  BCS first_sick ; jump if not neg
LDA #00 ;reset
SBC #Sick_dec ;-X for each trigger
STA Sick_counter
BCC Max_Sref ;jump if neg
CLC
LDA Sick_counter ;get again
SBC #Max_sick ;ck if at minimum allowed count
BCS first_sick ;jump if not at min
Max_Sref:
LDA #Max_sick ;set to min
first_sick:
STA Sick_counter
JMP Hunger1
;end mod testr3a

Sick_inc:
INC Sick_counter ;+1 if is
BNE No_sick_inc ;jump if didnt roll over
LDA FFh ;if did the set to max
STA Sick_counter
No_sick_inc:

Hunger1:
LDA Sick_counter ;ck how sick
CLC
SBC #Really_sick ;decide if too sick to play
BCC Hunger2 ;jump if <
LDA Hungry_counter ;check how hungry he is
CLC
SBC #Need_food ;ck if getting hungry
BCC Decd_Hung_norm ;jump if is
Life_normal:
SEC ;tell sensor to do normal routine
RTS ;done

Hunger2:
LDA Hungry_counter ;check how hungry he is
CLC
SBC #Sick_ref ;ck if very hungry and i.sick
BCC Decd_Hung_sick ;only speak hungry / sick
LDA Hungry_counter ;check how hungry he is
CLC
SBC #Need_food ;ck if getting hungry
BCC Decd_Sick_norm ;jump if is
JMP Decd_Hung_sick_norm ;do hungry & sick speech
Decd_Hung_sick_norm:
JSR Random ;need 3-way decision
CLC
SBC #A0h ;hi split
LDA Life_normal >A0 = normal sensor
LDA Seed_1 ;get again
BMI Say_sick ;>80
JMP Say_hungry ;<80

Decd_Sick_norm:
JSR Random  ; go get random 50/50 decision
BMI Life_normal ;
JMP Say_hunger ;

Decd_Sick_norm:
JSR Random  ; go get random 50/50 decision
BMI Life_normal ;
JMP Say_sick ;

Decd_Hung_sick:
JSR Random  ; go get random 50/50 decision
BMI Say_hunger ;
JMP Say_sick ;

Say_hunger:
LDA #Hunger_split ; get random/sequential split
STA IN_DAT ; save for random routine

LDX #Seq_hunger ; get how many sequential selections
LDA #Ran_hunger ; get number of random selections
JSR Ran_seq ; go decide random/sequential
BCS Hunger_ran  ; Random mode when carry SET

LDA Sensor_timer ; check if timed out since last selection
BEQ Hunger_reset  ; yep
INC Hungr_count ; if not then next table entry
LDA Hungr_count ; get
CLC
SBC #Seq_hunger-1 ; check if > assignment
BCC Hunger_side ; jump if <
LDA #Seq_hunger-1 ; don't inc off end
STA Hungr_count ;
JMP Hunger_side ; do it

Hunger_reset:
LDA #00  ; reset to 1st entry of sequential
STA Hungr_count ;

Hunger_side:
LDA #Global_time ; get timer reset value
STA Sensor_timer  ; reset it
LDA Hungr_count ; get current pointer to tables

Hunger_ran:
JSR Decid_age  ; do age calculation for table entry
LDX TEMPO  ; age offset
LDA Hunger_S1,X ; get lo byte
STA Macro_Lo ; save lo byte of Macro table entry

INX
LDA Hunger_S1,X ; get hi byte
STA Macro_hi ; save hi byte of Macro table entry
JSR Get_macro ; go start motor/speech
JSR Notrdy ; Do / get status for speech and motor
CLC  ; tells sensor to do nothing
RTS

Say_sick:
LDA #Sick_split ; get random/sequential split
STA IN_DAT ; save for random routine

LDX #Seq_sick ; get how many sequential selections
LDA #Ran_sick ; get number of random selections
JSR Ran_seq ; decide random/sequential
BCS Sick_ran ; random mode when carry SET

LDA Sensor_timer ; check if timed out since last action
BEQ Sick_reset ; yep
INC Sickr_count ; if not then next table entry
LDA Sickr_count ; get
CLC
SBC #Seq_sick-1 ; check if > assignment
BCC Sick_aide ; jump if <
LDA #Seq_sick-1 ; don't inc off end
STA Sick_count ;
JMP Sick_side ; do it

Sick_reset:
LDA $00 ; reset to 1st entry of sequential
STA Sickr_count ;

Sick_side:
LDA #Global_time ; get timer reset value
STA Sensor_timer ; reset it
LDA Sickr_count ; get current pointer to tables

Sick_ran:
JSR Decid_age ; do age calculation for table entry
LDX TEMPr ; age offset
LDA Sick_Sl,X ; get lo byte
STA Macro_Lo ; save lo byte of Macro table entry
INX
LDA Sick_Sl,X ; get hi byte
STA Macro_Hi ; save hi byte of Macro table entry
JSR Get_macro ; go start motor/speech
JSR Noetry ; Do / get status for speech and motor
CLC ; tells sensor to do nothing
RTS

;.................. -----------------------------------------------
;.................. -----------------------------------------------

GoToSleep:
; save light sensor fail or sleep command in 'Seed_2' into EEPROM

LDA Stat_0 ; system
AND #Dark_sleep_prev ;
BEQ Nodrk_prev ; jump if none
LDA #01 ; set flag that it was done
STA Seed_2 ; save in EEPROM
JMP G:2

Nodrk_prev:
LDA #0 ; set flag that it was clear
STA Seed_2 ; save in EEPROM

G2:

;-----------------------------------------------
;-----------------------------------------------

; EEPROM WRITE
; Enter with 'TEMPO' holding adrs of 0-63. Areg holds lo byte and
; Xreg holds hi byte. If carry is clear then it was successful. If
; carry is set the write failed.

; MODIFIED eeprom, load lo byte in temp1 and hi byte in temp2
; call EEWRIT2.

LDA $00 ;use DAC output to put TI in reset
STA DAC1 ;
SEI ;turn IRQ off

LDA $00 ;EEPROM adrs to write data o
STA Sgroup ;save adrs
LDA #13 ;number of ram adrs to transfer (x/2)
STA Which_delay ;save
LDA $00 ;Xreg offset
STA Which_motor ;save

; Need one read cycle before a write to wake up EEPROM

LDX Which_motor ;eeprom address to read from
JSR EEREAD ;get data (wakes up eeprom)

IWrite_loop:

LDA Sgroup ;get next EEPROM adrs
STA TEMP0 ;buffer
LDX Which_motor ;ram source
LDA Age, X ;lo byte (data byte #1)
STA TEMP1 ;save data bytes
INC Which_motor ;
INX ;
LDA Age, X ;
STA TEMP2 ;hi byte (data byte #2)
JSR EEWRIT2 ;send em ;
BCS EEfail ;jump if bad

INC Sgroup ;0-63 EEPROM adrs next
INC Sgroup ;0-63 EEPROM adrs next (eeprom writes 2
bytes)
INX Which_motor ;next adrs
DEC Which_delay ;how many to send
BNE IWrite_loop ;send some more

;********************************************************************************

GoToSleep_2:

Include Sleep.asm ;
CAUTION

Any ram location written outside of IRQ can only be read in the IRQ. Likewise if written in the IRQ, then can only be read outside the IRQ. THIS WILL PREVENT DATA CORRUPTION.

NMI:
RTI ;Not used

IRQ:
PHA ;push acc on stack
PHP ;push cpu status on stack

;***** timer A = 166 uSEC ******

CkTimerA:
LDA Interrupts ;get who did it
AND #20H ;test for timerA
BNE Do_ta ;jump if is
JMP Ck_timerB :

;Do_ta:

;***** timer B = 700 uSEC ******

Ck_timerB:
LDA Interrupts ;get status again
AND #10H ;test for timer B
BNE Do_timeB ;jump if request true
JMP Intt_false ;bypass all if not

; also changed TimerB reload value from $10h to 00 in EQU

Do_timeB:

;-------------------------------

RE-CALIBRATE SWITCH for motor position

; This counter must meet a threshold to decide if the calposition switch is really engaged.
LDA Port_C ;get I/O
AND #Motor_cal ;to when limit hit
BNE No_cal_sw ;no position switch found
INC Cal_switch_cnt ;inc each time found low
BNE Cal_roll ;jump if didn't roll over (stopped on switch)
LDA $31 ;max count
STA Cal_switch_cnt ;

Cal_roll:
LDA Cal_switch_cnt ;
CLC
SBC $30 ;ck if enough counts
BCC No_lim_stp ;jump if not enough
LDA #Cal_pos_fwd ; force value
STA Pot_timeL2 ;reset both

A-100
JMP No_lim_stp ; done

No_cal_sw:
LDA #00 ; clear count if hi
STA Cal_switch_cnt ; update

-----------------------------

No_lim_stp:
LDA Wait_time ; 4 times thru loop = 2.9 mSec
BNE WTa ; >0
LDA #04 ; counter reset
STA Wait_time ; reload
JMP Timer_norm ;

WTa:
DEC Wait_time ;
JMP TimerB_dn ; bypass timers until done

Timer_norm:
; ******** Below routines run at 2.9 mSec
LDA Mot_speed_cnt ; check for active
BEQ No_spd_m ; jump if not
DEC Mot_speed_cnt ; -1

No_spd_m:
LDA motorstoped ; motor drift timer
BEQ No_mstop ; jump if done
DEC motorstoped ; -1

No_mstop:
LDA Motor_led_timer ; Motor_led timer = 742 mSec
BEQ TimeB1 ; jump if done
DEC Motor_led_timer ; -1

TimeB1:
LDA Cycle_timer ; 2.9 mSec timer = cycle reload
LQ TimeB2 ; jump if done
DEC Cycle_timer ; -1

TimeB2:
; m LDA Motor_pulse ; 2.9 mSec timer = Motor_pulse
; m BEQ TimeB3 ; jump if done
; m DEC Motor_pulse ; -1

TimeB3:
DEC Mili_sec ; -1 & allow rollover
BNE TimerB_dn ; wait for rollover (2.9 mS * 256 = 742 mSec)
INC Mili_sec_flag ; tell task rtn to decrement timers

TimerB_dn:
; ******** We could test all interrupts here as needed

; Ck2Khz:
; Ck500hz:
; Ck60hz:

; ******** Check motor position - IR slot in wheel sensor
; This version does two reads to eliminate noise and sets a done flag to
; prevent multiple counta. It also reads twice when no slot is present to
; clear the done flag.

LDA  Port_C  ;get I/O
AND  #Pos_sen ;ck position sensor
BNE  Clr_pos  ;jump if no I/O
LDA  Port_C  ;get I/O
AND  #Pos_sen ; READ 2x to prevent noise trigger
BNE  Clr_pos  ;jump if no IR trigger
LDA  Slot_vote ;get prev cycle
BEQ  Pc_done  ;bail if prev counted
LDA  #00  ;
STA  Slot_vote ;set ram to 0. (faster than setting a bit)
JMP  Force_int ;go count slot

Clr_pos:
LDA  Port_C  ;get I/O
AND  #Pos_sen ; READ 2x to prevent noise trigger
BEQ  Pc_done  ;not 2 equal reads so bypass this cycle
STA  Slot_vote ;set ram to 1. (faster than setting a bit)
JMP  Pc_done  ;

;***********************************************************************

Ext_portC:
JMP  Intt_false ;this should be turned off
; LDA  Ints  ;get status again
; AND  #01H  ;test for port C bit 1 rising edge
; BEQ  Pc_done  ;jump if not

Force_int:
; LDA  Port_D_Imag  ;system
; AND  #Motor_led  ;ck if position I.R. led ia on
; BEQ  Pc_done  ;jump if not off

LDA  Stat_2  ;get system
AND  #Motor_fwd  ;if set then FWD else REV
BEQ  Cnt_rev  ;jump if clr
INC  Pot_timeL2  ;sensor counter
CLC
LDA  Pot_timeL2  ;current
SEC  #207  ;ck for > 207
BCC  Updt_cnt  ;jump if not
LDA  #00  ;roll over
STA  Pot_timeL2  ;
JMP  Updt_cnt  ;

Cnt_rev:
DEC  Pot_timeL2  ;-1
CLC
LDA  #208  ;max count
; Pot_timeL2  ;ck for negative ( >207 )
S  Updt_cnt  ;jump if not

Cnt_vice:
LDA  #207  ;when neg roll over to max count
STA  Pot_timeL2  ;
Updt_cnt:
INC  Drift_counter  ;to be used for braking pulse

A-102
LDA  Pot_timeL2  ;get current count
STA  Pot_timeL  ;save in motor routine counter

; This routine used to calculate motor speed based on battery voltage.
LDA  Mot_speed_cnt  ;ck for active
BEQ  Pc_done  ;jump if not
INC  Mot_opto_cnt

Pc_done:
LDA  Motor_led_timer  ;ck if active (>0)
BEQ  Mot_led_off  ;jump if done
LDA  Port_D_Image  ;system
ORA  $Motor_led  ;turn LED on
JMP  Mot_led_dn

Mot_led_off:
LDA  Port_D_Image  ;system
AND  $Motor_led_off  ;turn LED off

Mot_led_dn:
STA  Port_D_Image  ;update motor led

M_drft_F1:
LDA  Drift_fwd  ;get delay value
BEQ  M_drft_F1  ;jump if prev done
LDA  Drift_fwd  ;get delay value
CMP  $01  ;01=turn motors off
BEQ  M_drft_F2  ;send it
DEC  LDA  AND  STA  LDA  ORA  AND  JMP M_drft_F2  DEC  LDA  ORA  JMP M_drft_R2  DEC

;m32
LDA  Port_D_Image  ;get system (note lo is tranys off)
AND  $3Fh  ;turn both motors off to prevent transistors
STA  Port_D  ;on at same time

;m32
LDA  Port_D_Image  ;get system
ORA  $Motor_off  ;turn both motors off
AND  $Motor_fwds  ;move motor in fwd dir to stop motion
JMP  Intt_motor_end

M_drft_F2:
DEC  LDA  Port_D_Image  ;get system
ORA  $Motor_off  ;turn both motors off
AND  $Motor_fwds  ;move motor in fwd dir to stop motion
JMP  Intt_motor_end

M_drft_R1:
LDA  Drift_rev  ;get delay value
BEQ  Intt_motor  ;jump if prev done
LDA  Drift_rev  ;get delay value
CMP  $01  ;01=turn motors off
BEQ  M_drft_R2  ;send it
DEC  LDA  Drift_rev  ;-1

;m32
LDA  Port_D_Image  ;get system (note lo is tranys off)
AND  $3Fh  ;turn both motors off to prevent transistors
STA  Port_D  ;on at same time

;m32
LDA  Port_D_Image  ;get system
ORA  $Motor_off  ;turn both motors off
AND  $Motor_fwds  ;move motor in rev dir to stop motion
JMP Intt_motor_end

M_drft_R2:
DEC Drift_rev ;-1
LDA Port_D_Image ;get system
ORA Motor_off ;turn both motors off
JMP Intt_motor_end

Intt_motor:
LDA Stat_3
AND #C0h ;get motor command bits
STA Intt_Temp ;sav. motor direction

;_____ Purby1: .. move motor pulse width to interrupt routine

LDA Motor_pulse1 ;get on time
BEQ Intmotor1 ;jump if 0
DEC Motor_pulse1 ;-1
JMP Intmotor_dn ;exit (dont change Intt_temp if on)

Intmotor1:
LDA Motor_pulse2 ;get off time
BEQ Intmotor2 ;get reset timer
DEC Motor_pulse2 ;-1
LDA #C0h ;shut motor off
STA Intt_Temp ;
JMP Intmotor_dn ;exit

Intmotor2:
LDA Mon_len ;reset on time
STA Motor_pulse1 ;
LDA Moff_len ;reset off time
STA Motor_pulse2 ;

Intmotor_dn

;----- end motor pulse width

LDA Port_D_Image ;get system
AND #3Fh ;clear motor direction bits
CLC
ADC Intt_Temp ;put in motor commands

Intt_motor_end:
STA Port_D_Image ;update system

; at Tracker

EOR #11000000 ;Tracker add invert motor drivers

; end Tracker

STA Port_D ;output

Intt_done:

; general turn

LDA Stat_3 ;syst.
ORA #IRQ_dn ;flag item IRQ occurred
STA Stat_3 ;uplat

Intt_false:
LDA #00H ;clear all ints first
STA Interrupts ;
LDA #Intt_df1 ;get default for interrupt reg
STA Interrupts ;set reg & clear intt flag

PLP ;recover CPU
PLA        ;recover ACC
RTI        ;reset interrupt

Communication protocol with the TI is:

FF is a no action command. (used as end of speech command)
FE sets the command data mode and the TI expects two
additional data bytes to complete the string. (3 TOTAL)
ALL OTHERS (0-FD) ARE CONSIDERED START OF A SPEECH WORD!
Command data structure is BYTE 1 = BYTE 2 = BYTE 3

BYTE 1 is always FE

Command 1
BYTE 2 = FE is pitch table control;
BYTE 3 = bit 7 set = subtract value from current course value
       clr = add value to current course value
bit 6 set = select music pitch table
       clr = select normal speech pitch table
bit 0-5 value to change course value (no change = 0)

Command 2
BYTE 2 = FD is Infrared transmit cmd
BYTE 3 = the I.R. code to send ( 0 - 0FH only )

Command 3
BYTE 2 = FC is the speech speed control
BYTE 3 = a value of 0 - 255 where 2EH is normal speed.

Enter subroutine with TEMPl = command byte (1st)
        TEMP2 = data byte (2nd)

Xmit_TI:
LDA      #FEH       ;tells TI command data to follow
JSR      Spch_more  ;out data
LDA      TEMPl     ;command code
JSR      Spch_more  ;out data
LDA      TEMP2     ;data to send
JSR      Spch_more  ;out data
RTS        ;done

There is an entry for each bank of speech and only the words in that
bank are in the list. This is a subroutine call.

The first time thru, we call SAY_x and as long as WORD_ACTIV or
SAY_ACTIV
is set we call DO_NEXTSFNT until sayevent is done.

There are 4 groups of 128 pointers in each group. This gives 512
; 1. Enter with 'Which_word' holding 0-127 and 'Sgroup' for the 1 of 4
tables;
which points to two byte adrs of a saysent. These two bytes are
lo-ded into Saysent_lo & Saysent_hi.

; 2. Data is shuffled to the TI according to the BUSY/REQ line

; Currently we have 167 speech words or sounds in ROM. Words 1 - 12
are in bank 0 and 13 - 122 are in bank 1 & 123 - 167 in bank 2.

Say_0:

| LDA | i ord | get offset |
| LDA | group | load offset to Xreg |
| CMP | #03   | is it table group 4 |
| BEQ | Dec_say4 | jump if is |
| CMP | #02   | is it table group 3 |
| BEQ | Dec_say3 | jump if is |
| CMP | #01   | is it table group 2 |
| BEQ | Dec_say2 | jump if is |

Dec_say1:  
LDA Spch_grp1,X ;get lo pointer  
STA Saysent_lo ;save  
INX ;X+1  
LDA Spch_grp1,X ;get hi pointer  
STA Saysent_hi ;save  
JMP Dec_say5 ;go calc word

Dec_say2:  
LDA Spch_grp2,X ;get lo pointer  
STA Saysent_lo ;save  
INX ;X+1  
LDA Spch_grp2,X ;get hi pointer  
STA Saysent_hi ;save  
JMP Dec_say5 ;go calc word

Dec_say3:  
LDA Spch_grp3,X ;get lo pointer  
STA Saysent_lo ;save  
INX ;X+1  
LDA Spch_grp3,X ;get hi pointer  
STA Saysent_hi ;save  
JMP Dec_say5 ;go calc word

Dec_say4:  
LDA Spch_grp4,X ;get lo pointer  
STA Saysent_lo ;save  
INX ;X+1  
LDA Spch_grp4,X ;get hi pointer  
STA Saysent_hi ;save

Dec_say5:  
LDA (Saysent_lo,X) ;get data @ 1r bit adrs  
STA TEMP2 ;save new speech speed  
LDA #FCh ;command for TI to except speed data  
STA TEMP1  
JSR Xmit_TI ;send it to TI  
INC Saysent_lo ;next saysent pointer  
BNE Xney_say ;jump if no roll over  
INC Saysent_hi ;+1
Xnay_say:
LDX #00 ;no offset
LDA (Saysent_lo,X) ;get data @ 16 bit adrs
CLC
ADC Voice ;adjut to voice selected on power up
STA TEMP2 ;save new speech pitch
LDA $FEh ;command for TI to except pitch data
STA TEMPI

; The math routins converts the value to 00 for 80 and
; if 70 then subtracts from 80 to get the minus var. ;n of 00
; ie, if number is 70 then TI gets ssnt 10 1-1

LDA TEMP2 ;get voice with offset
BHI No_voice_chg ;if >80 then no char
LDA #80h ;remove offset if <80
CLC
SBC TEMP2 ;kill offset
STA TEMP2 ;update

No_voice_chg:
JSR Xmit_TI ;send it to TI

Do_nextsent:

First_say:
INC Saysent_lo ;next saysent pointer
BNE Scnd_say ;jump if no roll over
INC Saysent_hi ;+1

Scnd_say:
LDX #00 ;no offset
LDA (Saysent_lo,X) ;get data @ 16 bit adrs
CMP #FFH ;check for end
BEQ Say_end ;done
LDA (Saysent_lo,X) ;get data @ 16 bit adrs
STA Which_word ;

Wtssst:
CLC
SBC #12 ;ck if in bank 1
BCS Get_group1 ;jump if is

Get_group0:
LDA #00 ;set bank
STA Bank_ptr ;Bank number
CLC ;clear carry
LDA Which_word ;get word
ROL A ;2's offset
TAX ;load offset to Xreg
LDA Word_group0,X ;get lo pointer
STA Word_lo ;save
INX ;X+1
LDA Word_group0,X ;get hi pointer
STA Word_hi ;saves
JMP Word_fini ;go do it

Get_group1:
LDA Which_word ;selection
CLC
SBC #122 ;ck if in bank 2
BCS Get_group2 ;jump if is
; WAKE2
; adds deep sleep mode. If 'Deep_sleeplh then tilt will not
; wake us up. only invert.

; Power up reset decision for three types of startup:
; 1. Powerup with feed switch zeros ram & EEPROM, & calls 10-200-10 macro.
; 2. Power up from battery change wont clear EEPROM but calls 10-200-10 macro.
; 3. Wake up from Port_D clears ram and jumps directly to startup. No macro.

; SEI
LDX #COH
STX Interrupts ;disable Watch Dog
LDX #FFH
TXS

LDX #0
LDA Wake_up
STA TEMP5 ;Get the information from hardware to check whether reset is from power up or wakeup
STX Wake_up ;disable wakeup immediately, this action can stop the reset occupied by another changed on PortD, so once the program can execute to this line then chip will not be reset due to port changed again

AND #00000001 ;mask the rest of bit and just check the port wakeup information
BEC Power_battery ;jump to power up initial if not port D

; Need to debounce tilt and invert since they are very unstable

Ck_wakeup:
LDA #00 ;clear
STA TEMP1
STA TEMP2
LDX #FFh ;loop counter

Dbnc_lp1:
LDA Port_D ;chk tilt sw
AND #01
BEQ Dbnc_lp2 ;jump if not tilt
INC TEMP1 ;switch counter

Dbnc_lp2:
LDA Port_D ;chk invert sw
AND #02
BEQ Dbnc_lp3 ;jump if not invert
INC TEMP2 ;switch counter

Dbnc_lp3:
DEX ;-1 loop count
BNE Dbnc_lp ;loop
LDA Deep_sleep ;decide if normal or deep sleep
CL
#11h
BEQ Dbnc_lp4 ;if deep sleep then only test invert
LDA TEMP1 ;get tilt count
BEQ Dbnc_lp4 ;jump if 0
CLC
SBC #, ;min count to insure not noise
BCS Power_Port_D ;jump if > min
Wafe2.asm

Dbnc_lp4:
LDA TEMP2 ;get invert count
BEQ Dbnc_lp5 ;jump if 0
CLC
SBC $10 ;min count to insure not noise
BCS Power_Port_D ;jump if > min

Dbnc_lp5:
;Verify that Port_D is no longer changing before going to sleep.
;If not, the CPU will lock up without setting the low power node.
;Before we exit here when count is less than minimum count, we must
;be sure Port_D is not changing. If we jump to sleep routine when
;it is not stable, the sleep routine will wait forever to be stable
;which causes Furby app'er to be locked up.

LDA #00 ;counter
STA TEMPI
LDA Port_D ;get current status

Test_sleep:
CMP Port_D ;check if changed
BNE Ck_wakeup ;start over if did
DEC TEMPI ;-1 counter
BNE Test_sleep ;loop
JMP GoToSleep_2 ;otherwise, just goto sleep again

Power_Port_D:
LDA #11h ;signal port D wakeup
STA Warm_cold
JMP L_PowerOnInitial

Power_battery:
LDA #05h ;signal battery wakeup
STA Warm_cold

L_PowerOnInitial:
LDA #00 ;clear deep sleep command
STA Deep_sleep

Page 2
Light5.asm

;***********************************************************************

; MODS:

; LIGHT3.asm
Add test to light counter so that if the oscillator
fails, the system will ignore light sensor and keep running.

; Light4
When goes to complete dark and hits the 'Dark_sleep' level
end stays there until the ref level updates, at that point
we send Furby to sleep.

; Light5 (used in F-HEL52 )
Change detection of light threshold to prevent false or continuous trigger.

;***********************************************************************

| Bright     | EQU 15 | ; light sensor trigger > ref level (Hon) |
| Dim        | EQU 15 | ; light sensor trigger < ref level (Hon) |
| Shift_reff | EQU 10 | ; max count to set or clear prev done flag |
| Dark_sleep | EQU 80h| ; when timer A hi = 0f and timer A low is to this EQU then send him to sleep |

; The CDS light sensor generates a square wave of 500hz to 24khz based on
; light brightness. We can loop on the sense line and count time for the
; lo period to determine if light has changed and compare it to previous
; samples. This also determines going lighter or darker. We also set a timer
; so that if someone holds their hand over the sensor and we announce it,
; if the change isn't stable for 10 second, we ignore the change back to the
; previous state. If it does exist for > 10 seconds, then it becomes the
; new sample to compare against on the next cycle.

; In order to announce light change, the system must have a consistent
; count > 'Shift_reff'.

; If a previous ref has been set then the 'Up_light' bit is set to
; look for counts greater than the ref. The system passes through the
; light routine 'Shift_reff' times. If it is consistently greater than
; the ref level, we get a speech trigger. If any single pass is less
; than the ref, the counter is set back to zero. This scenario also
; is obeyed when the trigger goes away, ie remove your hand, and the system
; counts down to zero. ('Up_light' bit is cleared) If during this time any
; trigger greater than ref occurs, the count is set back to max.
; This should prevent false triggers.

Get_light: ;alt entry for diagnostics

; This uses timer A to get a count from the lo period of the clk

SEI          ; interrupts off
LDA #00CH    ; disable timer, clock, ext ints, & watchdog; select IRQ int.
STA Interrupts
LDA #00CH    ; set timer A for timer mode
STA TMA_CON  

;***********************************************************************
LDA #000H ; re-start timer A
STA TMA_LSB ;
LDA #000H ; now CPUCLK was #010H = CPUCLK/4 (Hon)
STA TMA_MSB ;

Ck_light2:
LDA TMA_MSB ; test for dead light osc
AND #0Fh ; get timer
CMP #0Fh ; ck for > 0E
BNE Ck_light2a ; jump if not
LDA TMA_LSB ; get lo byte
CLC
SBC #0Eh ; ck for > .(msb+lsb = 0FE0)
BCC Ck_light2a ; jump if not
JMP Light_fail ; bail out if >

Ck_light2a:
LDA Port_D ; get I/O
AND #Light_in ; ck light clk is hi
BEQ Ck_light2 ; wait for it to go hi
LDA #000H ; re-start timer A
STA TMA_LSB ;
LDA #000H ; now CPUCLK was #010H = CPUCLK/4 (Hon)
STA TMA_MSB ;

Ck_light3:
LDA TMA_MSB ; test for dead light osc
AND #0Fh ; get timer
CMP #0Fh ; ck for > 0E
BNE Ck_light3a ; jump if not
LDA TMA_LSB ; get lo byte
CLC
SBC #0Eh ; ck for > .(msb+lsb = 0FE0)
BCS Light_fail ; bail out if >

Ck_light3a:
LDA Port_D ; get I/O
AND #Light_in ; ck light clk is lo
BNE Ck_light3 ; wait for it to go lo to insure the clk edge

Ck_light4:
LDA #000H ; re-start timer A
STA TMA_LSB ;
LDA #000H ; now CPUCLK was #010H = CPUCLK/4 (Hon)
STA TMA_MSB ;

Ck_light4a:
LDA Port_D ; get I/O
AND #Light_in ; ck if still lo
BEQ Ck_light4a ; loop till hi

; Timer A holds count for lo period of clk

Light4cmp:
LDA TMA_MSB ; get timer high byte
AND #00FH ; mask out high nybble
STA TEMP2 ; and save it
LDA TMA_LSB ; get timer low byte
STA TEMP1 ; and save it
LDA TMA_MSB ; get timer A high byte again
Light5.asm

AND #00FH ; mask out high nybble
CMP TEMP2 ; and compare it with last reading
BNE Light4cmp ; loop until they're equal

; take 12 bit timer (2 bytes) and move to one byte and trash low nibble
; of low byte. End up with hi 8 bits out of 12.

LDX #04 ; loop counter

Light_byte:
ROR TEMP2 ; get lo bit into carry
ROR TEMP1 ; shuffle down and get carry from TEMP2
DEX ; -1
BNE Light_byte ; loop till done

Ck_light5:

Ck_light4b:

LDA #Intt_dflt ; Initialize timers, etc.
STA Interrupts ; re-establish normal system
CLI ; re-enable interrupt
JSR Kick_IRQ ; wait for motor R/C to start working again
CLC ; clear

--- now have new count in 'TEMP1'

LDA Light_reff ; get previous sample
SEC TEMP1 ; check against current sample
BCC Ck_light5 ; jump if negative
CLC
SEC #Bright ; check if difference > reff
BCC Ck_light5 ; go do speech
BCC Ck_light5 ; go do speech
JMP Kill_ltrf ; bail out if not

Ck_light5:

CLC
LDA TEMP1 ; try the reverse subtraction
SEC Light_reff ; prev
BCC Kill_ltrf ; quit if negative
CLC
SEC #Dim ; is diff < reff
BCC Kill_ltrf ; bail out if not

Light_dim:

LDA Stat_3 ; system
AND #Nt_light_stat ; clear bit to indicate dark tabe
STA Stat_3 ; update system
JMP Do_light ; go finish

Light_brt:

LDA Stat_3 ; system
ORA #Light_stat ; set bit to indicate light table
STA Stat_3 ; update system
JMP Do_light ;

Light_fail:

LDA #FFh ; forces lo number so no conflicts
STA TEMP1
LDA #Intt_dflt ; initialize timers, etc.
STA Interrupts ; re-establish normal system
CLI ; re-enable interrupt
JSR Kick_IRQ ; wait for motor R/C to start working again
CMP Kill_shift ; resets with no req

Do_light:
LDA Stat_1 ;system
AND #Up_light ;check if increment mode
BNE Rst_shftup ;jump if increment mode
LDA #Shift_reff ;set to max
STA Light_shift ;
JMP No_l todo ;

Rst_shftup:
INC Light_shift ;+1
LDA Light_shift ;get counter
CLC
SBC #Shift_reff ;check if > max reff count
BCC No_l todo ;jump if < max count
LDA #Shift_reff ;reset to max
STA Light_shift ;

LDA Stat_0 ;system
AND #Lt_prev_dn ;check if previously done
BNE New_ltreff ;jump if was

LDA Stat_0 ;system
ORA #Lt_prev_dn ;set previously done
STA Stat_0 ;update

; LDA Stat_1 ;system
; AND #FFh ;set system to shift decrement mode
; STA Stat_1 ;update

LDA #Light_reload ;reset for next trigger
STA Light_timer ;set it
JMP Do_ltraff ;go announce it

New_ltreff:
LDA Light_timer ;get current
BNE No_l todo ;nothing to do
LDA #2FFh ;get new count
STA Light_reff ;update system

; LDA Stat_1 ;system
; AND #FFh ;set system to shift decrement mode
; STA Stat_1 ;update

LDA #2FFh ;get current value
CLC
SBC #Dark_sleep ;check if > sleep level
BCS Ck_drk ;jump if >
LDA Stat_0 ;system
AND #FFh ;kill prev done
STA Stat_0 ;update
JMP Kill_ltrf ;

Ck_drk:
LDA Stat_0 ;system
AND #Dark_sleep_prev ;check if this was already done
BNE Kill_ltraff ;jump if was

LDA Stat_0 ;system
ORA #REQ_dark_sleep ;set it
ORA #Dark_sleep_prev ;set also
STA Stat_0 ;update

Kill_ltraff:
Light5.asm

LDA Stat_0
AND #Lt_prev_dn
BEQ No_lt_todo
LDA Light_shift
BEQ Kill_shift
LDA Stat_1
AND #Up_light
BEQ Ret_shift
LDA #00
STA Light_shift
JMP No_lt_todo

Ret_shift:
DEC Light_shift
JMP No_lt_todo

Kill_shift:
LDA Stat_0
AND #FDh
STA Stat_0
LDA Stat_1
ORA #Up_light
STA Stat_1

No_lt_todo:
SEC
RTS

/** alert system to start speech **/

Do_ltchg:
LDA Stat_3
AND #Light_stat
BNE LT_ref_brt
LDA Stat_4
ORA #Do_light_dim
JMP LTref_egg

LT_ref_brt:
LDA Stat_4
ORA #Do_light_brt

LTref_egg:
STA Stat_4
CLC
RTS

;done
Diag7.asm

; 'Diagnostics and calibration Routine

; Mods to the diagnostic routines :

; DIAG6 :
; Init memory, voice, name and write EEPROM before exiting.

; Diag7:
; EEPROM memory test, reads and writes all locations.
; On power up if port D woke us, then bypass diagnostics.

; refer to self test mode documentation

;************** START
;
; Diagnostic EQU's

Dwait_tilt EQU 02 ; full test waiting for no tilt (step 1)

Diagnostic:

; All speech / motor calls use standard macro routines, except we
; force the macro directly. Be careful to load the 'MACRO_LO' and
; 'MACRO_HI' bytes properly. We use a common subroutine to set the macro
; so 'MACRO_HI' is loaded only once in the subroutine. Be sure the macros
; are in the same 128 byte block. Initially chose adrs 400 (190) for these
; diags.

LDA Warm_cold ; get startup condition
CMP #11h ; check for port D wakeup
BEQ No_Diag ; jump if not

LDX #FFh ; loop counter

0portD_tst:
LDA Port_D ; get I/O
AND #03 ; check for tilt and invert
BNE No_Diag ; if either hi then bail out
DEX r-1
BNE 0portD_tst ; loop till done (ckg for Port D bounce)

LDA Port_C ; get I/O
AND #0Ch ; check for front and back switches made
BEQ Diag1 ; if both not lo then bail out else start diag

No_Diag:
JMP Test_byp ; no diagnostic request

Diag1:
; force voice to normal condition while diag is active
LDA #9 ; Tracker add for constant diag
STA Rvoice ; Tracker add
LDA #0 ; hi beep for start of test
JSR Diag_macro ; go send motor/speech

; wait for front & back to clear
LDA Port_C ; get I/O
AND #0Ch
CMP #0Ch
BNE Diag1

New_top:
LDA #03
JSR Half_delay

; get keys

Diag2a:
; press front key & go to EEPROM test
LDA Port_C
AND #Touch_fmt
BNE Diag2b

LDA #01
JSR Diag_macro

Diag2al:
LDA Port_C
AND BEQ #Touch_fmt
RAMset:
LDA #01H
LDX #Age
STA 00,X
CLC
ADC #01
INC Acc
INX
CPX #Age+26
BNE RAMset
INC Task_ptr
LDA #00
BSR Do_EE_write
LSR S_EEPROM_READ
LDA #00
STA Task_ptr
LDX #Age
RAMtest:
LDA 00,X
CLC
ADC Task_ptr
STA Task_ptr
BNE RAMtest
INX
CPX #Age+26
BNE RAMtest
LDA Task_ptr
CPX #Age+26
CMP $5fh
BNE EEfail
BNE EEpass

EEpass:
LDA #02
STA Feed_count
JMP EEdone

EEfail:
LDA #03
STA Feed_count

EEdone:
```assembly
CLI
JSR Kick_IRQ   ; enable IRQ
JSR TI_reset   ; clear TI from
LDA Feed_count ; get 10 byte of macro to call
JSR Diag_macro ; go send motor/speech

Diag2b:
; Speaker tone / I.R. xmit
LDA Port_C     ; get I/O
AND #Touch_bck ; wait for switch
BNE Diag2c     ; go check if next test is requesting
LDA @1          ; hi beep for start of test
JSR Diag_macro ; go send motor/speech

Diag2blp:
LDA Port_C     ; go send motor/speech
AND #Touch_bck ; mask for back switch
BNE Diag2bl    ; loop until back switch pressed

Diag2bl:
LDA #04         ; send long tone (lk sinewave)
JSR Diag_macro ; go send motor/speech
LDA Port_C     ; go check if next test is requesting
AND #Touch_bck ; mask for back switch
BNE Xmit_blp   ; loop until back switch pressed

Xmit_blp:
LDA #01         ; beep
JSR Diag_macro ; go send motor/speech
LDA Port_C     ; go check if next test is requesting
AND #Touch_bck ; mask for back switch
BNE Xmit_blp   ; loop until back switch pressed
LDA #05h        ; send 'S' to I.R. xmiter
STA TEMP2
LDA #FDh        ; send command I.R. to TI
STA TEMP1
JSR Xmit_TI    ; send it

dumb: LDA Port_C     ; get I/O
AND #Touch_bck ; wait for switch
BNE dumb       ; wait for back to be pressed

dumber: LDA Port_C     ; get I/O
AND #Touch_frt ; check switch
BEQ Next_1     ; if both not lo then bail out else start diag
JMP Xmit_1p

Next_1: LDA #2          ; hi beep for start of test
JSR Diag_macro ; go send motor/speech
LDA Port_C     ; get I/O
AND #0Ch        ; check for front and back switches made
BEQ Next_1     ; if both not lo then bail out else start diag
JMP New_top

; Full test starts here
Diag2c: LDA Port_D     ; get I/O
AND #Ball_invert ; wait for switch
BNE Diag2d      ; forward if key pressed
```
JMP  Diag2a ;loop back to top if none

Diag2d:
LDA  #01 ;hi beep for start of test
JSR  Diag_macro ;go send motor/speech

; FULL TEST MODE

DiagFl: ;wait for no tilt to start full diag
LDA  #Dwait_tilt ;set delay to be sure no tilts
STA  TEMPI ;

DiagFla:
LDA  Port_D
AND  #3
BNE  DiagFl
BC  TEMPI
BNE  DiagFla
LDA  #2 ;pass beep
JSR  Diag_macro ;go send motor/speech

DiagF2: ;test tilt 45 deg
LDA  Port_C
AND  #00001100b
CMP  #0Ch
BEQ  DiagF22
LDA  #3 ;fail beep
JSR  Diag_macro ;

DiagF22:
LDA  Port_D
AND  #2
BEQ  DiagF23
LDA  #3 ;fail beep
JSR  Diag_macro ;

DiagF23:
LDA  Port_D ;get I/O
AND  #Ball_side ;ck for tilt switch (hi = tilted)
BEQ  DiagF2a ;jump to error if so
LDA  Port_D ;get I/O
AND  #Ball_invert ;ck if invert sw made
BNE  DiagF2a ;jump to error if so
LDA  Port_C ;get I/O
AND  #0Ch ;get front & back
CMP  #0Ch ;must be hi else error
BEQ  DiagF2b ;if hi then pass

DiagF2a:
LDA  #1 ;fail beep
JSR  Diag_macro ;go send motor/speech
JMP  DiagF2 ;loop till no error

DiagF2b:
LDA  #2 ;pass beep
JSR  Diag_macro ;go send motor/speech

DiagF2c: ;wait for no tilt before continuing
LDA Port_C
AND #Touch_bck
BEQ DiagF3b

LDA Port_D
AND #Ball_side
BNE DiagF2c

; DANGER
LDA Port_C
AND #Touch_fmt
BEQ DiagF3

LDA Port_D
AND #C3
BEQ DiagF3b

LDA #3
JSR Diag_macro
JMP DiagF3

LDA Port_C
AND #Touch_fmt
BEQ DiagF3a

LDA Port_D
AND #C1
BEQ DiagF3b

LDA #2
JSR Diag_macro

LDA Port_C
AND #Touch_fmt
BEQ DiagF4

LDA Stat_2
ORA #Motor_fwd
ORA #Motor_actv
STA Stat_2
LDA Stat_3
ORA #Motor_off
AND #Motor_fwda
STA Stat_3

LDA Port_C
AND #Touch_fmt
BEQ DiagF4a2

LDA Port_C
AND #Touch_fmt
BEQ DiagF4a1

LDA Port_C
AND #Touch_fmt
BEQ DiagF4a2

; Send motor forward until front switch pressed

; Send motor reverse until front switch pressed
DiagF4a2:
LDA Port_C  ; get I/O wait for front to clear
AND #Touch_fnt ; check switch
BEQ DiagF4a2 ; if pressed then wait for release
LDA Stat_2 ; get system
AND #Motor_rev ; clear fwd flag
ORA #Motor_actv ; set motor in motion
STA Stat_2 ; update system
LDA Stat_3 ; get current status
ORA #Motor_off ; turn both motors off
AND #Motor_rev ; move motor in rev dir
STA Stat_3

DiagF4a3:
LDA Port_C  ; get I/O wait for front
AND #Touch_fnt ; check switch
BEQ DiagF4a3 ; got it
JMP DiagF4a3 ; loop till found

; Send motor end to end and stop on cal sw, else error

DiagF4a4:
LDA Stat_3 ; get current status
ORA #Motor_off ; turn both motors off
STA Stat_3 ; update
LDA Stat_2 ; get system
AND #Motor_inactv ; clear actv flag
STA Stat_2 ; update system
LDA #5 ; start motor test
JSR Diag_macro ; go
LDA #33 ; set delay for motor to stop
JSR Half_delay ; half sec delay
LDA Port_C ; get I/O
AND #Motor_cal ; false when hit
BNE DiagF4b ; no position switch found
LDA #2 ; pass beep
JSR Diag_macro ; go send it
JMP DiagF5 ; done

DiagF4b:
LDA #3 ; fail beep
JSR Diag_macro ; go send it

DiagF5: ; send motor to mouth open for feed sw test
LDA Port_C ; get I/O
AND #Touch_fnt ; wait for switch
BNE DiagF5 ; loop
LDA #6 ; feed position
JSR Diag_macro ; send it

DiagF6:
; check for feed sw, all other sw = error
; Remember to test invert before setting feed sw test, else conflict.
LDA #00 ; clear feed sw enable
STA DAC2
LDA Port_C ; get I/O
AND #0Ch ; check for front and back switches made
CMP #0Ch ; check both are clear
BNE DiagF6a ; wait till are
LDA Port_D ;get I/O
AND #03 ;check for tilt and invert
BNL Diag_F6a ;if either hi then wait till cleer
JMP Diag_F6b ;jump when all cleer

Diag_F6a:
LDA #3 ;fail beep when any other switch made
JSR Diag_macro ;send it
JMP Diag_F6 ;loop

Diag_F6b:
;mod diag6 ; inc random number seeds until feed switch down
INC Seed_1 ;create random based on switches
LDA TMA_LSB ;get timer A also (should be unknown)
STA Seed_2 ;save it

;end mod
LDA #FFh ;turn DAC2 on to enable feed switch
STA DAC2 ;out
LDA Port_D ;get I/O
AND #Ball_invert ;ck if feed switch closed
BEQ Diag_F6 ;loop until switch closed
LDA #00
STA DAC2 ;clear feed sw enable
LDA #7 ;pass beep
JSR Diag_macro ;go send motor/speech

; DiagF7: ;Light sensor test
;mod to compensate for new light sense routine
LDA #00 ;clear light timer to force new reff cycle
STA Light_timer ;set it
LDA Stat_3 ;get system
ORA #Lc_re££ ;make this pass a new light reff
STA Stat_3 ;update
JSR Get_light ;go get light level, establish 1st level
LDA Stat_4
AND #Nd_do_lt_dim ;clear indicating change > reff level
STA Stat_4 ;update system
JSR Get_light ;go get light level sample
LDA TEMP1 ;get new count
STA Light_re££ ;update system

Diag_F7a:
JSR Get_light ;go get again and test for lower level
LDA Stat_4 ;get system
AND #Do_light_dim ;check if went dinner
BEQ Diag_F7a ;loop if no change
LDA #8 ;pass beep and motor motion
JSR Diag_macro ;send it

Diag_F8: ;Sound sensor test
LDA #00 ;clear sound timer to force new reff cycle
STA Sound_timer ;set
LDA Stat_1 ;get system again
ORA #End_re££ ;make this pass a new sound reff
STA Stat_1 ; updates
JSR Get_sound ; go get light level, establish 1st level
LDA Stat_4 ;
AND #Nt_do_snd ; clear indicating change > ref level
STA Stat_4 ; update system

DiagF8a:
JSR Get_sound ; go get again and test for lower level
LDA Stat_4 ; get system
AJJ #Do_snd ; check if went louder
BEQ DiagF8a ; loop if no change
LDA #9 ; pass beep and motor motion
JSR Diag_macro ; send it

DiagF9: ; wait for I.R. data received
LDX #10 ; Tracker change, original is 100

DiagF9al:
LDA #1
JSR Half_delay
DEX
BNE DiagF9al
JSR Diag_IR_test ; go check for data
BCC DiagF9 ; loop until data receive
CMP #A5H ; is it the expected data
BNE DiagF9a ; jump if wrong data
LDA #1 ; pass beep and motor motion
JSR Diag_macro ; send it
JMP DiagF10 ; done

DiagF9a:
LDA #3 ; fail beep and motor motion
JSR Diag_macro ; send it

DiagF10: ; all tests complete, send to sleep mode
LDA #10 ;
JSR Half_delay ;

LDA #10 ; put furby in sleep position
JSR Diag_macro ; send it

; Clear RAM to 00H
; we dont clear Seed_1 or Seed_2 since they are randomized at startup.

;--------------------------------------------------------------------------
LDA #00H ; data for fill
LDX #D7h ; start at ram location

Clear:
STA 00,X ; base 00, offset X
DEX ; next ram location
CPX #7FH ; check for end
BNE Clear ; branch, not finished

; Random voice selection here
LDA #80H ; get random/sequential split
STA IN_DAT ; save for random routine
LDX #00 ; make sure only gives random
LDA #10h ; get number of random selections
JSR Ran_seq ; go get random selection
TAX
LDA Voice_table,X ; get new voice
STA RVoice ; set new voice pitch

; On power up or reset, Furby must go select a new name ..., aww how cute.
JSR Random
AND #1Fh ; get 32 possible
STA Name ; set new name pointer

; Clear training or all sensors
LDA #00
STA Map_ID
STA Temp_ID2
STA Tilt_learned
STA Tilt_lm_cnt
STA Feed_learned
STA Feed_lm_cnt
STA Light_learned
STA Light_lm_cnt
STA Dark_learned
STA Dark_lm_cnt
STA Front_learned
STA Front_lm_cnt
STA Sound_learned
STA Sound_lm_cnt
STA Wake_learned
STA Wake_lm_cnt
STA Invert_learned
STA Invert_lm_cnt
JMP GoToSleep ; write to memory YO
; Furby27.inc ;; change twinkle egg song to one pass in macro

; Lowered voice=10, voice=9 to voice=8
; Wayne's mods:
; Furby5b.inc = add voice selection table
;
; Dave's
; added feed (mouth open)
; 170,171,175,182,183,190,191,194
; mod for ir
; NOW 24 NAMES
;
; TABLES MACRO | SAY
;---------------------
; FRONT 2-64 | 1-61
; FORTUNE 65-83 | 62-78
; o-too-mah 84
; HANGOUT 85-101 | 79-106
; delay 102
; FEED 103-145 | 108-123
; WAKE 146-169 | 124-156
; HUNGER 170-201 | 157-168
; INVERT 202-238 | 169-192
; BACK 239-275 | 193-236
; SICK 276-292 | 237-250
; LIGHT 293-307 | 251-265
; DARK 308-331 | 266-289
; SOUND 332-351 | 290-309
; TILT 352-392 | 310-350
; IR 393-429 | 351-390
; FURBY SAYS 430-434 | 50 TICKLE, 196 PET, 71 SOUND, 391 LIGHT, 198 FURR
; 435-436 | 392 NO LIGHT, 393 LOUD SOUND
; 437-438 | 115,116 ; hide and seek sounds
; 95,96,97 | 98,99,100 ; hide and seek reuse
; 439 | furby says win sound
; Diagnostic 440-450 | 400-410
; 451,452 | 117,118 ; hide and seek sounds
; Names 453 | 399,395,110
; 454 | 399,395,396
; 455 | 399,395,112
; 456 | 399,395,397
; 457 | 399,395,114
; 458 | 399,395,117
; 459 | 399,395,398
; 460 | 399,395,120
; 399 | delay 1.3 seconds
; 461 | 399,395,131
; 462 | 399,395,143
; 463 | 399,395,145
; 464 | 399,395,152
; 465 | 399,395,166
; 466 | 399,395,175
; 467 | 399,395,177
;
; NEW EASTER EGGS
; 468 | DOdle do, me love you
; 469 | SING A SONG
; 470 | FURB ATTACK
; 471 |
; 472 | 46 ; furby says win sound
; 46 | furby says lose sound

A-126
; 473 53,123 ; me done (leaving any game)
; 474 394 ; LISTEN ME
; 475 411 ; HIDE ME (hide and seek)
; 412 ; aah, aah, aah feed dmh

; MORE NAMES
; 476 399,395,186 ; me loo-loo (joke)
; 477 399,395,194 ; me ah-may (pet)
; 478 399,395,201 ; me moo-loo (happy)
; 479 399,395,208 ; me may-may (love)
; 480 399,395,224 ; me may-lah (hug)
; 481 399,3-5,228 ; me dah-noh-lah (big dance)
; 482 399,395,398,152 ; me tch-loo-ka (like me)
; 483 399,395,152,166 ; me ka-da (me big)
; 484 399,395,224,152 ; me may-lah-ka (hug me)

; not used 476-511 | 413-510

; TRAP FOLLOW MACROS FOR NAME
;
; SENSOR
; HANGOUT 97
; WAKE-UP 149
; BACK 248
; LIGHT BRIGHT 305
; IR 393,404,414,421
;
; GAMES
; FORTUNE 69,77
; HIDE AND SEEK 475
; FURBY SAYS 474

; end trap macros for name
;
; reused ; reused ; reused ; reused ; reused
; 72,380 | ; furby says win sounde
; 15 15 ; LAUGH
; 395 ; me (for use with names)
DANCE 407,416 367,376 ; reused for dance easter egg
;
; not used 396-399

Sensor tables
Each sensor has 4 speech/motor tables based on age 1-4, of 16 entries each.
These tables are 16 bit entries, the user enters as a decimal 1-511
*** '00' is illegal ***
This number calls the MACRO tables to get specific speech and motor
tables. MACRO tables chain together multiple motor and speech tables.
The first 8 entries of speech are random selections and
the second 8 entries is sequential.

; one of three voice pitch selections, randomly load table and
table is randomly called on power up to select a new voice.
This gives a number added to voice 3 to create which voice will be
used.

Voice_table:

<table>
<thead>
<tr>
<th>DB</th>
<th>S_voice1</th>
<th>S_voice2</th>
<th>S_voice3</th>
<th>S_voice1</th>
<th>S_voice2</th>
<th>S_voice3</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB</td>
<td>S_voice2</td>
<td>S_voice2</td>
<td>S_voice2</td>
<td>S_voice2</td>
<td>S_voice2</td>
<td>S_voice2</td>
</tr>
<tr>
<td>DB</td>
<td>S_voice3</td>
<td>S_voice3</td>
<td>S_voice3</td>
<td>S_voice3</td>
<td>S_voice3</td>
<td>S_voice3</td>
</tr>
<tr>
<td>DB</td>
<td>S_voice1</td>
<td>S_voice1</td>
<td>S_voice1</td>
<td>S_voice1</td>
<td>S_voice1</td>
<td>S_voice1</td>
</tr>
</tbody>
</table>

; Ball tilt sensor table
; DO TILT

<table>
<thead>
<tr>
<th>Tilt_S1:</th>
<th>DW</th>
<th>352</th>
<th></th>
<th>#1 AGE 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>DW</td>
<td>353</td>
<td></td>
<td>#2 AGE 1</td>
<td></td>
</tr>
<tr>
<td>DW</td>
<td>354</td>
<td></td>
<td>#3 AGE 1</td>
<td></td>
</tr>
<tr>
<td>DW</td>
<td>352</td>
<td></td>
<td>#4 AGE 1</td>
<td></td>
</tr>
<tr>
<td>DW</td>
<td>355</td>
<td></td>
<td>#5 AGE 1</td>
<td></td>
</tr>
<tr>
<td>DW</td>
<td>356</td>
<td></td>
<td>#6 AGE 1</td>
<td></td>
</tr>
<tr>
<td>DW</td>
<td>357</td>
<td></td>
<td>#7 AGE 1</td>
<td></td>
</tr>
<tr>
<td>DW</td>
<td>358</td>
<td></td>
<td>#8 AGE 1</td>
<td></td>
</tr>
<tr>
<td>DW</td>
<td>359</td>
<td></td>
<td>#9 AGE 1</td>
<td></td>
</tr>
<tr>
<td>DW</td>
<td>360</td>
<td></td>
<td>#10 AGE 1</td>
<td></td>
</tr>
<tr>
<td>DW</td>
<td>361</td>
<td></td>
<td>#11 AGE 1</td>
<td></td>
</tr>
<tr>
<td>DW</td>
<td>362</td>
<td></td>
<td>#12 AGE 1</td>
<td></td>
</tr>
<tr>
<td>DW</td>
<td>363</td>
<td></td>
<td>#13 AGE 1</td>
<td></td>
</tr>
<tr>
<td>DW</td>
<td>364</td>
<td></td>
<td>#14 AGE 1</td>
<td></td>
</tr>
<tr>
<td>DW</td>
<td>365</td>
<td></td>
<td>#15 AGE 1</td>
<td></td>
</tr>
<tr>
<td>DW</td>
<td>366</td>
<td></td>
<td>#16 AGE 1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tilt_S2:</th>
<th>DW</th>
<th>366</th>
<th></th>
<th>#1 AGE 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>DW</td>
<td>367</td>
<td></td>
<td>#2 AGE 2</td>
<td></td>
</tr>
<tr>
<td>DW</td>
<td>366</td>
<td></td>
<td>#3 AGE 2</td>
<td></td>
</tr>
<tr>
<td>DW</td>
<td>355</td>
<td></td>
<td>#4 AGE 2</td>
<td></td>
</tr>
<tr>
<td>DW</td>
<td>358</td>
<td></td>
<td>#5 AGE 2</td>
<td></td>
</tr>
<tr>
<td>DW</td>
<td>357</td>
<td></td>
<td>#6 AGE 2</td>
<td></td>
</tr>
<tr>
<td>DW</td>
<td>359</td>
<td></td>
<td>#7 AGE 2</td>
<td></td>
</tr>
<tr>
<td>DW</td>
<td>360</td>
<td></td>
<td>#8 AGE 2</td>
<td></td>
</tr>
<tr>
<td>DW</td>
<td>361</td>
<td></td>
<td>#9 AGE 2</td>
<td></td>
</tr>
<tr>
<td>DW</td>
<td>360</td>
<td></td>
<td>#10 AGE 2</td>
<td></td>
</tr>
<tr>
<td>DW</td>
<td>371</td>
<td></td>
<td>#11 AGE 2</td>
<td></td>
</tr>
<tr>
<td>DW</td>
<td>372</td>
<td></td>
<td>#12 AGE 2</td>
<td></td>
</tr>
<tr>
<td>DW</td>
<td>373</td>
<td></td>
<td>#13 AGE 2</td>
<td></td>
</tr>
<tr>
<td>DW</td>
<td>374</td>
<td></td>
<td>#14 AGE 2</td>
<td></td>
</tr>
<tr>
<td>DW</td>
<td>355</td>
<td></td>
<td>#15 AGE 2</td>
<td></td>
</tr>
<tr>
<td>DW</td>
<td>376</td>
<td></td>
<td>#16 AGE 2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tilt_S3:</th>
<th>DW</th>
<th>366</th>
<th></th>
<th>#1 AGE 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>DW</td>
<td>355</td>
<td></td>
<td>#2 AGE 3</td>
<td></td>
</tr>
<tr>
<td>DW</td>
<td>NT</td>
<td>AGE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>-----</td>
<td></td>
<td></td>
</tr>
<tr>
<td>376</td>
<td>DW</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>377</td>
<td>DW</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>378</td>
<td>DW</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>379</td>
<td>DW</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>380</td>
<td>DW</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>381</td>
<td>DW</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>382</td>
<td>DW</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>383</td>
<td>DW</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>384</td>
<td>DW</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>385</td>
<td>DW</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>386</td>
<td>DW</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>387</td>
<td>DW</td>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>388</td>
<td>DW</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>389</td>
<td>DW</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>390</td>
<td>DW</td>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>391</td>
<td>DW</td>
<td>18</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Tilt_S4:

<table>
<thead>
<tr>
<th>DW</th>
<th>NT</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>366</td>
<td>DW</td>
<td>1</td>
</tr>
<tr>
<td>355</td>
<td>DW</td>
<td>2</td>
</tr>
<tr>
<td>387</td>
<td>DW</td>
<td>3</td>
</tr>
<tr>
<td>377</td>
<td>DW</td>
<td>4</td>
</tr>
<tr>
<td>388</td>
<td>DW</td>
<td>5</td>
</tr>
<tr>
<td>389</td>
<td>DW</td>
<td>6</td>
</tr>
<tr>
<td>380</td>
<td>DW</td>
<td>7</td>
</tr>
<tr>
<td>381</td>
<td>DW</td>
<td>8</td>
</tr>
<tr>
<td>382</td>
<td>DW</td>
<td>9</td>
</tr>
<tr>
<td>383</td>
<td>DW</td>
<td>10</td>
</tr>
<tr>
<td>390</td>
<td>DW</td>
<td>11</td>
</tr>
<tr>
<td>385</td>
<td>DW</td>
<td>12</td>
</tr>
<tr>
<td>391</td>
<td>DW</td>
<td>13</td>
</tr>
<tr>
<td>375</td>
<td>DW</td>
<td>14</td>
</tr>
<tr>
<td>384</td>
<td>DW</td>
<td>15</td>
</tr>
<tr>
<td>392</td>
<td>DW</td>
<td>16</td>
</tr>
</tbody>
</table>

Sick_S1:

<table>
<thead>
<tr>
<th>DW</th>
<th>NT</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>276</td>
<td>DW</td>
<td>1</td>
</tr>
<tr>
<td>280</td>
<td>DW</td>
<td>2</td>
</tr>
<tr>
<td>283</td>
<td>DW</td>
<td>3</td>
</tr>
<tr>
<td>286</td>
<td>DW</td>
<td>4</td>
</tr>
<tr>
<td>288</td>
<td>DW</td>
<td>5</td>
</tr>
<tr>
<td>289</td>
<td>DW</td>
<td>6</td>
</tr>
<tr>
<td>290</td>
<td>DW</td>
<td>7</td>
</tr>
<tr>
<td>291</td>
<td>DW</td>
<td>8</td>
</tr>
<tr>
<td>292</td>
<td>DW</td>
<td>9</td>
</tr>
<tr>
<td>288</td>
<td>DW</td>
<td>10</td>
</tr>
<tr>
<td>288</td>
<td>DW</td>
<td>11</td>
</tr>
<tr>
<td>288</td>
<td>DW</td>
<td>12</td>
</tr>
<tr>
<td>289</td>
<td>DW</td>
<td>13</td>
</tr>
<tr>
<td>290</td>
<td>DW</td>
<td>14</td>
</tr>
<tr>
<td>291</td>
<td>DW</td>
<td>15</td>
</tr>
<tr>
<td>292</td>
<td>DW</td>
<td>16</td>
</tr>
</tbody>
</table>

Sick_S2:

<table>
<thead>
<tr>
<th>DW</th>
<th>NT</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>277</td>
<td>DW</td>
<td>1</td>
</tr>
<tr>
<td>280</td>
<td>DW</td>
<td>2</td>
</tr>
<tr>
<td>284</td>
<td>DW</td>
<td>3</td>
</tr>
<tr>
<td>286</td>
<td>DW</td>
<td>4</td>
</tr>
<tr>
<td>288</td>
<td>DW</td>
<td>5</td>
</tr>
<tr>
<td>288</td>
<td>DW</td>
<td>6</td>
</tr>
<tr>
<td>289</td>
<td>DW</td>
<td>7</td>
</tr>
<tr>
<td>290</td>
<td>DW</td>
<td>8</td>
</tr>
<tr>
<td>DW 291</td>
<td>; #9 AGE 2</td>
<td></td>
</tr>
<tr>
<td>DW 292</td>
<td>; #10 AGE 2</td>
<td></td>
</tr>
<tr>
<td>DW 288</td>
<td>; #11 AGE 2</td>
<td></td>
</tr>
<tr>
<td>DW 288</td>
<td>; #12 AGE 2</td>
<td></td>
</tr>
<tr>
<td>DW 290</td>
<td>; #13 AGE 2</td>
<td></td>
</tr>
<tr>
<td>DW 291</td>
<td>; #14 AGE 2</td>
<td></td>
</tr>
<tr>
<td>DW 292</td>
<td>; #15 AGE 2</td>
<td></td>
</tr>
<tr>
<td>DW 292</td>
<td>; #16 AGE 2</td>
<td></td>
</tr>
</tbody>
</table>

**Sick_S3:**

| DW 276 | ; #1 AGE 3 |
| DW 281 | ; #2 AGE 3 |
| DW 285 | ; #3 AGE 3 |
| DW 287 | ; #4 AGE 3 |
| DW 288 | ; #5 AGE 3 |
| DW 288 | ; #6 AGE 3 |
| DW 289 | ; #7 AGE 3 |
| DW 290 | ; #8 AGE 3 |
| DW 291 | ; #9 AGE 3 |
| DW 292 | ; #10 AGE 3 |
| DW 288 | ; #11 AGE 3 |
| DW 288 | ; #12 AGE 3 |
| DW 289 | ; #13 AGE 3 |
| DW 290 | ; #14 AGE 3 |
| DW 291 | ; #15 AGE 3 |
| DW 292 | ; #16 AGE 3 |

**Sick_S4:**

| DW 279 | ; #1 AGE 4 |
| DW 282 | ; #2 AGE 4 |
| DW 285 | ; #3 AGE 4 |
| DW 287 | ; #4 AGE 4 |
| DW 288 | ; #5 AGE 4 |
| DW 288 | ; #6 AGE 4 |
| DW 289 | ; #7 AGE 4 |
| DW 290 | ; #8 AGE 4 |
| DW 291 | ; #9 AGE 4 |
| DW 292 | ; #10 AGE 4 |
| DW 288 | ; #11 AGE 4 |
| DW 288 | ; #12 AGE 4 |
| DW 289 | ; #13 AGE 4 |
| DW 290 | ; #14 AGE 4 |
| DW 291 | ; #15 AGE 4 |
| DW 292 | ; #16 AGE 4 |

; SWITCH FOR DO SEND js

**Sound_S1:**

| DW 332 | ; #1 AGE 1 |
| DW 333 | ; #2 AGE 1 |
| DW 334 | ; #3 AGE 1 |
| DW 335 | ; #4 AGE 1 |
| DW 336 | ; #5 AGE 1 |
| DW 337 | ; #6 AGE 1 |
| DW 338 | ; #7 AGE 1 |
| DW 339 | ; #8 AGE 1 |
| DW 332 | ; #9 AGE 1 |
| LW 333 | ; #10 AGE 1 |
| DW 334 | ; #11 AGE 1 |
Sound_S2: DW 332
  ; 01  AGE 2
  DW 333
  ; 02  AGE 2
  DW 340
  ; 03  AGE 2
  DW 341
  ; 04  AGE 2
  DW 342
  ; 05  AGE 2
  DW 337
  ; 06  AGE 2
  DW 343
  ; 07  AGE 2
  DW 344
  ; 08  AGE 2
  DW 332
  ; 09  AGE 2
  DW 333
  ; 10  AGE 2
  DW 340
  ; 11  AGE 2
  DW 341
  ; 12  AGE 2
  DW 337
  ; 13  AGE 2
  DW 343
  ; 14  AGE 2
  DW 344
  ; 15  AGE 2
  DW 342
  ; 16  AGE 2

Sound_S3: DW 332
  ; 01  AGE 3
  DW 333
  ; 02  AGE 3
  DW 345
  ; 03  AGE 3
  DW 346
  ; 04  AGE 3
  DW 342
  ; 05  AGE 3
  DW 337
  ; 06  AGE 3
  DW 347
  ; 07  AGE 3
  DW 339
  ; 08  AGE 3
  DW 332
  ; 09  AGE 3
  DW 333
  ; 10  AGE 3
  DW 345
  ; 11  AGE 3
  DW 346
  ; 12  AGE 3
  DW 342
  ; 13  AGE 3
  DW 337
  ; 14  AGE 3
  DW 347
  ; 15  AGE 3
  DW 339
  ; 16  AGE 3

Sound_S4: DW 348
  ; 01  AGE 4
  DW 333
  ; 02  AGE 4
  DW 349
  ; 03  AGE 4
  DW 346
  ; 04  AGE 4
  DW 342
  ; 05  AGE 4
  DW 350
  ; 06  AGE 4
  DW 347
  ; 07  AGE 4
  DW 351
  ; 08  AGE 4
  DW 348
  ; 09  AGE 4
  DW 333
  ; 10  AGE 4
  DW 349
  ; 11  AGE 4
  DW 346
  ; 12  AGE 4
  DW 342
  ; 13  AGE 4
  DW 350
  ; 14  AGE 4
  DW 347
  ; 15  AGE 4
  DW 351
  ; 16  AGE 4

*******************************************************************************
```plaintext
: DO HUNGER
:
Hunger_S1:

<table>
<thead>
<tr>
<th>DW</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>170</td>
<td>#1 AGE 1</td>
</tr>
<tr>
<td>171</td>
<td>#2 AGE 1</td>
</tr>
<tr>
<td>173</td>
<td>#3 AGE 1</td>
</tr>
<tr>
<td>175</td>
<td>#4 AGE 1</td>
</tr>
<tr>
<td>180</td>
<td>#5 AGE 1</td>
</tr>
<tr>
<td>182</td>
<td>#6 AGE 1</td>
</tr>
<tr>
<td>193</td>
<td>#7 AGE 1</td>
</tr>
<tr>
<td>185</td>
<td>#8 AGE 1</td>
</tr>
<tr>
<td>189</td>
<td>#9 AGE 1</td>
</tr>
<tr>
<td>194</td>
<td>#10 AGE 1</td>
</tr>
<tr>
<td>173</td>
<td>#11 AGE 1</td>
</tr>
<tr>
<td>195</td>
<td>#12 AGE 1</td>
</tr>
<tr>
<td>189</td>
<td>#13 AGE 1</td>
</tr>
<tr>
<td>193</td>
<td>#14 AGE 1</td>
</tr>
<tr>
<td>194</td>
<td>#15 ACF 1</td>
</tr>
<tr>
<td>199</td>
<td>#16 AGE 1</td>
</tr>
</tbody>
</table>

Hunger_S2:

<table>
<thead>
<tr>
<th>DW</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>171</td>
<td>#1 AGE 2</td>
</tr>
<tr>
<td>174</td>
<td>#2 AGE 2</td>
</tr>
<tr>
<td>177</td>
<td>#3 AGE 2</td>
</tr>
<tr>
<td>181</td>
<td>#4 AGE 2</td>
</tr>
<tr>
<td>183</td>
<td>#5 AGE 2</td>
</tr>
<tr>
<td>174</td>
<td>#6 AGE 2</td>
</tr>
<tr>
<td>196</td>
<td>#7 AGE 2</td>
</tr>
<tr>
<td>190</td>
<td>#8 AGE 2</td>
</tr>
<tr>
<td>193</td>
<td>#9 AGE 2</td>
</tr>
<tr>
<td>194</td>
<td>#10 AGE 2</td>
</tr>
<tr>
<td>174</td>
<td>#11 AGE 2</td>
</tr>
<tr>
<td>196</td>
<td>#12 AGE 2</td>
</tr>
<tr>
<td>190</td>
<td>#13 AGE 2</td>
</tr>
<tr>
<td>193</td>
<td>#14 AGE 2</td>
</tr>
<tr>
<td>194</td>
<td>#15 AGE 2</td>
</tr>
<tr>
<td>200</td>
<td>#16 AGE 2</td>
</tr>
</tbody>
</table>

Hunger_S3:

<table>
<thead>
<tr>
<th>DW</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>172</td>
<td>#1 AGE 3</td>
</tr>
<tr>
<td>174</td>
<td>#2 AGE 3</td>
</tr>
<tr>
<td>178</td>
<td>#3 AGE 3</td>
</tr>
<tr>
<td>181</td>
<td>#4 AGE 3</td>
</tr>
<tr>
<td>184</td>
<td>#5 AGE 3</td>
</tr>
<tr>
<td>185</td>
<td>#6 AGE 3</td>
</tr>
<tr>
<td>191</td>
<td>#7 AGE 3</td>
</tr>
<tr>
<td>193</td>
<td>#8 AGE 3</td>
</tr>
<tr>
<td>173</td>
<td>#9 AGE 3</td>
</tr>
<tr>
<td>175</td>
<td>#10 AGE 3</td>
</tr>
<tr>
<td>175</td>
<td>#11 ACP 3</td>
</tr>
<tr>
<td>197</td>
<td>#12 AGE 3</td>
</tr>
<tr>
<td>191</td>
<td>#13 AGE 3</td>
</tr>
<tr>
<td>193</td>
<td>#14 AGE 3</td>
</tr>
<tr>
<td>173</td>
<td>#15 AGE 3</td>
</tr>
<tr>
<td>200</td>
<td>#16 AGE 3</td>
</tr>
</tbody>
</table>

Hunger_S4:

<table>
<thead>
<tr>
<th>DW</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>171</td>
<td>#1 AGE 4</td>
</tr>
<tr>
<td>175</td>
<td>#2 AGE 4</td>
</tr>
</tbody>
</table>
```

MACRO 65-83, SAY 62-78

Fortune teller game

;GEORGE 07/04/98

Fortyes_S1:

DW 179 ; #3 AGE 4
DW 181 ; #4 AGE 4
DW 184 ; #5 AGE 4
DW 175 ; #6 AGE 4
DW 188 ; #7 AGE 4
DW 192 ; #8 AGE 4
DW 194 ; #9 AGE 4
DW 193 ; #10 AGE 4
DW 174 ; #11 AGE 4
DW 198 ; #12 AGE 4
DW 192 ; #13 AGE 4
DW 193 ; #14 AGE 4
DW 194 ; #15 AGE 4
DW 201 ; #16 AGE 4

MACRO 65-83, SAY 62-78

Fortyes_S2:

DW 065 ; #1 AGE 1
DW 066 ; #2 AGE 1
DW 067 ; #3 AGE 1
DW 068 ; #4 AGE 1
DW 069 ; #5 AGE 1
DW 070 ; #6 AGE 1
DW 071 ; #7 AGE 1
DW 072 ; #8 AGE 1
DW 073 ; #9 AGE 1
DW 074 ; #10 AGE 1
DW 075 ; #11 AGE 1
DW 076 ; #12 AGE 1
DW 077 ; #13 AGE 1
DW 078 ; #14 AGE 1
DW 079 ; #15 AGE 1
DW 080 ; #16 AGE 1

Fortyes_S3:

DW 081 ; #1 AGE 2
DW 082 ; #2 AGE 2
DW 083 ; #3 AGE 2
DW 065 ; #4 AGE 2
DW 066 ; #5 AGE 2
DW 067 ; #6 AGE 2
DW 068 ; #7 AGE 2
DW 069 ; #8 AGE 2
DW 070 ; #9 AGE 2
DW 071 ; #10 AGE 2
DW 072 ; #11 AGE 2
DW 073 ; #12 AGE 2
DW 074 ; #13 AGE 2
DW 075 ; #14 AGE 2
DW 076 ; #15 AGE 2
DW 077 ; #16 AGE 2

;END FORTUNE
;END GEORGE 07/04/98
;touch front sensor table
;GEORGE 07/03/98 MACRO 2-64, SAY 1-61

Tfnt_S1: DW 002 ; #1 AGE 1
DW 003 ; #2 AGE 1
DW 004 ; #3 AGE 1
DW 005 ; #4 AGE 1
DW 006 ; #5 AGE 1
DW 007 ; #6 AGE 1
DW 008 ; #7 AGE 1
DW 009 ; #8 AGE 1
DW 010 ; #9 AGE 1
DW 011 ; #10 AGE 1
DW 012 ; #11 AGE 1
DW 013 ; #12 AGE 1
DW 014 ; #13 AGE 1
DW 015 ; #14 AGE 1
DW 016 ; #15 AGE 1
DW 017 ; #16 AGE 1

Tfnt_S2: DW 018 ; #1 AGE 2
DW 019 ; #2 AGE 2
DW 020 ; #3 AGE 2
DW 021 ; #4 AGE 2
DW 022 ; #5 AGE 2
DW 023 ; #6 AGE 2
DW 024 ; #7 AGE 2
DW 025 ; #8 AGE 2
DW 026 ; #9 AGE 2
DW 027 ; #10 AGE 2
DW 028 ; #11 AGE 2
DW 029 ; #12 AGE 2
DW 030 ; #13 AGE 2
DW 031 ; #14 AGE 2
DW 032 ; #15 AGE 2
DW 033 ; #16 AGE 2

Tfnt_S3: DW 034 ; #1 AGE 3
DW 035 ; #2 AGE 3
DW 036 ; #3 AGE 3
DW 037 ; #4 AGE 3
DW 038 ; #5 AGE 3
DW 039 ; #6 AGE 3
DW 040 ; #7 AGE 3
DW 041 ; #25 ; #8 AGE 3
DW 042 ; #9 AGE 3
DW 043 ; #10 AGE 3
DW 044 ; #11 AGE 3
DW 045 ; #12 AGE 3
DW 046 ; #13 AGE 3
DW 047 ; #14 AGE 3
DW 048 ; #15 AGE 3
DW 049 ; #16 AGE 3

Tfnt_S4: DW 049 ; #1 AGE 4
DW 050 ; #2 AGE 4
DW 051 ; #3 AGE 4
DW 052 ; #4 AGE 4
DW 053 ; #5 AGE 4
DW 054 ; #6 AGE 4
DW 055 ; #7 AGE 4
| DW   | 056 | : #8 AGE 4 |
| DW   | 057 | : #9 AGE 4 |
| DW   | 058 | : #10 AGE 4 |
| DW   | 059 | : #11 AGE 4 |
| DW   | 060 | : #12 AGE 4 |
| DW   | 061 | : #13 AGE 4 |
| DW   | 062 | : #14 AGE 4 |
| DW   | 063 | : #15 AGE 4 |
| DW   | 064 | : #16 AGE 4 |

;END GEORGE 07/03/98

; feed sense table
; DO FEED (Do LINVERT)
;GEORGE 07/05/98

Feed_S1:

| DW   | 117 | : #1 AGE 1 |
| DW   | 103 | : #2 AGE 1 |
| DW   | 104 | : #3 AGE 1 |
| DW   | 105 | : #4 AGE 1 |
| DW   | 106 | : #5 AGE 1 |
| DW   | 107 | : #6 AGE 1 |
| DW   | 108 | : #7 AGE 1 |
| DW   | 109 | : #8 AGE 1 |
| DW   | 110 | : #9 AGE 1 |
| DW   | 111 | : #10 AGE 1 |
| DW   | 112 | : #11 AGE 1 |
| DW   | 113 | : #12 AGE 1 |
| DW   | 114 | : #13 AGE 1 |
| DW   | 115 | : #14 AGE 1 |
| DW   | 116 | : #15 AGE 1 |

Feed_S2:

| DW   | 118 | : #1 AGE 2 |
| DW   | 119 | : #2 AGE 2 |
| DW   | 120 | : #3 AGE 2 |
| DW   | 121 | : #4 AGE 2 |
| DW   | 122 | : #5 AGE 2 |
| DW   | 123 | : #6 AGE 2 |
| DW   | 124 | : #7 AGE 2 |
| DW   | 125 | : #8 AGE 2 |
| DW   | 126 | : #9 AGE 2 |
| DW   | 127 | : #10 AGE 2 |
| DW   | 128 | : #11 AGE 2 |
| DW   | 113 | : #12 AGE 2 |
| DW   | 114 | : #13 AGE 2 |
| DW   | 111 | : #14 AGE 2 |
| DW   | 129 | : #15 AGE 2 |
| DW   | 116 | : #16 AGE 2 |

Feed_S3:

| DW   | 118 | : #1 AGE 3 |
| DW   | 130 | : #2 AGE 3 |
| DW   | 131 | : #3 AGE 3 |
| DW   | 132 | : #4 AGE 3 |
| DW   | 122 | : #5 AGE 3 |
Feed_S4:

DW 107 ; #6 AGE 3
DW 133 ; #7 AGE 3
DW 134 ; #8 AGE 3
DW 110 ; #9 AGE 3
DW 111 ; #10 AGE 3
DW 113 ; #11 AGE 3
DW 114 ; #12 AGE 3
DW 111 ; #14 AGE 3
DW 135 ; #15 AGE 3
DW 116 ; #16 AGE 3

Wakeup_S1:

DW 145 ; #1 AGE 4
DW 136 ; #2 AGE 4
DW 137 ; #3 AGE 4
DW 138 ; #4 AGE 4
DW 139 ; #5 AGE 4
DW 140 ; #6 AGE 4
DW 141 ; #7 AGE 4
DW 142 ; #8 AGE 4
DW 110 ; #9 AGE 4
DW 111 ; #10 AGE 4
DW 143 ; #11 AGE 4
DW 113 ; #12 AGE 4
DW 114 ; #13 AGE 4
DW 111 ; #14 AGE 4
DW 144 ; #15 AGE 4
DW 116 ; #16 AGE 4

; END GEORGE 07/05/98
; touch front sensor table
; DO WAKES ; DONE SG

Wakeup_S2: DW 147 ; #1 AGE 2

DW 149 ; #2 AGE 2
DW 151 ; #3 AGE 2
DW 155 ; #4 AGE 2
DW 158 ; #5 AGE 2
DW 160 ; #6 AGE 2
DW 163 ; #7 AGE 2
DW 167 ; #8 AGE 2
DW 147 ; #9 AGE 2
DW 149 ; #10 AGE 2
Ball tilt sensor table:
; DO TILT (HANGING OUT)
; START HANGOUT MACRC &5-101, SAY 79-106
;GEORGE 07/04/98

DO HANGOUT
DO BORED

Bored SL:
;bored time out

085 ; #1 AGE 1
086 ; #2 AGE 1
087 ; #3 AGE 1
088 ; #4 AGE 1
089 ; #5 AGE 1
090 ; #6 AGE 1
091 ; #7 AGE 1; sleep
092 ; #8 AGE 1
093 ; #9 AGE 1; dobedo
094 ; #10 AGE 1; yawn
<table>
<thead>
<tr>
<th>Source:</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invrt_S1: DW</td>
<td>202</td>
<td>#1 AGE 1</td>
</tr>
<tr>
<td>DW</td>
<td>203</td>
<td>#2 AGE 1</td>
</tr>
<tr>
<td>DW</td>
<td>206</td>
<td>#3 AGE 1</td>
</tr>
<tr>
<td>DW</td>
<td>208</td>
<td>#4 AGE 1</td>
</tr>
<tr>
<td>DW</td>
<td>212</td>
<td>#5 AGE 1</td>
</tr>
<tr>
<td>DW</td>
<td>213</td>
<td>#6 AGE 1</td>
</tr>
<tr>
<td>DW</td>
<td>217</td>
<td>#7 AGE 1</td>
</tr>
<tr>
<td>DW</td>
<td>219</td>
<td>#8 AGE 1</td>
</tr>
<tr>
<td>DW</td>
<td>220</td>
<td>#9 AGE 1</td>
</tr>
<tr>
<td>DW</td>
<td>224</td>
<td>#10 AGE 1</td>
</tr>
<tr>
<td>DW</td>
<td>228</td>
<td>#11 AGE 1</td>
</tr>
<tr>
<td>DW</td>
<td>232</td>
<td>#12 AGE 1</td>
</tr>
<tr>
<td>DW</td>
<td>234</td>
<td>#13 AGE 1</td>
</tr>
<tr>
<td>DW</td>
<td>232</td>
<td>#14 AGE 1</td>
</tr>
<tr>
<td>DW</td>
<td>234</td>
<td>#15 AGE 1</td>
</tr>
<tr>
<td>DW</td>
<td>235</td>
<td>#16 AGE 1</td>
</tr>
<tr>
<td>Invrt_S2: DW</td>
<td>202</td>
<td>#1 AGE 2</td>
</tr>
<tr>
<td>DW</td>
<td>203</td>
<td>#2 AGE 2</td>
</tr>
<tr>
<td>DW</td>
<td>207</td>
<td>#3 AGE 2</td>
</tr>
<tr>
<td>DW</td>
<td>209</td>
<td>#4 AGE 2</td>
</tr>
<tr>
<td>DW</td>
<td>212</td>
<td>#5 AGE 2</td>
</tr>
<tr>
<td>DW</td>
<td>214</td>
<td>#6 AGE 2</td>
</tr>
<tr>
<td>DW</td>
<td>217</td>
<td>#7 AGE 2</td>
</tr>
<tr>
<td>DW</td>
<td>219</td>
<td>#8 AGE 2</td>
</tr>
<tr>
<td>DW</td>
<td>221</td>
<td>#9 AGE 2</td>
</tr>
<tr>
<td>DW</td>
<td>225</td>
<td>#10 AGE 2</td>
</tr>
<tr>
<td>DW</td>
<td>229</td>
<td>#11 AGE 2</td>
</tr>
<tr>
<td>DW</td>
<td>232</td>
<td>#12 AGE 2</td>
</tr>
<tr>
<td>DW</td>
<td>234</td>
<td>#13 AGE 2</td>
</tr>
<tr>
<td>DW</td>
<td>232</td>
<td>#14 AGE 2</td>
</tr>
<tr>
<td>DW</td>
<td>234</td>
<td>#15 AGE 2</td>
</tr>
<tr>
<td>DW</td>
<td>236</td>
<td>#16 AGE 2</td>
</tr>
<tr>
<td>Invrt_S3: DW</td>
<td>202</td>
<td>#1 AGE 3</td>
</tr>
<tr>
<td>DW</td>
<td>204</td>
<td>#2 AGE 3</td>
</tr>
<tr>
<td>DW</td>
<td>207</td>
<td>#3 AGE 3</td>
</tr>
<tr>
<td>DW</td>
<td>210</td>
<td>#4 AGE 3</td>
</tr>
<tr>
<td>DW</td>
<td>212</td>
<td>#5 AGE 3</td>
</tr>
<tr>
<td>DW</td>
<td>215</td>
<td>#6 AGE 3</td>
</tr>
<tr>
<td>DW</td>
<td>218</td>
<td>#7 AGE 3</td>
</tr>
<tr>
<td>DW</td>
<td>219</td>
<td>#8 AGE 3</td>
</tr>
<tr>
<td>DW</td>
<td>222</td>
<td>#9 AGE 3</td>
</tr>
<tr>
<td>DW</td>
<td>226</td>
<td>#10 AGE 3</td>
</tr>
<tr>
<td>DW</td>
<td>230</td>
<td>#11 AGE 3</td>
</tr>
<tr>
<td>DW</td>
<td>232</td>
<td>#12 AGE 3</td>
</tr>
<tr>
<td>DW</td>
<td>234</td>
<td>#13 AGE 3</td>
</tr>
<tr>
<td>DW</td>
<td>232</td>
<td>#14 AGE 3</td>
</tr>
<tr>
<td>DW</td>
<td>234</td>
<td>#15 AGE 3</td>
</tr>
<tr>
<td>DW</td>
<td>237</td>
<td>#16 AGE 3</td>
</tr>
<tr>
<td>Invrt_S4: DW</td>
<td>202</td>
<td>#1 AGE 4</td>
</tr>
<tr>
<td>DW</td>
<td>205</td>
<td>#2 AGE 4</td>
</tr>
<tr>
<td>DW</td>
<td>207</td>
<td>#3 AGE 4</td>
</tr>
<tr>
<td>DW</td>
<td>211</td>
<td>#4 AGE 4</td>
</tr>
</tbody>
</table>
;GEORGE 07/07/98
;BACK
;touch back sensor table
;
Tback_S1: DW 239 ; #1 AGE 1
DW 240 ; #2 AGE 1
DW 244 ; #3 AGE 1
DW 248 ; #4 AGE 1
DW 249 ; #5 AGE 1
DW 248 ; #6 AGE 1
DW 253 ; #7 AGE 1
DW 256 ; #8 AGE 1
DW 258 ; #9 AGE 1
DW 239 ; #10 AGE 1
DW 248 ; #11 AGE 1
DW 261 ; #12 AGE 1
DW 263 ; #13 AGE 1
DW 266 ; #14 AGE 1
DW 269 ; #15 AGE 1
DW 272 ; #16 AGE 1

Tback_S2: DW 239 ; #1 AGE 2
DW 241 ; #2 AGE 2
DW 245 ; #3 AGE 2
DW 248 ; #4 AGE 2
DW 250 ; #5 AGE 2
DW 248 ; #6 AGE 2
DW 253 ; #7 AGE 2
DW 257 ; #8 AGE 2
DW 259 ; #9 AGE 2
DW 239 ; #10 AGE 2
DW 248 ; #11 AGE 2
DW 262 ; #12 AGE 2
DW 264 ; #13 AGE 2
DW 267 ; #14 AGE 2
DW 270 ; #15 AGE 2
DW 273 ; #16 AGE 2

Tback_S3: DW 239 ; #1 AGE 3
DW 242 ; #2 AGE 3
DW 246 ; #3 AGE 3
DW 248 ; #4 AGE 3
DW 251 ; #5 AGE 3
DW 248 ; #6 AGE 3
DW 254 ; #7 AGE 3
DW 257 ; #8 AGE 3
DW 260 ; #9 AGE 3
| DW  | 239 | ; #10 AGE 3 |
| DW  | 248 | ; #11 AGE 3 |
| DW  | 261 | ; #12 AGE 3 |
| DW  | 265 | ; #13 AGE 3 |
| DW  | 268 | ; #14 AGE 3 |
| DW  | 271 | ; #15 AGE 3 |
| DW  | 274 | ; #16 AGE 3 |

| Tback_S4: DW  | 239 | ; #1 AGE 4 |
| DW  | 243 | ; #2 AGE 4 |
| DW  | 247 | ; #3 AGE 4 |
| DW  | 248 | ; #4 AGE 4 |
| DW  | 252 | ; #5 AGE 4 |
| DW  | 248 | ; #6 AGE 4 |
| DW  | 255 | ; #7 AGE 4 |
| DW  | 257 | ; #8 AGE 4 |
| DW  | 260 | ; #9 AGE 4 |
| DW  | 239 | ; #10 AGE 4 |
| DW  | 248 | ; #11 AGE 4 |
| DW  | 262 | ; #12 AGE 4 |
| DW  | 265 | ; #13 AGE 4 |
| DW  | 268 | ; #14 AGE 4 |
| DW  | 271 | ; #15 AGE 4 |
| DW  | 275 | ; #16 AGE 4 |

;END GEORGE 07/07/98

; I.R. receive table
; DO IR

| IR_S1: DW  | 393 | ; #1 AGE 1 |
| DW  | 393 | ; #2 AGE 1 |
| DW  | 393 | ; #3 AGE 1 |
| DW  | 394 | ; #4 AGE 1 |
| DW  | 395 | ; #5 AGE 1 |
| DW  | 396 | ; #6 AGE 1 |
| DW  | 396 | ; #7 AGE 1 |
| DW  | 291 | ; #8 AGE 1 |
| DW  | 399 | ; #9 AGE 1 |
| DW  | 399 | ; #10 AGE 1 |
| DW  | 400 | ; #11 AGE 1 |
| DW  | 401 | ; #12 AGE 1 |
| DW  | 401 | ; #13 AGE 1 |
| DW  | 402 | ; #14 AGE 1 |
| DW  | 403 | ; #15 AGE 1 |
| DW  | 404 | ; #16 AGE 1 |

| IR_S2: DW  | 404 | ; #1 AGE 2 |
| DW  | 404 | ; #2 AGE 2 |
| DW  | 404 | ; #3 AGE 2 |
| DW  | 405 | ; #4 AGE 2 |
| DW  | 405 | ; #5 AGE 2 |
| DW  | 406 | ; #6 AGE 2 |
| DW  | 407 | ; #7 AGE 2 |
| DW  | 407 | ; #8 AGE 2 |
| DW  | 291 | ; #9 AGE 2 |
| DW  | 409 | ; #10 AGE 2 |
| DW  | 409 | ; #11 AGE 2 |
; light sense table (bright sense)
; DO LIGHT
Light_S1:

DW 293  ; #1 AGE 1
DW 305  ; 003  ; #2 AGE 1
DW 294  ; #3 AGE 1
DW 295  ; #4 AGE 1
DW 296  ; #5 AGE 1
DW 297  ; #6 AGE 1
DW 298  ; #7 AGE 1
DW 299  ; #8 AGE 1
DW 293  ; #9 AGE 1
DW 305  ; 003  ; #10 AGE 1
DW 294  ; #11 AGE 1
DW 295  ; #12 AGE 1
DW 296  ; #13 AGE 1
DW 297  ; #14 AGE 1
| DW 298 | ; #15 AGE 1 |
| DW 299 | ; #16 AGE 1 |

**Light_S2:**

| DW 293 | ; #1 AGE 2 |
| DW 305 | ; 003 ; #2 AGE 2 |
| DW 294 | ; #3 AGE 2 |
| DW 300 | ; #4 AGE 2 |
| DW 296 | ; #5 AGE 2 |
| DW 301 | ; #6 AGE 2 |
| DW 298 | ; #7 AGE 2 |
| DW 299 | ; #8 AGE 2 |
| DW 293 | ; #9 AGE 2 |
| DW 305 | ; 003 ; #10 AGE 2 |
| DW 294 | ; #11 AGE 2 |
| DW 295 | ; #12 AGE 2 |
| DW 296 | ; #13 AGE 2 |
| DW 301 | ; #14 AGE 2 |
| DW 298 | ; #15 AGE 2 |
| DW 299 | ; #16 AGE 2 |

**Light_S3:**

| DW 302 | ; #1 AGE 3 |
| DW 305 | ; 003 ; #2 AGE 3 |
| DW 294 | ; #3 AGE 3 |
| DW 303 | ; #4 AGE 3 |
| DW 296 | ; #5 AGE 3 |
| DW 304 | ; #6 AGE 3 |
| DW 298 | ; #7 AGE 3 |
| DW 299 | ; #8 AGE 3 |
| DW 302 | ; #9 AGE 3 |
| DW 305 | ; 003 ; #10 AGE 3 |
| DW 294 | ; #11 AGE 3 |
| DW 303 | ; #12 AGE 3 |
| DW 296 | ; #13 AGE 3 |
| DW 304 | ; #14 AGE 3 |
| DW 298 | ; #15 AGE 3 |
| DW 299 | ; #16 AGE 3 |

**Light_S4:**

| DW 302 | ; #1 AGE 4 |
| DW 305 | ; 003 ; #2 AGE 4 |
| DW 294 | ; #3 AGE 4 |
| DW 306 | ; #4 AGE 4 |
| DW 296 | ; #5 AGE 4 |
| DW 307 | ; #6 AGE 4 |
| DW 298 | ; #7 AGE 4 |
| DW 299 | ; #8 AGE 4 |
| DW 302 | ; #9 AGE 4 |
| DW 305 | ; 003 ; #10 AGE 4 |
| DW 294 | ; #11 AGE 4 |
| DW 306 | ; #12 AGE 4 |
| DW 296 | ; #13 AGE 4 |
| DW 307 | ; #14 AGE 4 |
| DW 298 | ; #15 AGE 4 |
| DW 299 | ; #16 AGE 4 |

};

; light sense table (DARK SENSE)
; DO DARK
; Hide and Seek game table

Peek_S1:  DW  000  ;   #0  AGE 1
            DW  000  ;   #1  AGE 1
            DW  000  ;   #2  AGE 1
            DW  000  ;   #3  AGE 1
            DW  000  ;   #4  AGE 1
            DW  000  ;   #5  AGE 1
            DW  000  ;   #6  AGE 1
            DW  000  ;   #7  AGE 1
            DW  000  ;   #8  AGE 1
            DW  000  ;   #9  AGE 1
            DW  000  ;  #10  AGE 1
            DW  000  ;  #11  AGE 1
            DW  000  ;  #12  AGE 1
            DW  000  ;  #13  AGE 1
            DW  000  ;  #14  AGE 1
            DW  000  ;  #15  AGE 1

Peek_S2:  DW  000  ;   #0  AGE 2
            DW  000  ;   #1  AGE 2
            DW  000  ;   #2  AGE 2
            DW  000  ;   #3  AGE 2
            DW  000  ;   #4  AGE 2
            DW  000  ;   #5  AGE 2
            DW  000  ;   #6  AGE 2
            DW  000  ;   #7  AGE 2
            DW  000  ;   #8  AGE 2
            DW  000  ;   #9  AGE 2
            DW  000  ;  #10  AGE 2
            DW  000  ;  #11  AGE 2
            DW  000  ;  #12  AGE 2
            DW  000  ;  #13  AGE 2
            DW  000  ;  #14  AGE 2
            DW  000  ;  #15  AGE 2

Peek_S3:  DW  000  ;   #0  AGE 3
            DW  000  ;   #1  AGE 3
            DW  000  ;   #2  AGE 3
            DW  000  ;   #3  AGE 3
            DW  000  ;   #4  AGE 3
            DW  000  ;   #5  AGE 3
            DW  000  ;   #6  AGE 3
            DW  000  ;   #7  AGE 3
            DW  000  ;   #8  AGE 3
            DW  000  ;   #9  AGE 3
            DW  000  ;  #10  AGE 3
            DW  000  ;  #11  AGE 3
            DW  000  ;  #12  AGE 3
            DW  000  ;  #13  AGE 3
            DW  000  ;  #14  AGE 3
            DW  000  ;  #15  AGE 3
DW 000 ; #11 AGE 3
DW 000 ; #12 AGE 3
DW 000 ; #13 AGE 3
DW 000 ; #14 AGE 3
DW 000 ; #15 AGE 3
Peek_S4: DW 000 ; #0 AGE 4
DW 000 ; #1 AGE 4
DW 000 ; #2 AGE 4
DW 000 ; #3 AGE 4
DW 000 ; #4 AGE 4
DW 000 ; #5 AGE 4
DW 000 ; #6 AGE 4
DW 000 ; #7 AGE 4
DW 000 ; #8 AGE 4
DW 000 ; #9 AGE 4
DW 000 ; #10 AGE 4
DW 000 ; #11 AGE 4
DW 000 ; #12 AGE 4
DW 000 ; #13 AGE 4
DW 000 ; #14 AGE 4
DW 000 ; #15 AGE 4

Macro_grpl: ;points into macro tables

DW Tbl1_Macro0
DW Tbl1_Macro1,Tbl1_Macro2,Tbl1_Macro3,Tbl1_Macro4,Tbl1_Macro5
DW Tbl1_Macro6,Tbl1_Macro7,Tbl1_Macro8,Tbl1_Macro9,Tbl1_Macro10
DW Tbl1_Macro11,Tbl1_Macro12,Tbl1_Macro13,Tbl1_Macro14,Tbl1_Macro15
DW Tbl1_Macro16,Tbl1_Macro17,Tbl1_Macro18,Tbl1_Macro19,Tbl1_Macro20
DW Tbl1_Macro21,Tbl1_Macro22,Tbl1_Macro23,Tbl1_Macro24,Tbl1_Macro25
DW Tbl1_Macro26,Tbl1_Macro27,Tbl1_Macro28,Tbl1_Macro29,Tbl1_Macro30
DW Tbl1_Macro31,Tbl1_Macro32,Tbl1_Macro33,Tbl1_Macro34,Tbl1_Macro35
DW Tbl1_Macro36,Tbl1_Macro37,Tbl1_Macro38,Tbl1_Macro39,Tbl1_Macro40
DW Tbl1_Macro41,Tbl1_Macro42,Tbl1_Macro43,Tbl1_Macro44,Tbl1_Macro45
DW Tbl1_Macro46,Tbl1_Macro47,Tbl1_Macro48,Tbl1_Macro49,Tbl1_Macro50
DW Tbl1_Macro51,Tbl1_Macro52,Tbl1_Macro53,Tbl1_Macro54,Tbl1_Macro55
DW Tbl1_Macro56,Tbl1_Macro57,Tbl1_Macro58,Tbl1_Macro59,Tbl1_Macro60
DW Tbl1_Macro61,Tbl1_Macro62,Tbl1_Macro63,Tbl1_Macro64,Tbl1_Macro65
DW Tbl1_Macro66,Tbl1_Macro67,Tbl1_Macro68,Tbl1_Macro69,Tbl1_Macro70
DW Tbl1_Macro71,Tbl1_Macro72,Tbl1_Macro73,Tbl1_Macro74,Tbl1_Macro75
DW Tbl1_Macro76,Tbl1_Macro77,Tbl1_Macro78,Tbl1_Macro79,Tbl1_Macro80
DW Tbl1_Macro81,Tbl1_Macro82,Tbl1_Macro83,Tbl1_Macro84,Tbl1_Macro85
DW Tbl1_Macro86,Tbl1_Macro87,Tbl1_Macro88,Tbl1_Macro89,Tbl1_Macro90
DW Tbl1_Macro91,Tbl1_Macro92,Tbl1_Macro93,Tbl1_Macro94,Tbl1_Macro95
DW Tbl1_Macro96,Tbl1_Macro97,Tbl1_Macro98,Tbl1_Macro99
DW Tbl1_Macro100,Tbl1_Macro101,Tbl1_Macro102,Tbl1_Macro103,Tbl1_Macro104
DW Tbl1_Macro105,Tbl1_Macro106,Tbl1_Macro107,Tbl1_Macro108,Tbl1_Macro109
DW Tbl1_Macro110,Tbl1_Macro111,Tbl1_Macro112,Tbl1_Macro113,Tbl1_Macro114
DW Tbl1_Macro115,Tbl1_Macro116,Tbl1_Macro117,Tbl1_Macro118,Tbl1_Macro119
DW Tbl1_Macro120,Tbl1_Macro121,Tbl1_Macro122,Tbl1_Macro123,Tbl1_Macro124

A-146
DW Tbl1_Macro125, Tbl1_Macro126, Tbl1_Macro127
;
Macro_grp2: ; points into macro tables
DW Tbl2_Macro128
DW Tbl2_Macro129, Tbl2_Macro130, Tbl2_Macro131, Tbl2_Macro132, Tbl2_Macro133
DW Tbl2_Macro134, Tbl2_Macro135, Tbl2_Macro136, Tbl2_Macro137, Tbl2_Macro138
DW Tbl2_Macro139, Tbl2_Macro140, Tbl2_Macro141, Tbl2_Macro142, Tbl2_Macro143
DW Tbl2_Macro144, Tbl2_Macro145, Tbl2_Macro146, Tbl2_Macro147, Tbl2_Macro148
DW Tbl2_Macro149, Tbl2_Macro150, Tbl2_Macro151, Tbl2_Macro152, Tbl2_Macro153
DW Tbl2_Macro154, Tbl2_Macro155, Tbl2_Macro156, Tbl2_Macro157, Tbl2_Macro158
DW Tbl2_Macro159, Tbl2_Macro160, Tbl2_Macro161, Tbl2_Macro162, Tbl2_Macro163
DW Tbl2_Macro164, Tbl2_Macro165, Tbl2_Macro166, Tbl2_Macro167, Tbl2_Macro168
DW Tbl2_Macro169, Tbl2_Macro170, Tbl2_Macro171, Tbl2_Macro172, Tbl2_Macro173
DW Tbl2_Macro174, Tbl2_Macro175, Tbl2_Macro176, Tbl2_Macro177, Tbl2_Macro178
DW Tbl2_Macro179, Tbl2_Macro180, Tbl2_Macro181, Tbl2_Macro182, Tbl2_Macro183
DW Tbl2_Macro184, Tbl2_Macro185, Tbl2_Macro186, Tbl2_Macro187, Tbl2_Macro188
DW Tbl2_Macro189, Tbl2_Macro190, Tbl2_Macro191, Tbl2_Macro192, Tbl2_Macro193
DW Tbl2_Macro194, Tbl2_Macro195, Tbl2_Macro196, Tbl2_Macro197, Tbl2_Macro198
DW Tbl2_Macro199, Tbl2_Macro200, Tbl2_Macro201, Tbl2_Macro202, Tbl2_Macro203
DW Tbl2_Macro204, Tbl2_Macro205, Tbl2_Macro206, Tbl2_Macro207, Tbl2_Macro208
DW Tbl2_Macro209, Tbl2_Macro210, Tbl2_Macro211, Tbl2_Macro212, Tbl2_Macro213
DW Tbl2_Macro214, Tbl2_Macro215, Tbl2_Macro216, Tbl2_Macro217, Tbl2_Macro218
DW Tbl2_Macro219, Tbl2_Macro220, Tbl2_Macro221, Tbl2_Macro222, Tbl2_Macro223
DW Tbl2_Macro224, Tbl2_Macro225, Tbl2_Macro226, Tbl2_Macro227, Tbl2_Macro228
DW Tbl2_Macro229, Tbl2_Macro230, Tbl2_Macro231, Tbl2_Macro232, Tbl2_Macro233
DW Tbl2_Macro234, Tbl2_Macro235, Tbl2_Macro236, Tbl2_Macro237, Tbl2_Macro238
DW Tbl2_Macro239, Tbl2_Macro240, Tbl2_Macro241, Tbl2_Macro242, Tbl2_Macro243
DW Tbl2_Macro244, Tbl2_Macro245, Tbl2_Macro246, Tbl2_Macro247, Tbl2_Macro248
DW Tbl2_Macro249, Tbl2_Macro250, Tbl2_Macro251, Tbl2_Macro252, Tbl2_Macro253
DW Tbl2_Macro254, Tbl2_Macro255
;
Macro_grp3: ; points into macro tables

DW Tbl3_Macro256
DW Tbl3_Macro257, Tbl3_Macro258, Tbl3_Macro259, Tbl3_Macro260, Tbl3_Macro261
DW Tbl3_Macro262, Tbl3_Macro263, Tbl3_Macro264, Tbl3_Macro265, Tbl3_Macro266
DW Tbl3_Macro267, Tbl3_Macro268, Tbl3_Macro269, Tbl3_Macro270, Tbl3_Macro271
DW Tbl3_Macro272, Tbl3_Macro273, Tbl3_Macro274, Tbl3_Macro275, Tbl3_Macro276
DW Tbl3_Macro277, Tbl3_Macro278, Tbl3_Macro279, Tbl3_Macro280, Tbl3_Macro281
DW Tbl3_Macro282, Tbl3_Macro283, Tbl3_Macro284, Tbl3_Macro285, Tbl3_Macro286
DW Tbl3_Macro287, Tbl3_Macro288, Tbl3_Macro289, Tbl3_Macro290, Tbl3_Macro291
DW Tbl3_Macro292, Tbl3_Macro293, Tbl3_Macro294, Tbl3_Macro295, Tbl3_Macro296
DW Tbl3_Macro297, Tbl3_Macro298, Tbl3_Macro299, Tbl3_Macro300, Tbl3_Macro301
DW Tbl3_Macro302, Tbl3_Macro303, Tbl3_Macro304, Tbl3_Macro305, Tbl3_Macro306
DW Tbl3_Macro307, Tbl3_Macro308, Tbl3_Macro309, Tbl3_Macro310, Tbl3_Macro311
DW Tbl3_Macro312, Tbl3_Macro313, Tbl3_Macro314, Tbl3_Macro315, Tbl3_Macro316
DW Tbl3_Macro317, Tbl3_Macro318, Tbl3_Macro319, Tbl3_Macro320, Tbl3_Macro321
DW Tbl3_Macro322, Tbl3_Macro323, Tbl3_Macro324, Tbl3_Macro325, Tbl3_Macro326
DW Tbl3_Macro327, Tbl3_Macro328, Tbl3_Macro329, Tbl3_Macro330, Tbl3_Macro331
DW Tbl3_Macro332, Tbl3_Macro333, Tbl3_Macro334, Tbl3_Macro335, Tbl3_Macro336
DW Tbl3_Macro337, Tbl3_Macro338, Tbl3_Macro339, Tbl3_Macro340, Tbl3_Macro341
DW Tbl3_Macro342, Tbl3_Macro343, Tbl3_Macro344, Tbl3_Macro345, Tbl3_Macro346
DW Tbl3_Macro347, Tbl3_Macro348, Tbl3_Macro349, Tbl3_Macro350, Tbl3_Macro351
DW Tbl3_Macro352, Tbl3_Macro353, Tbl3_Macro354, Tbl3_Macro355, Tbl3_Macro356
DW Tbl3_Macro357, Tbl3_Macro358, Tbl3_Macro359, Tbl3_Macro360, Tbl3_Macro361
DW Tbl3_Macro362, Tbl3_Macro363, Tbl3_Macro364, Tbl3_Macro365, Tbl3_Macro366
DW Tbl3_Macro367, Tbl3_Macro368, Tbl3_Macro369, Tbl3_Macro370, Tbl3_Macro371
DW Tbl3_Macro372, Tbl3_Macro373, Tbl3_Macro374, Tbl3_Macro375, Tbl3_Macro376
DW Tbl3_Macro377, Tbl3_Macro378, Tbl3_Macro379, Tbl3_Macro380, Tbl3_Macro381
DW Tbl3_Macro382, Tbl3_Macro383

Macro_grp4: ; points into macro tables

DW Tbl4_Macro384
DW Tbl4_Macro385, Tbl4_Macro386, Tbl4_Macro387, Tbl4_Macro388, Tbl4_Macro389
DW Tbl4_Macro390, Tbl4_Macro391, Tbl4_Macro392, Tbl4_Macro393, Tbl4_Macro394
MACRO TABLES

The sensor tables point into the Macro table. This table in turn gets speech and motor table data. This can be an entry of 1-511 and effectively chains motor and speech tables together to reuse previous speech motor segments.
The first group of numbers is the speech/motor table value.
The last line is the terminator of 00. (00 so 'DB' takes 1 less byte)
ex: 1 = will call the saysent 1 and the motor table 1.

Tbl1_Macro0:
  DW 511
  DW 00 ; end

; FOR NAME TESTING EMH
; WAKE
  DW 124 ; 02
  DW 125
  DW 126

  DW 399 ; delay
  DW 395 ; ME
  DW 224 ; MAY-LAH-KA
  DW 152
  DW 00 ; end

; (MIDDLE)

; put sounds and motions together
; DW 5  (first sound and motion, in this case "5")
; DW 3  (next sound and motion, in this case "3")
; DW 00  (end of sequence)

Tbl1_Macro1:
  DW 01
  DW 00 ; end

;GEORGE 07/03/98

Tbl1_Macro2:
  DW 001 ; FRONT SEQ1AGE1
  DW 00 ; end

Tbl1_Macro3:
  DW 002 ; FRONT SEQ2AGE1
  DW 00 ; end

Tbl1_Macro4:
  DW 003 ; FRONT SEQ3AGE1
  DW 004
  DW 00 ; end

Tbl1_Macro5:
  DW 003 ; FRONT SEQ4AGE1
  DW 005
  DW 00 ; end

Tbl1_Macro6:
  DW 006 ; FRONT SEQ5AGE1
  DW 00 ; end
Tbl1_Macro7:
  DW 006 ;FRONTSEQ6AGE1
  DW 007
  DW 00 ;end

Tbl1_Macro8:
  DW 008 ;FRONT SEQ7AGE1
  DW 003
  DW 00 ;end

Tbl1_Macro9:
  DW 009 ;FRONTSEQ8AGE1
  DW 003
  DW 00 ;end

Tbl1_Macro10:
  DW 010 ;FRONT SEQ9AGE1
  DW 00 ;end

Tbl1_Macro11:
  DW 011
  DW 011 ;frontseq10age1
  DW 00 ;end

Tbl1_Macro12:
  DW 012 ;seq11 FRONT AGE1 ADD SAY001
  DW 00 ;end

Tbl1_Macro13:
  DW 001
  DW 013
  DW 00 ;end ;seq12 FRONT AGE1 ADD SAY001

Tbl1_Macro14:
  DW 014 ;seq13 FRONT AGE1 ADD SAY003
  DW 00 ;end

Tbl1_Macro15:
  DW 015
  DW 00 ;end ;seq14 FRONT AGE1

Tbl1_Macro16:
  DW 016
  DW 00 ;end ;seq15 FRONT AGE1

Tbl1_Macro17:
  DW 001
  DW 017
  DW 018
  DW 001
  DW 00 ;end ;seq16 FRONT AGE1 BETWEEN 2(20)

Tbl1_Macro18:
  DW 019
  DW 00 ;end ;FRONT SEQ1AGE2

Tbl1_Macro19:
  DW 001
; FRONT SEQ2 AGE2
DW 020 ;end

Thu1Macro20:
    DW 010 ;SEQ3 AGE2 FRONT ADD SEQ9 AGE1
    DW 021 ;end

Thu1Macro21:
    DW 022 ;SEQ4 AGE2 FRONT
    DW 023 ;end

Thu1Macro22:
    DW 024 ;SEQ5 AGE2 FRONT
    DW 00 ;end

Thu1Macro23:
    DW 025 ;SEQ6 AGE2 FRONT
    DW 00 ;end

Thu1Macro24:
    DW 026 ;SEQ7 AGE2 FRONT PART1
    DW 027 ;end

Thu1Macro25:
    DW 026 ;SEQ8 AGE2 FRONT
    DW 026 ;end

Thu1Macro26:
    DW 026 ;SEQ9 FRONT
    DW 00 ;end

Thu1Macro27:
    DW 030 ;SEQ10 FRONT AGE2
    DW 029 ;end

Thu1Macro28:
    DW 022 ;SEQ11 FRONT AGE2
    DW 031 ;end

Thu1Macro29:
    DW 001 ;SEQ12 FRONT AGE2
    DW 032 ;end

Thu1Macro30:
    DW 014 ;SEQ13 FRONT AGE1&2 ADD SAY 003
    DW 003 ;end

Thu1Macro31:
    DW 033 ;SEQ14 FRONT AGE2
nm

Tb11_Macro32:
DW 001 ;end
;SEQ15 FRONT AGE2

Tb11_Macro33:
DW 035 ;SEQ16 FRONT AGE2

Tb11_Macro34:
DW 001 ;SEQ1 FRONT AGE3
DW 036 ;end

Tb11_Macro35:
DW 003
DW 037 ;SEQ2 FRONT AGE3
DW 00 ;end

Tb11_Macro36:
DW 015 ;SEQ3 FRONT AGE3
DW 038 ;end

Tb11_Macro37:
DW 015 ;SEQ4 FRONT AGE3
DW 039 ;end

Tb11_Macro38:
DW 015 ;SEQ5 FRONT AGE3
DW 023 ;end

Tb11_Macro39:
DW 040 ;SEQ6 FRONT AGE3
DW 00 ;end

Tb11_Macro40:
DW 041 ;SEQ7 FRONT AGE3
DW 003 ;end

Tb11_Macro41:
DW 042 ;SEQ8 FRONT AGE3
DW 003 ;end

Tb11_Macro42:
DW 043 ;SEQ9 FRONT AGE3
DW 001 ;end

Tb11_Macro43:
DW 044 ;SEQ10 FRONT AGE3
DW 00 ;end
Table Macro 44:
   DW 045
   DW 001
   DW 00       ; SEQ12 FRONT AGE3 (HEEY, TICKLE ME) ADD20
   ; end

Table Macro 45:
   DW 001
   DW 046
   DW 047
   DW 00       ; SEQ13 FRONT AGE3 (NANNY, NANNY) ADD20
   ; RASBERAY HE HE HE
   ; end

Table Macro 46:
   DW 003
   DW 028
   DW 00       ; SEQ14 FRONT AGE3
   ; end

Table Macro 47:
   DW 034
   DW 001
   DW 00       ; SEQ15 FRONT AGE3
   ; end

Table Macro 48:
   DW 001
   DW 048
   DW 049
   DW 00       ; SEQ16 FRONT AGE3
   ; end

Table Macro 49:
   DW 044
   DW 00       ; SEQ1 FRONT AGE4
   ; end

Table Macro 50:
   DW 001
   DW 050
   DW 051
   DW 00       ; SEQ2 FRONT AGE4
   ; end

Table Macro 51:
   DW 003
   DW 052
   DW 050
   DW 053
   DW 00       ; SEQ3 (YOU) FRONT AGE4
   ; end

Table Macro 52:
   DW 026
   DW 053
   DW 054
   DW 050
   DW 001
   DW 00       ; SEQ4 FRONT AGE4
   ; end

Table Macro 53:
   DW 007
   DW 055
   DW 056
   DW 00       ; SEQ5 FRONT AGE4
   ; end

Table Macro 54:
; SEQ5 FRONT AGE4
DW 026
DW 053
DW 054
DW 052
DW 018
DW 00

; SEQ7 FRONT AGE4
DW 001
DW 046
DW 055
DW 00

; SEQ8 FRONT AGE4
DW 026
DW 057
DW 050
DW 051
DW 058
DW 003
DW 00

; SEQ9 FRONT AGE4
DW 042,001
DW 00

; SEQ10 FRONT AGE4
DW 059
DW 050
DW 00

; SEQ11 FRONT AGE4
DW 044
DW 003
DW 00

; SEQ12
DW 001
DW 00

; SEQ13 FRONT AGE4
DW 001
DW 046
DW 047
DW 00

; SEQ14 FRONT AGE4
DW 026
DW 060
DW 00

; SEQ15 FRONT AGE4
DW 061
DW 003
DW 00

; SEQ16 FRONT AGE4
DW 007
DW 051
DW 00

A-155
; END GEORGE 07/03/98

; GEORGE 07/04/98
; START FORTUNE

; Tbl1_Macro65:
  DW 062
  DW 051 ;72 ; FORTUNE 1
  DW 00 ; end

; Tbl1_Macro66:
  DW 003
  DW 063
  DW 003
  DW 00 ; end

; Tbl1_Macro67:
  DW 090 ;94
  DW 064
  DW 063 ; FORTUNE 3
  DW 00 ; end

; Tbl1_Macro68:
  DW 065 ; FORTUNE 4
  DW 063
  DW 00 ; end

; Tbl1_Macro69:
  DW 067 ; FORTUNE 5
  DW 068
  DW 053
  DW 066
  DW 063
  DW 00 ; end

; Tbl1_Macro70:
  DW 069 ; FORTUNE 6
  DW 070
  DW 00 ; end

; Tbl1_Macro71:
  DW 067 ; FORTUNE 7
  DW 068
  DW 071
  DW 073
  DW 072
  DW 00 ; end

; Tbl1_Macro72:
  DW 074 ; FORTUNE 8
  DW 00 ; end

; Tbl1_Macro73:
  DW 074 ; FORTUNE 9
  DW 063
  DW 00 ; end

; Tbl1_Macro74:
DW 069 ;FORTUNE 10
DW 00 ;end

;
Tbl1_Macro75:
DW 064 ;FORTUNE 11
DW 069
DW 00 ;end

;
Tbl1_Macro76:
DW 073
DW 064 ;FORTUNE 12
DW 069
DW 00 ;end

;
Tbl1_Macro77:
: MODIFIED TO WORK WITH NAME DMH
;
DW 067
DW 068
DW 053 ;FORTUNE 13
DW 066
DW 069
DW 00 ;end

;
Tbl1_Macro78:
DW 071
DW 073
DW 069
DW 075 ;FORTUNE 14
DW 00 ;end

;
Tbl1_Macro79:
DW 076
DW 077 ;FORTUNE 15
DW 00 ;end

;
Tbl1_Macro80:
DW 076
DW 069 ;FORTUNE 16
DW 00 ;end

;
Tbl1_Macro81:
DW 078 ;FORTUNE 17 SEQ1 AGE2
DW 00 ;end

;
Tbl1_Macro82:
DW 078 ;FORTUNE 18 SEQ2 AGE2
DW 063
DW 00 ;end

;
Tbl1_Macro83:
DW 078 ;FORTUNE 19 SEQ2 AGE2
DW 069
DW 00 ;end

Tbl1_Macro84:
: SPECIAL "O TWO HA"
DW 067
DW 068
DW 00
;END GEORGE 07/04/98
;END FORTUNE
;START HANGOUT
;GEORGE 07/04/98
Tbl1_Macro85:
  DW 079
  DW 080
  DW 079 ;SEQ1 HANGING
  DW 080
  DW 00 ;end
;
Tbl1_Macro86:
  DW 081 ;SEQ2 HANGING
  DW 081
  DW 00 ;end
;
Tbl1_Macro87:
  DW 082
  DW 083
  DW 083
  DW 084 ;SEQ3 HANGING (YA DA DA OMPAH DRUMM BABABUM)
  DW 00 ;end
;
Tbl1_Macro88:
  DW 085
  DW 085
  DW 086
  DW 087 ;SEQ4 HANGING (LA LA)
  DW 00 ;end
;
Tbl1_Macro89:
  DW 087
  DW 088 ;SEQ5 HANGING
  DW 00 ;end
;
Tbl1_Macro90:
  DW 089
  DW 089
  DW 090 ;SEQ6 HANGING
  DW 091
  DW 092
  DW 00 ;end
;
Tbl1_Macro91:
  DW 093 ;SEQ7 HANGING (SOFTER)
  DW 093
  DW 093
  DW 094
  DW 00 ;end
;
Tbl1_Macro92:
  DW 095
  DW 095
  DW 095 ;WAS 76 ;SEQ8 HANGING
  DW 00 ;end
;
Tbl1_Macro93:
  DW 096 ;SEQ9 HANGING
  DW 00 ;end
Tbll_Macro94:
  DW 097 ; SEQ10 HANGING
  DW 00 ; end

Tbll_Macro95:
  DW 098 ; SEQ11 AND SEQ12 HANGING (STGH)
  DW 00 ; end

Tbll_Macro96:
  DW 099 ; SEQ13 HANGING (HAA)
  DW 00 ; end

Tbll_Macro97:
  DW 100 ; SEQ14 SEQ15 HANGING (hEEY)
  DW 00 ; end

Tbll_Macro98:
  DW 101 ; SEQ16 HANGING (F.ONE)
  DW 102
  DW 101
  DW 101
  DW 001 ; 20
  DW 00 ; end

Tbll_Macro99:
  DW 089 ; SEQ6 HANGING AGE2
  DW 089
  DW 090
  DW 091
  DW 103
  DW 00 ; end

Tbll_Macro100:
  DW 089 ; SEQ6 HANGING AGE2
  DW 089
  DW 090
  DW 105
  DW 104
  DW 103
  DW 00 ; end

Tbll_Macro101:
  DW 087 ; SEQ5 AGE3 4
  DW 106 ; SEQ5 AGE3 4
  DW 00 ; end

; END HANGOUT

Tbll_Macro102:
  DW 107 ; Fortune pause
  DW 00 ; end

; END GEORGE 07/04/98
; GEORGE 07/05/98

; FEED TABLE
Tbll_Macro103:
  DW 108
  DW 110 ; SEQ2 FEED AGE1
  DW 109
  DW 00 ; end
; Tbl1_Macro04:
  DW 108 ;SEQ3 FEED AGE1
  DW 111
  DW 112
  DW 109
  DW 00 ;end

; Tbl1_Macro05:
  DW 108 ;SEQ4 FEED AGE1
  DW 110
  DW 113
  DW 109
  DW 00 ;end

; Tbl1_Macro06:
  DW 108 ;SEQ5 FEED AGE1
  DW 108
  DW 078 ;127
  DW 110
  DW 109
  DW 00 ;end

; Tbl1_Macro07:
  DW 108 ;SEQ6 FEED AGE1
  DW 105
  DW 114
  DW 00 ;end

; Tbl1_Macro08:
  DW 108 ;SEQ7 FEED AGE1
  DW 115
  DW 116
  DW 117
  DW 110
  DW 00 ;end

; Tbl1_Macro09:
  DW 076 ;125 ;SEQ8 FEED AGE1
  DW 117
  DW 120
  DW 118
  DW 00 ;end

; Tbl1_Macro10:
  DW 108 ;SEQ9 FEED AGE1
  DW 115
  DW 20
  DW 00 ;end

; Tbl1_Macro11:
  DW 108 ;SEQ10 FEED AGE1
  DW 109
  DW 00 ;end

; Tbl1_Macro12:
  DW 108 ;SEQ11 FEED AGE1
  DW 076 ;125
  DW 117
  DW 119
Tbll_Macro113:
  DW  108 ;SEQ12 FEED AGE1
  DW  108
  DW  109
  DW  00 ;end

Tbll_Macro114:
  DW  108 ;SEQ13 REUSE 10 FOR14 FEED AGE1
  DW  115
  DW  001 ;20
  DW  00 ;end

Tbll_Macro115:
  DW  108 ;SEQ15 FEED AGE1
  DW  076 ;125
  DW  117
  DW  119
  DW  00

Tbll_Macro116:
  DW  108
  DW  108
  DW  109 ;SEQ1 FEED AGE1 ()
  DW  00 ;end

Tbll_Macro117:
  ;WIERD SHIT SEE 201
  DW  108
  DW  120
  DW  109
  DW  00 ;end

Tbll_Macro118:
  DW  108
  DW  121
  DW  109 ;SEQ1 FEED AGE2
  DW  00 ;end

Tbll_Macro119:
  DW  108
  DW  051 ;72
  DW  109 ;SEQ2 FEED AGE2
  DW  00 ;end

Tbll_Macro120:
  DW  108
  DW  073 ;122
  DW  112
  DW  109 ;SEQ3 FEED AGE2
  DW  00 ;end

Tbll_Macro121:
  DW  108
  DW  051 ;72
  DW  113
  DW  109 ;SEQ4 FEED AGE2
  DW  00 ;end

Tbll_Macro122:

A-161
DW 108
DW 108
DW 078 ;127 ;SEQ5 FEED AGE2
DW 051 ;72
DW 109
DW 00 ;end

; Tbl1_Macro123:
DW 108
DW 105 ;109
DW 114 ;SEQ6 FEED AGE2
DW 00 ;end

; Tbl1_Macro124:
DW 108
DW 115
DW 116
DW 069 ;118 ;SEQ7 FEED AGE2
DW 110
DW 00 ;end

; Tbl1_Macro125:
DW 076 ;125
DW 057 ;78
DW 120
DW 116 ;SEQ8 FEED AGE2
DW 00 ;end

; Tbl1_Macro126:
DW 108
DW 115 ;SEQ9 FEED AGE2
DW 001 ;20
DW 00 ;end

; Tbl1_Macro127:
DW 108
DW 109 ;SEQ10 FEED AGE2
DW 00 ;end

; Macro grp2 was here

; Tbl2_Macro128:
DW 108
DW 076 ;125
DW 069 ;118
DW 119 ;SEQ11 FEED AGE2
DW 00 ;end

; Macro grp2 was here

Tbl2_Macro129:
DW 108
DW 076 ;125
DW 069 ;118
DW 119 ;SEQ15 FEED AGE2
DW 00 ;end

---------------------------------END AGE2-------------
Tbl2_Macro130:
  DW 108
  DW 110
  DW 109 ;SEQ2 FEED AGE3
  DW 00 ;end

Tbl2_Macro131:
  DW 108
  DW 111
  DW 072 ;143
  DW 109 ;SEQ3 FEED AGE3
  DW 00 ;end

Tbl2_Macro132:
  DW 108
  DW 110
  DW 058 ;144
  DW 109 ;SEQ4 FEED AGE3
  DW 00 ;end

Tbl2_Macro133:
  DW 108
  LW 115
  DW 116
  DW 117
  DW 051 ;72 ;SEQ7 FEED AGE3
  DW 00 ;end

Tbl2_Macro134:
  DW 076 ;125
  DW 117
  DW 121
  DW 118 ;SEQ8 FEED AGE3
  DW 00 ;end

Tbl2_Macro135:
  DW 108
  DW 076 ;125
  DW 117 ;SEQ11 FEED AGE3
  DW 122
  DW 00 ;end

Tbl2_Macro136:
  DW 108
  DW 051 ;72
  DW 109
  DW 00 ;end

Tbl2_Macro137:
  DW 108
  DW 073 ;122
  DW 072 ;121
  DW 109
  DW 00 ;end

Tbl2_Macro138:
  DW 108
  DW 051 ;72
  DW 058 ;144
  DW 109
DW 00 ; end

// Table Macro139:
DW 108
DW 108
DW 078 ; 127
DW 051 ; 72
DW 109
DW 00 ; end

// Table Macro140:
DW 108 ; SEQ 6
DW 105 ; 109
DW 123
DW 00 ; end

// Table Macro141:
DW 108
DW 115
DW 116
DW 057 ; 78
DW 051 ; 72
DW 00 ; end

// Table Macro142:
DW 076 ; 125
DW 069 ; 118
DW 121
DW 118
DW 00 ; end

// Table Macro143:
DW 108
DW 125
DW 057 ; 78
DW 122
DW 00 ; end

// Table Macro144:
DW 108
DW 125
DW 057 ; 78
DW 122
DW 00 ; end

// Table Macro145:
DW 108
DW 121
DW 109
DW 00 ; end

//END FEED
//END GEORGE 07/05/98

//NAME
//GEORGE 07/06/98
//Table Macro146: ; SG DONE
DW 124 ; 02
DW 125
DW 126
DW 00 ; end
; Tbl2_Macro147:  ;SG DONE
    DW    124
    DW    125
    DW    127
    DW    00  ;end

; Tbl2_Macro148:  ;SG DONE
    DW    124
    DW    128
    DW    127
    DW    00  ;end

; Tbl2_Macro149:  ;SG DONE
    DW    124
    DW    129
    DW    055  ;*00
    DW    00  ;end

; Tbl2_Macro150:  ;SG DONE
    DW    124
    DW    130
    DW    131
    DW    132
    DW    00  ;end

; Tbl2_Macro151:  ;SG DONE
    DW    124
    DW    130
    DW    131
    DW    123  ;*12
    DW    00  ;end

; Tbl2_Macro152:  ;SG DONE
    DW    124
    DW    130
    DW    133
    DW    132
    DW    00  ;end

; Tbl2_Macro153:  ;SG DONE
    DW    124
    DW    130
    DW    133
    DW    123  ;*12
    DW    00  ;end

; Tbl2_Macro154:  ;SG DONE
    DW    124
    DW    134
    DW    135
    DW    131
    DW    00  ;end

; Tbl2_Macro155:  ;SG DONE
    DW    124
    DW    134
    DW    136
    DW    131
    DW    00  ;end
Tbl2_Macro156: ; SG DONE
  DW 124
  DW 134
  DW 135
  DW 133
  DW 00 ; end

Tbl2_Macro157: ; SG DONE
  DW 124
  DW 134
  DW 136
  DW 137
  DW 133
  DW 00 ; end

Tbl2_Macro158: ; SG DONE
  DW 124
  DW 138
  DW 139
  DW 00 ; end

Tbl2_Macro159: ; SG DONE
  DW 124
  DW 140
  DW 141
  DW 00 ; end

Tbl2_Macro160: ; SG DONE
  DW 124
  DW 142
  DW 143
  DW 141
  DW 00 ; end

Tbl2_Macro161: ; SG DONE
  DW 124
  DW 144
  DW 145
  DW 146
  DW 141
  DW 00 ; end

Tbl2_Macro162: ; SG DONE
  DW 124
  DW 147
  DW 141
  DW 00 ; end

Tbl2_Macro163: ; SG DONE
  DW 124
  DW 146
  DW 00 ; end

Tbl2_Macro164: ; SG DONE
  DW 124
  DW 053 ;29
  DW 149
  DW 150
  DW 00 ; end
Tbl2_Macro165: ;SG DONE
DW 124
DW 151
DW 00 ;end

Tbl2_Macro166: ;SG DONE
DW 124
DW 152
DW 131
DW 153
DW 154
DW 00 ;end

Tbl2_Macro167: ;SG DONE
DW 124
DW 152
DW 155
DW 153
DW 154
DW 00 ;end

Tbl2_Macro168: ;SG DONE
DW 124
DW 152
DW 153
DW 121
DW 156
DW 154
DW 00 ;end

Tbl2_Macro169: ;SG DONE
DW 124
DW 053 ;38
DW 155
DW 156
DW 154
DW 00 ;end

;END WAKE 07/06/98
;END GEORGE

;GEORGE 07/06/98
/HUNGER

Tbl2_Macro170: ;SG DONE ;HUNGER
DW 159
DW 165
DW 412 ;DMH
DW 00 ;end

Tbl2_Macro171: ;SG DONE
DW 160
DW 165
DW 412 ;DMH
DW 00 ;end

Tbl2_Macro172: ;SG DONE
DW 160
DW 00 ;end

Tbl2_Macro173: ;SG DONE
DW  168
DW  159
DW  165
DW  412 ;DMH
DW  00 ;end

; Tbl2_Macro174: ;SG DONE
DW  168
DW  160
DW  165
DW  412 ;DMH
DW  00 ;end

; Tbl2_Macro175: ;SG DONE
DW  168
DW  160
DW  412 ;DMH
DW  00 ;end

; Tbl2_Macro176: ;SG DONE
DW  163
DW  156
DW  159
DW  00 ;end

; Tbl2_Macro177: ;SG DONE
DW  163
DW  158
DW  160
DW  00 ;end

; Tbl2_Macro178: ;SG DONE
DW  163
DW  157
DW  159
DW  00 ;end

; Tbl2_Macro179: ;SG DONE
DW  163
DW  157
DW  160
DW  00 ;end

; Tbl2_Macro180: ;SG DONE
DW  163
DW  168
DW  159
DW  163
DW  00 ;end

; Tbl2_Macro181: ;SG DONE
DW  163
DW  168
DW  160
DW  163
DW  00 ;end

; Tbl2_Macro182: ;SG DONE
DW  163
```
DW 163
DW 168
DW 161
DW 159
DW 165
DW 412 ; DMH
DW 00 ; end

; Tbl2_Macro183: ; SG DONE
DW 163
DW 163
DW 168
DW 161
DW 160
DW 165
DW 412 ; DMH
DW 00 ; end

; Tbl2_Macro184: ; SG DONE
DW 163
DW 163
DW 168
DW 162
DW 160
DW 00 ; end

; Tbl2_Macro185: ; SG DONE
DW 168
DW 161
DW 159
DW 00 ; end

; Tbl2_Macro186: ; SG DONE
DW 168
DW 161
DW 160
DW 00 ; end

; Tbl2_Macro187: ; SG DONE
DW 168
DW 162
DW 159
DW 00 ; end

; Tbl2_Macro188: ; SG DONE
DW 168
DW 162
DW 160
DW 00 ; end

; Tbl2_Macro189: ; SG DONE
DW 168
DW 166
DW 159
DW 00 ; end

; Tbl2_Macro190: ; SG DONE
DW 168
DW 167
DW 159
```
DW 00 ; end

Tbl2_Macro200: ; SG DONE
  DW 164
  DW 168 ; fs40
  DW 162
  DW 159
  DW 165
  DW 00 ; end

Tbl2_Macro201: ; SG DONE
  DW 164
  DW 168 ; 40
  DW 162
  DW 160
  DW 165
  DW 00 ; end

; END HUNGER
; END GEORGE 07/06/98
;
; INVERT
; GEORGE 07/07/98
Tbl2_Macro202: ; SG DONE ; INVERT
  DW 164 ; 64
  DW 00 ; end

Tbl2_Macro203: ; SG DONE
  DW 164 ; 64
  DW 169
  DW 00 ; end

Tbl2_Macro204: ; SG DONE
  DW 164 ; 64
  DW 1c8 ; 40
  DW 174
  DW 166
  DW 175
  DW 00 ; end

Tbl2_Macro205: ; SG DONE
  DW 164 ; 64
  DW 176
  DW 00 ; end

Tbl2_Macro206: ; SG DONE
  DW 188
  DW 177
  DW 00 ; end

Tbl2_Macro207: ; SG DONE
  DW 188
  DW 178
  DW 00 ; end

Tbl2_Macro208: ; SG DONE
  DW 170
  DW 177
  DW 177
DW 00 ;end

Tbl2_Macro209: ;SG DONE
    DW 170
    DW 178
    DW 177
    DW 00 ;end

Tbl2_Macro210: ;SG DONE
    DW 170
    DW 177
    DW 178
    DW 00 ;end

Tbl2_Macro211: ;SG DONE
    DW 170
    DW 178
    DW 00 ;end

Tbl2_Macro212: ;SG DONE
    DW 171
    DW 163 ;63
    DW 00 ;end

Tbl2_Macro213: ;SG DONE
    DW 171
    DW 168 ;40
    DW 179
    DW 180
    DW 165 ;65
    DW 00 ;end

Tbl2_Macro214: ;SG DONE
    DW 171
    DW 168 ;40
    DW 181
    DW 180
    DW 165 ;65
    DW 00 ;end

Tbl2_Macro215: ;SG DONE
    DW 171
    DW 168
    DW 179
    DW 182
    DW 165 ;65
    DW 00 ;end

Tbl2_Macro216: ;SG DONE
    DW 171
    DW 168 ;40
    DW 181
    DW 182
    DW 00 ;end

Tbl2_Macro217: ;SG DONE
    DW 164 ;64
    DW 175
    DW 164 ;64
DW 00 ; end

; Tbl2_Macro218: ; SG DONE
  DW 164 ; 64
  DW 163
  DW 164 ; 64
  DW 00 ; end

; Tbl2_Macro219: ; SG DONE
  DW 164 ; 64
  DW 170
  DW 170
  DW 00 ; end

; Tbl2_Macro220: ; SG DONE
  DW 171
  DW 179
  DW 180
  DW 00 ; end

; Tbl2_Macro221: ; SG DONE
  DW 171
  DW 181
  DW 180
  DW 00 ; end

; Tbl2_Macro222: ; SG DONE
  DW 171
  DW 179
  DW 184
  DW 163 ; 63
  DW 00 ; end

; Tbl2_Macro223: ; SG DONE
  DW 171
  DW 181
  DW 185
  DW 00 ; end

; Tbl2_Macro224: ; SG DONE
  DW 164 ; 64
  DW 179
  DW 186
  DW 00 ; end

; Tbl2_Macro225: ; SG DONE
  DW 164 ; 64
  DW 181
  DW 186
  DW 00 ; end

; Tbl2_Macro226: ; SG DONE
  DW 164 ; 64
  DW 181
  DW 185
  DW 00 ; end

; Tbl2_Macro227: ; SG DONE
  DW 164 ; 64
  DW 181
DW 184
DW 163 ; 63
DW 00 ; end
;
Tbl2_Macro228: ; SG DONE
DW 164 ; 64
DW 179
DW 157
DW 00 ; end
;
Tbl2_Macro229: ; SG DONE
DW 164 ; 64
DW 181
DW 187
DW 00 ; end
;
Tbl2_Macro230: ; SG DONE
DW 172
DW 158
DW 178
DW 00 ; end
;
Tbl2_Macro231: ; SG DONE
DW 164 ; 64
DW 181
DW 189
DW 00 ; end
;
Tbl2_Macro232: ; SG DONE
DW 172
DW 175
DW 00 ; end
;
Tbl2_Macro233: ; SG DONE
DW 172
DW 183
DW 00 ; end
;
Tbl2_Macro234: ; SG DONE
DW 172
DW 172
DW 164 ; 64
DW 00 ; end
;
Tbl2_Macro235: ; SG DONE
DW 173
DW 00 ; end
;
Tbl2_Macro236: ; SG DONE
DW 190
DW 00 ; end
;
Tbl2_Macro237: ; SG DONE
DW 191
DW 00 ; end
;
Tbl2_Macro238: ; SG DONE
DW 192
DW 00 ; end
;
FND GEORGE 07/07/98
; END INVERT

; GEORGE 07/07/98
; BACK
Tbl2_Macro239: ; BACKSG ; SGDONE
   DW 193
   DW 193
   DW 00 ; end
;
Tbl2_Macro240: ; SGDONE
   DW 193
   DW 194
   DW 195
   DW 00 ; end
;
Tbl2_Macro241: ; SGDONE
   DW 193
   DW 196
   DW 195
   DW 00 ; end
;
Tbl2_Macro242: ; SGDONE
   DW 193
   DW 194
   DW 197
   DW 00 ; end
;
Tbl2_Macro243: ; SGDONE
   DW 193
   DW 196
   DW 197
   DW 00 ; end
;
Tbl2_Macro244: ; SGDONE
   DW 198
   DW 199
   DW 200
   DW 201
   DW 00 ; end
;
Tbl2_Macro245: ; SGDONE
   DW 198
   DW 199
   DW 202
   DW 201
   DW 00 ; end
;
Tbl2_Macro246: ; SGDONE
   DW 198
   DW 199
   DW 200
   DW 184 ; 148 ; 212
   DW 00 ; end
;
Tbl2_Macro247: ; SGDONE
   DW 198
   DW 199
   DW 202
   DW 184 ; 148 ; 212
   DW 00 ; end
; Tbl2_Macro248: ;SGDONE
  DW 198
  DW 198
  DW 00 ; end

; Tbl2_Macro249: ;SGDONE
  DW 198
  DW 203
  DW 204
  DW 00 ; end

; Tbl2_Macro250: ;SGDONE
  DW 198
  DW 205
  DW 206
  DW 207
  DW 204
  DW 00 ; end

; Tbl2_Macro251: ;SGDONE
  DW 198
  DW 205
  DW 206
  DW 233
  DW 204
  DW 00 ; end

; Tbl2_Macro252: ;SGDONE
  DW 198
  DW 205
  DW 206
  DW 233
  DW 204
  DW 00 ; end

; Tbl2_Macro253: ;SGDONE
  DW 198
  DW 205
  DW 210
  DW 00 ; end

; Tbl2_Macro254: ;SGDONE
  DW 198
  DW 209
  DW 212
  DW 213
  DW 00 ; end

; Tbl2_Macro255: ;SGDONE
  DW 198
  DW 209
  DW 214
  DW 00 ; end

; Tbl3_Macro256: ;SGDONE
  DW 198
  DW 215
  DW 216
DW 217
DW 00 ;end
;
Tbl3_Macro257: ;SGDONE
DW 198
DW 215
DW 216
DW 218
DW 00 ;end
;
Tbl3_Macro258: ;SGDONE
DW 219
DW 220
DW 209
DW 217
DW 199
DW 214
DW 00 ;end
;
Tbl3_Macro259: ;SGDONE
DW 219
DW 220
DW 209
DW 205
DW 217
DW 234
DW 00 ;end
;
Tbl3_Macro260: ;SGDONE
DW 219
DW 220
DW 209
DW 205
DW 216
DW 234
DW 00 ;end
;
Tbl3_Macro261: ;SGDONE
DW 221
DW 222
DW 00 ;end
;
Tbl3_Macro262: ;SGDONE
DW 221
DW 223
DW 222
DW 00 ;end
;
Tbl3_Macro263: ;SGDONE
DW 198
DW 224
DW 199
DW 00 ;end
;
Tbl3_Macro264: ;SGDONE
DW 198
DW 224
DW 205
DW 00 ;end
DW 198
DW 235
DW 232
DW 205
DW 00 ;end

Tbl3_Macro275: ;SGDONE
    DW 198
    DW 236
    DW 232
    DW 205
    DW 00 ;end

;END GEORGE 07/07/98
;END BACK
;
:GEORGE 07/08/98
:SICK

Tbl3_Macro276: ;SJ DONE ;SICK
    DW 237
    DW 168 135 40
    DW 117 41
    DW 238
    DW 00 ;end

Tbl3_Macro277: ;SG DONE
    DW 237
    DW 168 135 40
    DW 239
    DW 238
    DW 00 ;end

Tbl3_Macro278: ;SG DONE
    DW 237
    DW 168 135 40
    DW 117 41
    DW 240
    DW 00 ;end

Tbl3_Macro279: ;SG DONE
    DW 237
    DW 53 45
    DW 239
    DW 240
    DW 70 ;end

Tbl3_Macro280: ;SG DONE
    DW 237
    DW 241
    DW 00 ;end

Tbl3_Macro281: ;SG DONE
    DW 237
    DW 242
    DW 00 ;end

Tbl3_Macro282: ;SG DONE
    DW 237
    DW 243
DW 244
DW 00 ; end

; Tbl3_Macro283: ; SG DONE
DW 250
DW 117 ; 41
DW 245
DW 00 ; end

; Tbl3_Macro284: ; SG DONE
DW 250
DW 239
DW 245
DW 00 ; end

; Tbl3_Macro285: ; SG DONE
DW 250
DW 239
DW 182 ; 51
DW 00 ; end

; Tbl3_Macro286: ; SG DONE
DW 237
DW 246
DW 250
DW 00 ; end

; Tbl3_Macro287: ; SG DONE
DW 237
DW 247
DW 250
DW 00 ; end

; Tbl3_Macro288: ; SG DONE
DW 237
DW 00 ; end

; Tbl3_Macro289: ; SG DONE
DW 237
DW 248
DW 250
DW 00 ; end

; Tbl3_Macro290: ; SG DONE
DW 237
DW 249
DW 00 ; end

; Tbl3_Macro291: ; SG DONE
DW 250
DW 250
DW 00 ; end

; Tbl3_Macro292: ; SG DONE
DW 250
DW 248
DW 00 ; end
; END SICK
; END GEORGE 07/08/98
;GEORGE 07/08/98
;LIGHT
Tbl3_Macro293:
    DW 251
    DW 00 ; end RB

Tbl1_Macro294:
    DW 253
    DW 00 ; end RB

Tbl3_Macro295:
    DW 252
    DW 00 ; end RB

Tbl3_Macro296:
    DW 254
    DW 00 ; end RB

Tbl3_Macro297:
    DW 255
    DW 00 ; end RB

Tbl3_Macro298:
    DW 256
    DW 00 ; end

Tbl3_Macro299:
    DW 257
    DW 00 ; end

Tbl3_Macro300:
    DW 258
    DW 00 ; end

Tbl3_Macro301:
    DW 259
    DW 00 ; end

Tbl3_Macro302:
    DW 260
    DW 00 ; end

Tbl3_Macro303:
    DW 261
    DW 00 ; end

Tbl3_Macro304:
    DW 262
    DW 00 ; end

Tbl3_Macro305:
    DW 263
    DW 00 ; end

Tbl3_Macro306:
    DW 264
DW 00 ;end

; Tbl3_Macro307:
  DW 265
  DW 00 ;end

; Tbl3_Macro308:
  DW 266
  DW 00 ;end

; Tbl3_Macro309:
  DW 267
  DW 00 ;end

; Tbl3_Macro310:
  DW 268
  DW 00 ;end

; Tbl3_Macro311:
  DW 269
  DW 00 ;end

; Tbl3_Macro312:
  DW 270
  DW 00 ;end

; Tbl3_Macro313:
  DW 271
  DW 00 ;end

; Tbl3_Macro314:
  DW 272
  DW 00 ;end

; Tbl3_Macro315:
  DW 273
  DW 00 ;end

; Tbl3_Macro316:
  DW 274
  DW 00 ;end

; Tbl3_Macro317:
  DW 275
  DW 00 ;end

; Tbl3_Macro318:
  DW 276
  DW 00 ;end

; Tbl3_Macro319:
  DW 277
  DW 00 ;end

; Tbl3_Macro320:
  DW 278
DW 00 ; end

Tbl3_Macro321:
  DW 279
  DW 00 ; end

Tbl3_Macro322:
  DW 280
  DW 00 ; end

Tbl3_Macro323:
  DW 281
  DW 00 ; end

Tbl3_Macro324:
  DW 282
  DW 00

Tbl3_Macro325:
  DW 283
  DW 00 ; end

Tbl3_Macro326:
  DW 284
  DW 00 ; end

Tbl3_Macro327:
  DW 285
  DW 00 ; end

Tbl3_Macro328:
  DW 286
  DW 00 ; end

Tbl3_Macro329:
  DW 287
  DW 00 ; end

Tbl3_Macro330:
  DW 288
  DW 00 ; end

Tbl3_Macro331:
  DW 289
  DW 00 ; end

; END DARK
; END GEORGE 07/08/98

; SOUND

Tbl3_Macro332:
  DW 290 ; S1-A1/S9-A1/S1-A2 SOUND js
  DW 00 ; end

Tbl3_Macro333:
  DW 291 ; S2-A1/S10-A1/S2-A2 SOUND js
  DW 00 ; end
<table>
<thead>
<tr>
<th>Tbl3_Macro335:</th>
<th>DW 292</th>
<th>;S3-A1/S11-A1 SOUND js</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DW 00</td>
<td>;end</td>
</tr>
<tr>
<td>Tbl3_Macro336:</td>
<td>DW 293</td>
<td>;S4-A1/S12-A1 SOUND js</td>
</tr>
<tr>
<td></td>
<td>DW 00</td>
<td>;end</td>
</tr>
<tr>
<td></td>
<td>DW 310</td>
<td></td>
</tr>
<tr>
<td>Tbl3_Macro337:</td>
<td>DW 294</td>
<td>;S5-A1/S13-A1 SOUND (with say/m2) js</td>
</tr>
<tr>
<td></td>
<td>DW 00</td>
<td>;end</td>
</tr>
<tr>
<td>Tbl3_Macro338:</td>
<td>DW 295</td>
<td>;S6-A1/S14-A1 SOUND js</td>
</tr>
<tr>
<td></td>
<td>DW 00</td>
<td>;end</td>
</tr>
<tr>
<td>Tbl3_Macro339:</td>
<td>DW 296</td>
<td>;S7-A1/S15-A1 SOUND (with say/m2) js</td>
</tr>
<tr>
<td></td>
<td>DW 00</td>
<td>;end</td>
</tr>
<tr>
<td>Tbl3_Macro340:</td>
<td>DW 297</td>
<td>;S8-A1/S16-A1 SOUND js</td>
</tr>
<tr>
<td></td>
<td>DW 00</td>
<td>;end</td>
</tr>
<tr>
<td>Tbl3_Macro341:</td>
<td>DW 298</td>
<td>;S3-A2 SOUND js</td>
</tr>
<tr>
<td></td>
<td>DW 00</td>
<td>;end</td>
</tr>
<tr>
<td>Tbl3_Macro342:</td>
<td>DW 299</td>
<td>;S4-A2 SOUND js</td>
</tr>
<tr>
<td></td>
<td>DW 00</td>
<td>;end</td>
</tr>
<tr>
<td></td>
<td>DW 310</td>
<td></td>
</tr>
<tr>
<td>Tbl3_Macro343:</td>
<td>DW 300</td>
<td>;S5-A2 SOUND (with say/m2) js</td>
</tr>
<tr>
<td></td>
<td>DW 00</td>
<td>;end</td>
</tr>
<tr>
<td>Tbl3_Macro344:</td>
<td>DW 301</td>
<td>;S7-A2 SOUND (with say/m2) js</td>
</tr>
<tr>
<td></td>
<td>DW 00</td>
<td>;end</td>
</tr>
<tr>
<td>Tbl3_Macro345:</td>
<td>DW 302</td>
<td>;S8-A2 SOUND js</td>
</tr>
<tr>
<td></td>
<td>DW 00</td>
<td>;end</td>
</tr>
<tr>
<td>Tbl3_Macro346:</td>
<td>DW 303</td>
<td>;S3-A3 SOUND js</td>
</tr>
<tr>
<td></td>
<td>DW 00</td>
<td>;end</td>
</tr>
<tr>
<td>Tbl3_Macro347:</td>
<td>DW 304</td>
<td>;S4-A3 SOUND js</td>
</tr>
<tr>
<td></td>
<td>DW 00</td>
<td>;end</td>
</tr>
<tr>
<td></td>
<td>DW 310</td>
<td>;S7-A3 SOUND (with say/m2) js</td>
</tr>
<tr>
<td></td>
<td>DW 00</td>
<td>;end</td>
</tr>
</tbody>
</table>
Tbl3_Macro348:
  DW 306 ;S1-A4 SOUND js
  DW 00 ;end

Tbl3_Macro349:
  DW 307 ;S3-A4 SOUND js
  DW 00 ;end

Tbl3_Macro350:
  DW 308 ;S6-A4 SOUND js
  DW 00 ;end

Tbl3_Macro351:
  DW 309 ;S8-A4 SOUND js
  DW 00 ;end

;END GEORGE 07/08/98
;END SOUND

;TILT
;GEORGE 07/09/93

Tbl3_Macro352:
  DW 310 ;S1 A1 TILT/S4 A1 TILT js
  DW 00 ;end

Tbl3_Macro353:
  DW 311 ;S2 A1 TILT js
  DW 00 ;end

Tbl3_Macro354:
  DW 312 ;S3 A1 TILT js
  DW 00 ;end

Tbl3_Macro355:
  DW 313 ;S5 A1 TILT js
  DW 00 ;end

Tbl3_Macro356:
  DW 314 ;S6 A1 TILT js
  DW 00 ;end

Tbl3_Macro357:
  DW 315 ;S7 A1 TILT js
  DW 00 ;end

Tbl3_Macro358:
  DW 313 ;S8 A1 TILT js
  DW 316
  DW 00 ;end

Tbl3_Macro359:
  DW 317 ;S9 A1 TILT js
  DW 00 ;end

Tbl3_Macro360:
  DW 318 ;S10 A1 TILT js
  DW 00 ;end

Tbl3_Macro361:
DW 310 ; S11 A1 TILT js
DW 319
DW 00 ; end

Tbl3_Macro362:
DW 320 ; S12 A1 TILT js
DW 00 ; end

Tbl3_Macro363:
DW 321 ; S13 A1 TILT js
DW 00 ; end

Tbl3_Macro364:
DW 322 ; S15 A1 TILT js
DW 00 ; end

Tbl3_Macro365:
DW 323 ; S16 A1 TILT js
DW 00 ; end

Tbl3_Macro366:
DW 324 ; S1 A1 TILT js
DW 00 ; end

Tbl3_Macro367:
DW 324
DW 325 ; S2 A1 TILT js
DW 00 ; end

Tbl3_Macro368:
DW 326 ; S5 A2 TILT js
DW 00 ; end

Tbl3_Macro369:
DW 313
DW 327 ; S7 A2 TILT js
DW 00 ; end

Tbl3_Macro370:
DW 313
DW 328 ; S8 A2 TILT js
DW 00 ; end

Tbl3_Macro371:
DW 310
DW 329 ; S11 A2 TILT js
DW 00 ; end

Tbl3_Macro372:
DW 330 ; S12 A2 TILT js
DW 00 ; end

Tbl3_Macro373:
DW 313
DW 331 ; S13 A2 TILT js
DW 00 ; end

Tbl3_Macro374:
DW 332 ; S12 A2 TILT js
DW 00 ; end
; Tbl3_Macro375:
  DW 333
  DW 00 ;end
; Tbl3_Macro376:
  DW 334
  DW 00 ;end
; Tbl3_Macro377:
  DW 334
  DW 335
  DW 00 ;end
; Tbl3_Macro378:
  DW 336
  DW 00 ;end
; Tbl3_Macro379:
  DW 313
  DW 337
  DW 00 ;end
; Tbl3_Macro380:
  DW 313
  DW 338
  DW 00 ;end
; Tbl3_Macro381:
  DW 313
  DW 339
  DW 00 ;end
; Tbl3_Macro382:
  DW 317
  DW 340
  DW 00 ;end
; Tbl3_Macro383:
  DW 341
  DW 00 ;end
; Tbl4_Macro384:
  DW 310
  DW 329
  DW 342
  DW 00 ;end
; Tbl4_Macro385:
  DW 313
  DW 343
  DW 00 ;end
; Tbl4_Macro386:
  DW 313
  DW 344
  DW 00 ;end
; Tbl4_Macro387:
  DW 334
  DW 345
DW 00 ;end

; Tbl4_Macro388:
   DW 346
   DW 00 ;end

; Tbl4_Macro389:
   DW 313
   DW 347
   DW 00 ;end

; Tbl4_Macro390:
   DW 310
   DW 348
   DW 00 ;end

; Tbl4_Macro391:
   DW 313
   DW 349
   DW 00 ;end

; Tbl4_Macro392:
   DW 313
   DW 350
   DW 00 ;end

; END TILT
; END GEORGE 07/09/98

; IR
; GEORGE 07/09/98

Tbl4_Macro393:
   DW 351
   DW 00 ;end

; Tbl4_Macro394:
   DW 352 ;seq5, IR agel
   DW 00 ;end

; Tbl4_Macro395:
   DW 353 ;seq6, IR agel
   DW 354
   DW 00 ;end

; Tbl4_Macro396:
   DW 356 ;seq7 ir agel
   DW 355
   DW 00 ;end

; Tbl4_Macro397:
   DW 357 ;seq8 ir agel
   DW 00 ;end

; Tbl4_Macro398:
   DW 358 ;seq9 ir agel
   DW 00 ;end

; Tbl4_Macro399:
   DW 359 ;seq 10,360 ir agel
   DW 00 ;end
; Tbl4_Macro400:
    DW 00 ; seq12 ir age1, age2, age3
; Tbl4_Macro401:
    DW 00 ; seq13,14 ir age1
; Tbl4_Macro402:
    DW 00 ; seq15 ir age1
; Tbl4_Macro403:
    DW 00 ; seq16 ir age1
; Tbl4_Macro404:
    DW 00 ; seq1,2,3 ir age2
; Tbl4_Macro405:
    DW 00 ; seq4,5 ir age2
; Tbl4_Macro406:
    DW 00 ; seq6 ir age2
; Tbl4_Macro407:
    DW 00 ; seq7,8 ir age2
; Tbl4_Macro408:
    DW 00 ; seq9 ir age2
; Tbl4_Macro409:
    DW 00 ; seq10 ir age2
; Tbl4_Macro410:
    DW 00 ; seq11 ir age2
; Tbl4_Macro411:
    DW 00 ; seq13,14 ir age2
; Tbl4_Macro412:
    DW 00 ; seq15 ir age2
; Tbl4_Macro413:
    DW 00 ; seq16 ir age2
; Tbl4_Macro414:
    DW 00 ; seq1,2,3,4,5 ir age3
; seq15 ir age4

; END GEORGE
; END IR

; START FURBY SAYS DMH

tbl4_macro430:
    dw 50 ; TICKLE
    dw 00 ;end

; tbl4_macro431:
    dw 196 ; PET
    dw 00 ;end

; tbl4_macro432:
    dw 71 ; SOUND
    dw 00 ;end

; tbl4_macro433:
    dw 391 ; LIGHT
    dw 00 ;end

; tbl4_macro434:
    dw 196 ; soft purr
    dw 00 ;end

; tbl4_macro435:
    dw 392 ; no light
    dw 00 ;end

; tbl4_macro436:
    dw 393 ; loud sound
    dw 00 ;end

; tbl4_macro437:
    dw 115 ; burp (hide and seek)
    dw 00 ;end

; tbl4_macro438:
    dw 116 ; sigh (hide and seek)
    dw 00 ;end

; tbl4_macro439:
    ; win sound (dmh)
    dw 376
    dw 376
    dw 367
    dw 00 ;end

; END FURBY SAYS DMH

; start diagnostic tables

; start diagnostic beeps

; press key beep

A-191
; Tbl4_Macro442: ; pass beep
  DW 402
  DW 00 ; end

; Tbl4_Macro443: ; fail beep
  DW 403
  DW 00 ; end

; Tbl4_Macro444: ; speaker test tone
  DW 404
  DW 00 ; end

; Tbl4_Macro445: ; motor cal
  DW 405
  DW 00 ; end

; Tbl4_Macro446: ; feed1
  DW 406
  DW 00 ; end

; Tbl4_Macro447: ; feed2
  DW 407
  DW 00 ; end

; Tbl4_Macro448: ; light
  DW 408
  DW 00 ; end

; Tbl4_Macro449: ; sound
  DW 409
  DW 00 ; end

; Tbl4_Macro450: ; go to sleep
  DW 410
  DW 00 ; end

; end of diagnostic tables dhm

; Tbl4_Macro451:
  DW 117 ; HIDE AND SEEK SOUND DHM
  DW 00 ; end

; Tbl4_Macro452:
  DW 118 ; HIDE AND SEEK SOUND DHM
  DW 00 ; end

; Tbl4_Macro453:
  DW 399 ; delay
  DW 395 ; ME DHM
  DW 110 ; NAME "KOKO" DHM
  DW 00 ; end

; Tbl4_Macro454:
  DW 399 ; delay
  DW 395 ; ME DHM
  DW 396 ; NAME "MEME" DHM
  DW 00 ; end

; Tbl4_Macro455:
DW 399 ; delay
DW 395 ; ME
DW 112 ; NAME "E-DAY" DMH
DW 00 ; end

Tbl4_Macro456:
DW 399 ; delay
DW 395 ; ME
DW 397 ; NAME "DO-MOH" DMH
DW 00 ; end

Tbl4_Macro457:
DW 399 ; delay
DW 395 ; ME
DW 114 ; NAME "TO-TYE" DMH
DW 00 ; end

Tbl4_Macro458:
DW 399 ; delay
DW 395 ; ME
DW 117 ; NAME "BOO" DMH
DW 00 ; end

Tbl4_Macro459:
DW 399 ; delay
DW 395 ; ME
DW 131 ; NAME "A-LOH" DMH

Tbl4_Macro460:
DW 399 ; delay
DW 395 ; ME
DW 120 ; NAME "A-TAY" DMH
DW 00 ; end

Tbl4_Macro461:
DW 399 ; delay
DW 395 ; ME
DW 131 ; NAME "WAY-LOH" DMH
DW 00 ; end

Tbl4_Macro462:
DW 399 ; delay
DW 395 ; ME
DW 143 ; NAME "U-TYE"
DW 00

Tbl4_Macro463:
DW 399 ; delay
DW 395 ; ME
DW 145 ; NAME "A-LOH" DMH
DW 00 ; end

Tbl4_Macro464:
DW 399 ; delay
DW 395 ; ME
DW 152 ; NAME "KA" DMH
DW 00 ; end

Tbl4_Macro465:
DW 395 ; delay
; Tbl4_Macro466:
   DW 399   ; delay
   DW 3    ; ME
   DW 175  ; NAME "BOH-BAY" DMH
   DW 00   ; end

; Tbl4_Macro467:
   DW 399   ; delay
   DW 395  ; ME
   DW 177  ; NAME 'NAH-BAH' DMH
   DW 00   ; end

; Tbl4_Macro468:
   DW 129   ; dodle do, me love you DMH
   DW 129
   DW 151
   DW 00   ; end

; Tbl4_Macro469:
   DW 219   ; SING A SONG DMH
   DW 220
   DW 219
   DW 220
   DW 220
   DW 00   ; end

; Tbl4_Macro470:
   DW 115   ; BURE ATTACK DMH
   DW 115
   DW 115
   DW 115
   DW 115
   DW 115
   DW 115
   DW 00   ; end

; Tbl4_Macro471:
   DW 313   ; WIN SOUND DMH
   DW 338
   DW 376
   DW 00   ; end

; Tbl4_Macro472:
   DW 46    ; end

; Tbl4_Macro473:
   DW 53    ; ME DONE (DMH)
   DW 123
   DW 00   ; end

; Tbl4_Macro474:
   DW 394   ; LISTEN ME (DMH)
; Tbl4_Macro475:
  DW  411 ; end

; Tbl4_Macro476:
  DW  399 ; delay
  DW  395 ; ME
  DW  186 ; NAME "LOG-LOG" DH
  DW  00 ; end

; Tbl4_Macro477:
  DW  399 ; delay
  DW  395 ; ME
  DW  194 ; NAME "AH-MAY" DH
  DW  00 ; end

; Tbl4_Macro478:
  DW  399 ; delay
  DW  395 ; ME
  DW  201 ; NAME "LOO-LOO" DH
  DW  00 ; end

; Tbl4_Macro479:
  DW  399 ; delay
  DW  395 ; ME
  DW  208 ; ME "MAY-MAY" DH
  DW  00 ; end

; Tbl4_Macro480:
  DW  399 ; delay
  DW  395 ; ME
  DW  224 ; NAME "MAY-LAH" DH
  DW  00 ; end

; Tbl4_Macro481:
  DW  399 ; delay
  DW  395 ; ME
  DW  228 ; DAH-NOH-LAH
  DW  00 ; end

; Tbl4_Macro482:
  DW  399 ; delay
  DW  395 ; ME
  DW  398 ; NAME "TOH-LOO-LAH" DH
  DW  152 ;
  DW  00 ; end

; Tbl4_Macro483:
  DW  399 ; delay
  DW  395 ; ME
  DW  152 ; KA-DA
  DW  166
  DW  00 ; end

; Tbl4_Macro484:
  DW  399 ; delay
  DW  395 ; ME
```
DW 224 , MAY-LAH-XA
DW 152
DW 00 ; end

; Tbl4_Macro485:
  DW 4
  DW 00 ; end

; Tbl4_Macro486:
  DW 4
  DW 00 ; end

; Tbl4_Macro487:
  DW 4
  DW 00 ; end

; Tbl4_Macro488:
  DW 4
  DW 00 ; end

; Tbl4_Macro489:
  DW 4
  DW 00 ; end

; Tbl4_Macro490:
  DW 4
  DW 00 ; end

; Tbl4_Macro491:
  DW 4
  DW 00 ; end

; Tbl4_Macro492:
  DW 4
  DW 00 ; end

; Tbl4_Macro493:
  DW 4
  DW 00 ; end

; Tbl4_Macro494:
  DW 4
  DW 00 ; end

; Tbl4_Macro495:
  DW 4
  DW 00 ; end

; Tbl4_Macro496:
  DW 4
  DW 00 ; end

; Tbl4_Macro497:
  DW 4
  DW 00 ; end

; Tbl4_Macro498:
  DW 4
  DW 00 ; end
```
Tbl4_Macro499:
    DW  4
    DW  00 ; end
;
Tbl4_Macro500:
    DW  4
    DW  00 ; end
;
Tbl4_Macro501:
    DW  4
    DW  00 ; end
;
Tbl4_Macro502:
    DW  4
    DW  00 ; end
;
Tbl4_Macro503:
    DW  4
    DW  00 ; end
;
Tbl4_Macro504:
    DW  4
    DW  00 ; end
;
Tbl4_Macro505:
    DW  4
    DW  00 ; end
;
Tbl4_Macro506:
    DW  4
    DW  00 ; end
;
Tbl4_Macro507:
    DW  4
    DW  00 ; end
;
Tbl4_Macro508:
    DW  4
    DW  00 ; end
;
Tbl4_Macro509:
    DW  4
    DW  00 ; end
;
Tbl4_Macro510:
    DW  4
    DW  00 ; end
;
Tbl4_Macro511:
    DW  4
    DW  00 ; end
;
***************************************************************************
***************************************************************************
***************************************************************************
SAYSENT pointer tables (128 max per table ---- 256 tables max)

Spch_grp1:

    DW Tbl1_say000
    DW Tbl1_say001, Tbl1_say002, Tbl1_say003, Tbl1_say004, Tbl1_say005
    DW Tbl1_say006, Tbl1_say007, Tbl1_say008, Tbl1_say009, Tbl1_say010
    DW Tbl1_say011, Tbl1_say012, Tbl1_say013, Tbl1_say014, Tbl1_say015
    DW Tbl1_say016, Tbl1_say017, Tbl1_say018, Tbl1_say019, Tbl1_say020
    DW Tbl1_say021, Tbl1_say022, Tbl1_say023, Tbl1_say024, Tbl1_say025
    DW Tbl1_say026, Tbl1_say027, Tbl1_say028, Tbl1_say029, Tbl1_say030
    DW Tbl1_say031, Tbl1_say032, Tbl1_say033, Tbl1_say034, Tbl1_say035
    DW Tbl1_say036, Tbl1_say037, Tbl1_say038, Tbl1_say039, Tbl1_say040
    DW Tbl1_say041, Tbl1_say042, Tbl1_say043, Tbl1_say044, Tbl1_say045
    DW Tbl1_say046, Tbl1_say047, Tbl1_say048, Tbl1_say049, Tbl1_say050
    DW Tbl1_say051, Tbl1_say052, Tbl1_say053, Tbl1_say054, Tbl1_say055
    DW Tbl1_say056, Tbl1_say057, Tbl1_say058, Tbl1_say059, Tbl1_say060
    DW Tbl1_say061, Tbl1_say062, Tbl1_say063, Tbl1_say064, Tbl1_say065
    DW Tbl1_say066, Tbl1_say067, Tbl1_say068, Tbl1_say069, Tbl1_say070
    DW Tbl1_say071, Tbl1_say072, Tbl1_say073, Tbl1_say074, Tbl1_say075
    DW Tbl1_say076, Tbl1_say077, Tbl1_say078, Tbl1_say079, Tbl1_say080
    DW Tbl1_say081, Tbl1_say082, Tbl1_say083, Tbl1_say084, Tbl1_say085
    DW Tbl1_say086, Tbl1_say087, Tbl1_say088, Tbl1_say089, Tbl1_say090
    DW Tbl1_say091, Tbl1_say092, Tbl1_say093, Tbl1_say094, Tbl1_say095
    DW Tbl1_say096, Tbl1_say097, Tbl1_say098, Tbl1_say099
    DW Tbl1_say100, Tbl1_say101, Tbl1_say102, Tbl1_say103, Tbl1_say104
    DW Tbl1_say105, Tbl1_say106, Tbl1_say107, Tbl1_say108, Tbl1_say109
    DW Tbl1_say110, Tbl1_say111, Tbl1_say112, Tbl1_say113, Tbl1_say114
    DW Tbl1_say115, Tbl1_say116, Tbl1_say117, Tbl1_say118, Tbl1_say119
    DW Tbl1_say120, Tbl1_say121, Tbl1_say122, Tbl1_say123, Tbl1_say124
    DW Tbl1_say125, Tbl1_say126, Tbl1_say127

A-198
ALL SPEECH SAYSENT START HERE

The first line of each group is the speech speed command.
This is a number from 40 - 55 where 46 is standard speed.

The next line is PITCH control which works as follows:
Actual numeric value for TI pitch control

bit 7 set = subtract value from current course value
clr = add value to current course value
bit 6 set = select music pitch table
clr = select normal speech pitch table
bit 0-5 value to change course value (no change = 0)
8Fh ; hi voice (8f is very squeeeeeke) (BF=143)
81h ; one step higher than normal use range 81-8F (129-143)
00 ; normal voice
01 ; one step lower than normal
2Fh ; lo voice (very low) use range 01-7F (01-47)

A math routine in 'say_0' converts the value for + or -
i.e., if number is 70 then TI gets 10 (which is -10)
If number is 80 or > 80 then get sent literal as positive.

NOTE: MAX POSITIVE IS 8B
MAX NEGATIVE is 2F (80h - 2Ph or 51h)
8Bh is hi voice (8f is very squeeeeeke)
2Fh lo voice (very low)

When entering changes, 'Voice' holds the current pitch for Furby
and it is modified by adding or subtracting a pitch change:

ex: Voice+8 increases the pitch from the current voice by 8
ex: Voice-10 decreases the pitch from the current voice by 10

The next group of entries are the speech words.
The last line is the terminator of 'FF'

(BOTTOM)
1 is very fast
46 is average
255 is very slow

DB 46 (speed of speech)
DB 123 (do sound 123)
DB 43 (do sound 43)
DB FFh

PITCH PROGRAMMING RANGE:
Voice+8 (highest)
Voice-20 (lowest)

Tbll_say01:
DB 46
DB Voice
DB 163
DB FFh

; GEORGE 07/03/98
Tbll_say001: ;don start seq1 age1
DB 46 ; speech speed
DB Voice+8
DB 169,162,162,164,149 ; done 1front seq1
DB FFh ; end

Tbll_say002:
DB 52 ;speech speed
DB Voice+8 ;system pitch setting
DB 117,59 ;DONE 1FRONT SEQ2 age1
DB FFH ;end

Tbl1_say003:
DB 46 ;speech speed
DB Voice-4 ;system pitch setting
DB 118 ;1front seq3 - seq4-part1-SEQ7-PART2
DB FFH ;end

Tbl1_say004:
DB 46 ;speech speed
DB Voice ;system pitch setting
DB 62.22,85 ;1front seq3 part2
DB FFH ;end

Tbl1_say005:
DB 50 ;speech speed
DB Voice+8 ;system pitch setting
DB 58.39 ;1front seq4 part 2
DB FFH ;end

Tbl1_say006:
DB 46 ;speech speed
DB Voice ;pitch control
DB 162,162,99,117 ;seq5 age1 front part of seq7
DB FFH ;end

Tbl1_say007:
DB 55 ;speech speed
DB Voice+8 ;system pitch setting
DB 156 ;seq6 age1 front back part
DB FFH ;end

Tbl1_say008:
DB 46 ;speech speed
DB Voice ;pitch control
DB 162,162,99,10,39 ;SEQ7 FRONT AGE1 ADD SAY 003
DB FFH ;end

Tbl1_say009:
DB 46 ;speech speed
DB Voice ;system pitch setting
DB 99,39,165 ;SEQ8 FRONT AGE1
DB FFH ;end

Tbl1_say010:
DB 46 ;speech speed
DB Voice ;system pitch setting
DB 98 ;seq9 FRONT AGE1
DB FFH ;end

Tbl1_say011:
DB 30 ;speech speed
DB Voice+8 ;system pitch setting
DB 96,165,165,129,149 ;seq10 FRONT AGE1 ADD SAY20
DB FFH ;end

Tbl1_say012:
<table>
<thead>
<tr>
<th>Function</th>
<th>Speed</th>
<th>Pitch Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tbll_say022</td>
<td>56</td>
<td>156 SEQ4 AGE2 FRONT</td>
</tr>
<tr>
<td>Tbll_say023</td>
<td>46</td>
<td>6,162,22 SEQ4 AGE2 FRONT</td>
</tr>
<tr>
<td>Tbll_say024</td>
<td>46</td>
<td>117,81,27 SEQ5 AGE2 FRONT</td>
</tr>
<tr>
<td>Tbll_say025</td>
<td>46</td>
<td>99 SEQ7 AGE2 FRONT PART 1</td>
</tr>
<tr>
<td>Tbll_say026</td>
<td>46</td>
<td>60,39,117 SEQ7 AGE2 FRONT PART 2</td>
</tr>
<tr>
<td>Tbll_say027</td>
<td>46</td>
<td>145 SEQ8 AGE2 FRONT say45(2)+22</td>
</tr>
<tr>
<td>Tbll_say028</td>
<td>46</td>
<td>149,162,164,149 FRONT SEQ9 AGE3</td>
</tr>
<tr>
<td>Tbll_say029</td>
<td>60</td>
<td>96,163,163,139 SEQ10 FRONT AGE 2 ADD 46</td>
</tr>
<tr>
<td>Tbll_say030</td>
<td>60</td>
<td>19,63 SEQ11 FRONT AGE 2</td>
</tr>
</tbody>
</table>
Tb11_say032:
DB 46 ;speech speed
DB Voice*7 ;system pitch setting
DB 128,117 ;SEQ12 FRONT AGE2 ADD 20
DB FFH ;end

Tb11_say033:
DB 56 ;speech speed
DB Voice*7 ;system pitch setting
DB 99,55,162,28 ;SEQ14 FRONT AGE2
DB FFH ;end

Tb11_say034:
DB 46 ;speech speed
DB Voice*6 ;system pitch setting
DB 136,34 ;SEQ15 FRONT AGE2 ADD 20
DB FFH ;end

Tb11_say035:
DB 56 ;speech speed
DB Voice*6 ;system pitch setting
DB 35,162,162,93,133 ;SEQ16 FRONT AGE2 ADD 20 TO

BE GGINING
DB FFH ;end

Tb11_say036:
DB 50 ;speech speed
DB Voice*3 ;system pitch setting
DB 162,7 ;SEQ1 FRONT AGE3 ADD 20
DB FFH ;end

Tb11_say037:
DB 46 ;speech speed
DB Voice*3 ;system pitch setting
DB 51,77,51 ;SEQ2 FRONT AGE3
DB FFH ;end

Tb11_say038:
DB 46 ;speech speed
DB Voice*8 ;system pitch setting
DB 1,1 ;SEQ3 FRONT AGE3 ADD 29
DB FFH ;end

Tb11_say039:
DB 50 ;speech speed
DB Voice*6 ;system pitch setting
DB 162,14,77 ;SEQ4 FRONT AGE4 ADD 41
DB FFH ;end

;ERROR
Tb11_say040:
DB 46 ;speech speed
DB Voice*6 ;system pitch setting
DB FFH ;end
Tbll_say040:
DB  46 ;speech speed
    DB  Voice ;system pitch setting
    DB  99,35,47,58 ;SEQ6 FRONT AGE3
    DB  FFH ;end

Tbll_say041:
DB  46 ;speech speed
    DB  Voice ;system pitch setting
    DB  99,60,77,23 ;SEQ7 FRONT AGE3 ADD 22
    DB  FFH ;end

Tbll_say042:
DB  46 ;speech speed
    DB  Voice ;system pitch setting
    DB  99,145 ;SEQ8 FRONT AGE3 ADD 22
    DB  FFH ;end
    ;ERROR

Tbll_say043:
DB  30 ;speech speed
    DB  Voice*8 ;system pitch setting
    DB  96,165,165,165,129,149 ;SEQ10 FRONT AGE3 ADD 20
    DB  FFH ;end

Tbll_say044:
DB  50 ;speech speed
    DB  Voice*4 ;system pitch setting
    DB  145 ;SEQ11 FRONT AGE3
    DB  FFH ;end

Tbll_say045:
DB  46 ;speech speed
    DB  Voice ;system pitch setting
    DB  119,77 ;SEQ12 FRONT AGE3 (HEEY,TICKLE ME) ADD20
    DB  FFH ;end

Tbll_say046:
DB  46 ;speech speed
    DB  Voice ;system pitch setting
    DB  128 ;SEQ13 FRONT AGE3 (HARRY,HARRY) ADD10
    DB  FFH ;end

Tbll_say047:
DB  46 ;speech speed
    DB  Voice ;system pitch setting
    DB  136,117 ;SEQ14 FRONT AGE3 (RASPBERRY, he he he) ADD20
    DB  FFH ;end

Tbll_say048:
DB  46 ;speech speed
Tbll_say049:
DB 56 ;speech speed
DB Voice+6 ;system pitch setting
DB 81,133 ;SEQ16 (U-NYE QUICK KISS) FRONT AGE3 ADD20
DB FFH ;end

Tbll_say050:
DB 46 ;speech speed
DB Voice ;system pitch setting
DB 77 ;SEQ2 (TICKLE) FRONT AGE4
DB FFH ;end

Tbll_say051:
DB 46 ;speech speed
DB Voice+6 ;system pitch setting
DB 1 ;SEQ2 (AGAIN) FRONT AGE4
DB FFH ;end

Tbll_say052:
DB 46 ;speech speed
DB Voice+6 ;system pitch setting
DB 52 ;SEQ3 (YOU) FRONT AGE4
DB FFH ;end

Tbll_say053:
DB 46 ;speech speed
DB Voice+6 ;system pitch setting
DB 47 ;SEQ4 (LOVE) FRONT AGE4
DB FFH ;end

Tbll_say054:
DB 46 ;speech speed
DB Voice+6 ;system pitch setting
DB 117 ;SEQ5 (HE HE HE) FRONT AGE4
DB FFH ;end

Tbll_say055:
DB 46 ;speech speed
DB Voice+8 ;system pitch setting
DB 8,27 ;SEQ5 (BIG FUN) FRONT AGE4 ADD26
DB FFH ;end

Tbll_say056:
DB 46 ;speech speed
DB Voice+6 ;system pitch setting
DB 60 ;SEQ8 (NO) FRONT AGE4
DB FFH ;end

Tbll_say057:
DB 46 ;speech speed
DB Voice+6 ;system pitch setting
DB 60 ;SEQ8 (ND) FRONT AGE4
DB FFH ;end

Tbll_say058:
DB 46 ;speech speed
DB Voice ; system pitch setting
DB 68 ; SEQ8 (PLEASE) FRONT AGE4
DB FFH ; end

Tbll_say059:
DB 46 ; speech speed
DB Voice+8 ; system pitch setting
DB 119 ; SEQ9 (HEEY) FRONT AGE4 ADD71
DB FFH ; end

Tbll_say060:
DB 46 ; speech speed
DB Voice ; system pitch setting
DB 66 ; SEQ14 (PARTY) FRONT AGE4
DB FFH ; end

Tbll_say061:
DB 46 ; speech speed
DB Voice ; system pitch setting
DB 108 ; SEQ15 (WA WA WA) FRONT AGE4 ADD 22
DB FFH ; end

; END GEORGE 07/03/98

; GEORGE 07/04/98
; START SAY FORTUNE

Tbll_say062:
DB 46 ; speech speed
DB Voice+6 ; system pitch setting
DB 3 ; FORTUNE TELL (ASK)
DB FFH ; end

Tbll_say063:
DB 46 ; speech speed
DB Voice ; system pitch setting
DB 92 ; FORTUNE TELL (YES)
DB FFH ; end

Tbll_say064:
DB 46 ; speech speed
DB Voice ; system pitch setting
DB 8 ; FORTUNE TELL (BIG)
DB FFH ; end

Tbll_say065:
DB 46 ; speech speed
DB Voice+6 ; system pitch setting
DB 84,8 ; FORTUNE TELL (VERY, BIG)
DB FFH ; end

Tbll_say066:
DB 100 ; speech speed
DB Voice ; system pitch setting
DB 162,70 ; FORTUNE TELL (SEE YES)
DB FFH ; end

Tbll_say067:
DB ; speech speed
DB Voice+4 ; system pitch setting
DB 157,162,157 ; Fortune tell (SLOW WHINE)
DB FFH ; end

END GEORGE 07/03/98
Tbll_say087:
  DB  46  ;speech speed
  DB  Voice ;system pitch setting
  DB  101 ;SEQ5 HANGING (HUMMMHHH)
  DB  FFH ;end

Tbll_say088:
  DB  46  ;speech speed
  DB  Voice ;system pitch setting
  DB  111 ;SEQ5 HANGING (BO DAH WA LO)
  DB  FFH ;end

Tbll_say089:
  DB  46  ;speech speed
  DB  Voice*! ;system pitch setting
  DB  143,163 ;SEQ6 HANGING (SNORE)
  DB  FFH ;end

Tbll_say090:
  DB  46  ;speech speed
  DB  Voice*4 ;system pitch setting
  DB  148 ;SEQ6 HANGING (SHOUT)
  DB  FFH ;end

Tbll_say091:
  DB  46  ;speech speed
  DB  Voice*7 ;system pitch setting
  DB  63,75 ;SEQ6 HANGING (OK,PAH)
  DB  FFH ;end

Tbll_say092:
  DB  46  ;speech speed
  DB  Voice*6 ;system pitch setting
  DB  62 ;SEQ6 HANGING (U-TYE)
  DB  FFH ;end

Tbll_say093:
  DB  60  ;speech speed
  DB  Voice*8 ;system pitch setting
  DB  144 ;SEQ7 HANGING (SOFTER)
  DB  FFH ;end

Tbll_say094:
  DB  46  ;speech speed
  DB  Voice*4 ;system pitch setting
  DB  144 ;SEQ7 HANGING (SOFTER)
  DB  FFH ;end

Tbll_say095:
  DB  46  ;speech speed
  DB  Voice*8 ;system pitch setting
  DB  124,162 ;SEQ8 HANGING (KITTY KITTY)
  DB  FFH ;end

Tbll_say096:
  DB  56  ;system pitch setting
  DB  112 ;SEQ9 HANGING (DO BE DOBE DO)
  DB  FFH ;end
Tbll_say097:
             DB    60 ; speech speed
             DB    Voice*7 ; system pitch setting
             DB    161,164,164,161 ; SEQ10 HANGING (YAWN)
             DB    FFH ; end

Tbll_say098:
             DB    100 ; speech speed
             DB    Voice*6 ; system pitch setting
             DB    140 ; SEQ11 AND SEQ12 HANGING (SIGH)
             DB    FFH ; end

Tbll_say099:
             DB    46 ; speech speed
             DB    Voice*8 ; system pitch setting
             DB    100 ; SEQ13 SEQ14 HANGING (HAA)
             DB    FFH ; end

Tbll_say100:
             DB    46 ; speech speed
             DB    Voice ; system pitch setting
             DB    116 ; SEQ14 HANGING (HEEY)
             DB    FFH ; end

Tbll_say101:
             DB    46 ; speech speed
             DB    Voice ; system pitch setting
             DB    132,165,132 ; SEQ15 HANGING (PHONE) ADD20
             DB    FFH ; end

Tbll_say102:
             DB    46 ; speech speed
             DB    Voice ; system pitch setting
             DB    165,165,165,165 ; SEQ16 HANGING (PAUSE) ADD20
             DB    FFH ; end

Tbll_say103:
             DB    46 ; speech speed
             DB    Voice*5 ; system pitch setting
             DB    63 ; SEQ6 HANGING (UP)
             DB    FFH ; end

Tbll_say104:
             DB    46 ; speech speed
             DB    Voice ; system pitch setting
             DB    52 ; SEQ6 HANGING AGE3 (ME)
             DB    FFH ; end

Tbll_say105:
             DB    46 ; speech speed
             DB    Voice ; system pitch setting
             DB    63 ; SEQ6 HANGING AGE3 (OK)
             DB    FFH ; end

Tbll_say106:
             DB    46 ; speech speed
             DB    Voice ; system pitch setting
             DB    13 ; SEQ5 HANGING AGE3 AND 4
             DB    FFH ; end
; END HANGOUT
;
;
Tbl1_say107:
  DB  46 ;speech speed
  DB  Voice ;system pitch setting
  DB  165,165 ;Fortune delay
  DB  FFH ;end
;
;END GEORGE 07/04/98
;START FEED
;GEORGE 07/05/98

;------------------------------------------=START FEED

; spch_grp2 was here
; Saysent groups forTbl 2

;STARTS AT 128

Tbl1_say108:
  DB  100 ;speech speed
  DB  Voice ;system pitch setting
  DB  166 ;SEQ1 FEED AGE1 (ULMAM)
  DB  FFH ;end

;NOT USED

Tbl12_say129:
  DB  46 ;speech speed
  DB  Voice+3 ;system pitch setting
  DB  FFH ;end

Tbl11_say109:
  DB  100 ;speech speed
  DB  Voice ;system pitch setting
  DB  167,167 ;SEQ1 FEED AGE1 (AAAAH)
  DB  FFH ;end

Tbl11_say110:
  DB  56 ;speech speed
  DB  Voice+3 ;system pitch setting
  DB  39 ;SEQ2 FEED AGE1 (KOH-KOH)
  DB  FFH ;end

Tbl11_say111:
  DB  56 ;speech speed
  DB  Voice+7 ;system pitch setting
  DB  55 ;SEQ2 FEED AGE1 (MEE MEE)
  DB  FFH ;end

Tbl11_say112:
  DB  50 ;speech speed
  DB  Voice ;system pitch setting
  DB  25 ;SEQ2 FEED AGE1 (E-DAY)
  DB  FFH ;end

A-213
Tbl1_say113:
  DB  58 ;speech speed
  DB Voice+7 ;system pitch setting
  DB  23 ;SEQ2 FEED AGE1 (DO MOH)
  DB FFH ;end

Tbl1_say114:
  DB  58 ;speech speed
  DB Voice ;system pitch setting
  DB  79 ;TOH-DYE
  DB FFH ;end

Tbl1_say115:
  DB  46 ;speech speed
  DB Voice ;system pitch setting
  DB  97 ;BURP
  DB FFH ;end

Tbl1_say116:
  DB  46 ;speech speed
  DB Voice ;system pitch setting
  DB 140 ;SIGH
  DB FFH ;end

Tbl1_say117:
  DB  46 ;speech speed
  DB Voice ;system pitch setting
  DB  10 ;BOO
  DB FFH ;end

Tbl1_say118:
  DB  46 ;speech speed
  DB Voice ;system pitch setting
  DB  85 ;WAH
  DB FFH ;end

Tbl1_say119:
  DB  60 ;speech speed
  DB Voice+8 ;system pitch setting
  DB  80 ;TOH-LOO
  DB FFH ;end

Tbl1_say120:
  DB  46 ;speech speed
  DB Voice+8 ;system pitch setting
  DB  7 ;A TAY
  DB FFH ;end

Tbl1_say121:
  DB  46 ;speech speed
  DB Voice ;system pitch setting
  DB 33 ;SEQ1 FEED AGE2 HUNGRY
  DB FFH ;end

143 SAME AS TBL1_SAY072

Tbl1_say143:
  DB  46 ;speech speed
  DB Voice ;system pitch setting
  DB  28 ;SEQ2 FEED AGE3 (GOOD)
  DB FFH ;end
DB 82
DB FFH ; end

; PASS
Tbl1_say127: ; SG DONE
DB 55 ; speech speed
DB Voice ; system pitch setting
DB 164,83
DB FFH ; end

; Tbl2_say128: ; SG DONE
DB 55 ; speech speed
DB Voice ; system pitch setting
DB 63,52
DB FFH ; end

; Tbl2_say129: ; SG DONE
DB 40 ; speech speed
DB Voice ; system pitch setting
DB 163,119
DB FFH ; end

; TBL1_SAY55
; Tbl1_say8: ; SG DONE
: DB 46 ; speech speed
: DB Voice+8 ; system pitch setting
: DB 117
: DB FFH ; end

; Tbl2_say130: ; SG DONE
DB 55 ; speech speed
DB Voice+2 ; system pitch setting
DB 63
DB FFH ; end

; Tbl2_say131: ; SG DONE
DB 46 ; speech speed
DB Voice+6 ; system pitch setting
DB 86
DB FFH ; end

; Tbl2_say132: ; SG DONE
DB 46 ; speech speed
DB Voice ; system pitch setting
DB 79
DB FFH ; end

; TBL1_SAY112
; Tbl1_say12: ; SG DONE
: DB 46 ; speech speed
: DB Voice ; system pitch setting
: DB 20
: DB FFH ; end

; Tbl2_say133: ; SG DONE
DB 46 ; speech speed
DB Voice ; system pitch setting
DB 72
DB FFH ; end

; Tbl2_say134: ; SG DONE
DB 55 ; speech speed
DB Voice+3 ; system pitch setting
DB 14
DB FFH ;end

; Tbl2_say145: ;SG DONE
DB 46 ;speech speed
DB Voice ;pitch control
DB 6
DB FFH ;end

; Tbl2_say146: ;SG DONE
DB 46 ;speech speed
DB Voice ;system pitch setting
DB 83
DB FFH ;end

; Tbl2_say147: ;SG DONE
DB 70 ;speech speed
DB Voice ;pitch control
DB 76
DB FFH ;end

; Tbl2_say148: ;SG DONE
DB 60 ;speech speed
DB Voice ;system pitch setting
DB 37
DB FFH ;end

; TBL1_SAY53
; Tbl1_say29: ;SG DONE
; DB 46 ;speech speed
; DB Voice ;system pitch setting
; DB 52
; DB FFH ;end

; Tbl2_say149: ;SG DONE
DB 30 ;speech speed
DB Voice ;system pitch setting
DB 47
DB FFH ;end

; Tbl2_say150: ;SG DONE
DB 60 ;speech speed
DB Voice ;system pitch setting
DB 81
DB FFH ;end

; Tbl2_say151: ;SG DONE
DB 55 ;speech speed
DB Voice ;system pitch setting
DB 53
DB FFH ;end

; Tbl2_say152: ;SG DONE
DB 40 ;speech speed
DB Voice ;system pitch setting
DB 35
DB FFH ;end

; Tbl2_say153: ;SG DONE
DB 46 ;speech speed
DB Voice ;system pitch setting
DB 39
DB FFH ;end

; Tbl2_say154: ;SG DONE
DB 55 ;speech speed
DB Voice ;system pitch setting
DB FFH ;end

; Tbl2_say155: ;SG DONE
DB 46 ;speech speed
DB Voice ;system pitch setting
DB 72
DB FFH ;end

; Tbl2_say156: ;SG DONE
DB 60 ;speech speed
DB Voice ;system pitch setting
DB 1
DB FFH ;end

; TBL1_SAY53
; Tbl1_say38: ;SG DONE
; DB 46 ;speech speed
; DB Voice ;system pitch setting
; DB 52
; DB FFH ;end
; END GEORGE 07/06/98
; END WAKE
;
; GEORGE 07/06/98
; HUNGER
Tbl2_say157: ;SG DONE ;HUNGER
DB 65 ;speech speed
DB Voice ;system pitch setting
DB 68
DB FFH ;end

; Tbl2_say158: ;SG DONE
DB 75 ;speech speed
DB Voice ;system pitch setting
DB 23
DB FFH ;end

; Tbl2_say159: ;SG DONE
DB 40 ;speech speed
DB Voice ;system pitch setting
DB 7
DB FFH ;end

; Tbl2_say160: ;SG DONE
DB 55 ;speech speed
DB Voice ;system pitch setting
DB 33
DB FFH ;end

; Tbl2_say161: ;SG DONE
DB 75 ;speech speed
DB  voice ; system pitch setting
DB  55
DB  FFH ; end

Tbl2_say162: ; SG DONE
DB  40 ; speech speed
DB  Voice -15 ; system pitch setting
DB  64
DB  FFH ; end

Tbl2_say163: ; SG DONE
DB  65 ; speech speed
DB  Voice +8 ; system pitch setting
DB  157
DB  FFH ; end

Tbl2_say164: ; SG DONE
DB  55 ; speech speed
DB  Voice +6 ; system pitch setting
DB  119
DB  FFH ; end

Tbl2_say165: ; SG DONE
DB  65 ; speech speed
DB  Voice +8 ; system pitch setting
DB  85
DB  FFH ; end

Tbl2_say166: ; SG DONE
DB  55 ; speech speed
DB  Voice ; system pitch setting
DB  14
DB  FFH ; end

Tbl2_say167: ; SG DONE
DB  40 ; speech speed
DB  Voice ; system pitch setting
DB  8
DB  FFH ; end

Tbl2_say168: ; SG DONE ; SAME AS SAY135 WITH DIFFERENT MOTOR
DB  46 ; speech speed
DB  Voice ; system pitch setting
DB  35
DB  FFH ; end

; END GEORGE 07/06/98
; END HUNGER

; /
; GEORGE 07/07/96
; INVERT
; WAS68
Tbl2_say169: ; SG DONE ; INVERT
DB  85 ; speech speed
DB  Voice ; system pitch setting
DB  36
DB  FFH ; end

A-220
Tbl2_sayl70: ;SG DONE
DB 55 ;speech speed
DB Voice+8 ;system pitch setting
DB 94
DB FFH ;end

Tbl2_sayl71: ;SG DONE
DB 70 ;speech speed
DB Voice+8 ;system pitch setting
DB 158
DB FFH ;end

Tbl2_sayl72: ;SG DONE
DB 55 ;speech speed
DB Voice+8 ;system pitch setting
DB 148
DB FFH ;end

Tbl2_sayl73: ;SG DONE
DB 100 ;speech speed
DB Voice+8 ;system pitch setting
DB 97
DB FFH ;end

Tbl2_sayl74: ;SG DONE
DB 50 ;speech speed
DB Voice+5 ;system pitch setting
DB 6
DB FFH ;end

Tbl2_sayl75: ;SG DONE
DB 55 ;speech speed
DB Voice+5 ;system pitch setting
DB 9
DB FFH ;end

Tbl2_sayl76: ;SG DONE
DB 50 ;speech speed
DB Voice-10 ;system pitch setting
DB 54
DB FFH ;end

Tbl2_sayl77: ;SG DONE
DB 70 ;speech speed
DB Voice-6 ;system pitch setting
DB 57
DB FFH ;end

Tbl2_sayl78: ;SG DONE
DB 4 ;speech speed
DB Voice ;system pitch setting
DB 24
DB FFH ;end

Tbl2_sayl79: ;SG DONE
DB 55 ;speech speed
DB Voice-5 ;system pitch setting
DB 10
DB FFH ;end
Tbl2_say180: ;SG DONE
DB 65  ;speech speed
DB Voice-5  ;system pitch setting
DB 80
DB FFH ;end

Tbl2_say131: ;SG DONE
DB 55  ;speech speed
DB Voice-10  ;system pitch setting
DB 60
DB FFH ;end

Tbl2_say182: ;SG DONE
DB 55  ;speech speed
DB Voice-10  ;system pitch setting
DB 43
DB FFH ;end

Tbl2_say183: ;SG DONE
DB 75  ;speech speed
DB Voice-8  ;system pitch setting
DB 90
DB FFH ;end

Tbl2_say184: ;SG DONE
DB 75  ;speech speed
DB Voice-4  ;system pitch setting
DB 29
DB FFH ;end

Tbl2_say185: ;SG DONE
DB 55  ;speech speed
DB Voice-5  ;system pitch setting
DB 34
DB FFH ;end

Tbl2_say186: ;SG DONE
DB 65  ;speech speed
DB Voice-2  ;system pitch setting
DB 45
DB FFH ;end

Tbl2_say187: ;SG DONE
DB 65  ;speech speed
DB Voice-7  ;system pitch setting
DB 39
DB FFH ;end

Tbl2_say188: ;SG DONE
DB 35  ;speech speed
DB Voice  ;system pitch setting
DB 130
DB FFH ;end

Tbl2_say153: ;SG DONE

DB 75  ;speech speed
DB Voice  ;system pitch setting
DB 23
DB FFH ;end
Tbl2_say189:  ;SG DONE
    DB  55  ;speech speed
    DB   Voice ;system pitch setting
    DB   1
    DB  FFH  ;end
;
Tbl2_say190:
    DB   100 ;speech speed
    DB   Voice ;system pitch setting
    DB   97
    DB  FFH  ;end
;
Tbl2_say191:
    DB   100 ;speech speed
    DB   Voice-10 ;system pitch setting
    DB   97
    DB  FFH  ;end
;
Tbl2_say192:
    DB   100 ;speech speed
    DB   Voice-20 ;system pitch setting
    DB   97
    DB  FFH  ;end
;
END GEORGE 07/07/98
END INVERT

;start at 202
Tbl2_say193:  ;SG DONE ;BACKSG
    DB   70  ;speech speed
    DB   Voice ;system pitch setting
    DB   153
    DB  FFH  ;end
;
Tbl2_say194:  ;SG DONE
    DB   75  ;speech speed
    DB   Voice ;system pitch setting
    DB   2
    DB  FFH  ;end
;
Tbl2_say195:  ;SG DONE
    DB   55  ;speech speed
    DB   Voice ;system pitch setting
    DB   39
    DB  FFH  ;end
;
Tbl2_say196:  ;SG DONE
    DB   65  ;speech speed
    DB   Voice*4 ;system pitch setting
    DB   67 ; PET
    DB  FFH  ;end
;
Tbl2_say197:  ;SG DONE
    DB   75  ;speech speed
    DB   Voice*5 ;system pitch setting
    DB   1
    DB  FFH  ;end
;
Tbl2_say198:  ;SG DONE
    DB   55  ;speech speed
    DB   Voice-10 ;system pitch setting
DE 146
DB FFH ;end

Tbl2_say199: ;SG DONE
DB 55 ;speech speed
DB Voice-5 ;system pitch setting
DB 35
DB FFH ;end

Tbl2_say200: ;SG DONE
DB 80 ;speech speed
DB Voice-5 ;system pitch setting
DB 55
DB FFH ;end

Tbl2_say201: ;SG DONE
DB 70 ;speech speed
DB Voice-5 ;system pitch setting
DB 62
DB FFH ;end

Tbl2_say202: ;SG DONE
DB 80 ;speech speed
DB Voice-5 ;system pitch setting
DB 84
DB FFH ;end

Tbl2_say203: ;SG DONE
DB 70 ;speech speed
DB Voice ;system pitch setting
DB 29
DB FFH ;end

Tbl2_say204: ;SG DONE
DB 55 ;speech speed
DB Voice ;system pitch setting
DB 152
DB FFH ;end

Tbl2_say205: ;SG DONE
DB 65 ;speech speed
DB Voice-5 ;system pitch setting
DB 52
DB FFH ;end

Tbl2_say206: ;SG DONE
DB 65 ;speech speed
DB Voice-2 ;system pitch setting
DB 47
DB FFH ;end

Tbl2_say207: ;SG DONE
DB 65 ; speech speed
DB Voice-3 ; system pitch setting
DB 81
DB FFH ; end

Tbl2_say208: ; SG DONE
DB 70 ; speech speed
DB Voice+6 ; system pitch setting
DB 48
DB FFH ; end

Tbl2_say209: ; SG DONE
DB 70 ; speech speed
DB Voice+3 ; system pitch setting
DB 161
DB FFH ; end

Tbl2_say210: ; SG DONE
DB 55 ; speech speed
DB Voice ; system pitch setting
DB 15
DB FFH ; end

Tbl2_say211: ; SG DONE
DB 45 ; speech speed
DB Voice-10 ; system pitch setting
DB 8
DB FFH ; end

Tbl2_say212: ; SG DONE
DB 55 ; speech speed
DB Voice-10 ; system pitch setting
DB 42
DB FFH ; end

Tbl2_say213: ; SG DONE
DB 65 ; speech speed
DB Voice-15 ; system pitch setting
DB 57
DB FFH ; end

Tbl2_say214: ; SG DONE
DB 50 ; speech speed
DB Voice ; system pitch setting
DB 75
DB FFH ; end

Tbl2_say215: ; SG DONE
DB 55 ; speech speed
DB Voice ; system pitch setting
DB 101
DB FFH ; end

Tbl2_say216: ; SG DONE
DB 70 ; speech speed
DB Voice-3 ; system pitch setting
DB 49
DB FFH ; end

Tbl2_say217: ; SG DONE
DB 75 ;speech speed
DB Voice+5 ;system pitch setting
DB 86
DB FFH ;end
;
Tbl2_say218: ;SG DONE
DB 55 ;speech speed
DB Voice ;system pitch setting
DB 72
DB FFH ;end
;
Tbl2_say219: ;SG DONE
DB 55 ;speech speed
DB Voice+5 ;system pitch setting
DB 150
DB FFH ;end
;
Tbl2_say220: ;SG DONE
DB 55 ;speech speed
DB Voice+5 ;system pitch setting
DB 151
DB FFH ;end
;
Tbl2_say221: ;SG DONE
DB 55 ;speech speed
DB Voice ;system pitch setting
DB 97
DB FFH ;end
;
Tbl2_say222: ;SG DONE
DB 70 ;speech speed
DB Voice ;system pitch setting
DB 165,149
DB FFH ;end
;
Tbl2_say223: ;SG DONE
DB 55 ;speech speed
DB Voice ;system pitch setting
DB 129
DB FFH ;end
;
Tbl2_say224: ;SG DONE
DB 75 ;speech speed
DB Voice-4 ;system pitch setting
DB 50
DB FFH ;end
;
Tbl2_say225: ;SG DONE
DB 55 ;speech speed
DB Voice+5 ;system pitch setting
DB 32
DB FFH ;end
;
Tbl2_say226: ;SG DONE
DB 55 ;speech speed
DB Voice+5 ;system pitch setting
DB 165,140
DB FFH ;end
;
Tbl2_say227: ;SG DONE
DB 65 ;speech speed
DB Voice ;system pitch setting
DB 144
DB FFH ;end

Tbl2_say228: ;SG DONE
DB 85 ;speech speed
DB Voice ;system pitch setting
DB 18
DB FFH ;end

Tbl2_say229: ;SG DONE
DB 50 ;speech speed
DB Voice+8 ;system pitch setting
DB 118
DB FFH ;end

Tbl2_say230: ;SG DONE
DB 65 ;speech speed
DB Voice ;system pitch setting
DB 66
DB FFH ;end

Tbl2_say231: ;SG DONE
DB 70 ;speech speed
DB Voice+8 ;system pitch setting
DB 87
DB FFH ;end

Tbl2_say232: ;SG DONE
DB 60 ;speech speed
DB Voice+8 ;system pitch setting
DB 71
DB FFH ;end

Tbl2_say233: ;SG DONE
DB 55 ;speech speed
DB Voice ;system pitch setting
DB 93
DB FFH ;end

Tbl2_say234: ;SG DONE
DB 46 ;speech speed
DB Voice-20 ;system pitch setting
DB 161
DB FFH ;end

Tbl2_say235:
DB 70 ;speech speed
DB Voice ;system pitch setting
DB 81
DB FFH ;end

Tbl2_say236:
DB 70 ;speech speed
DB Voice ;system pitch setting
DB 93
DB FFH ;end
; Sick
; George 07/08/98
; start at 39
Tbl2_say237: ; SG DONE ; SICK1
  DB 55 ; speech speed
  DB Voice+5 ; system pitch setting
  DB 165,141
  DB FFH ; end
Tbl2_say235
Tbl2_say40: ; SG DONE
  DB 46 ; speech speed
  DB Voice ; system pitch setting
  DB 35
  DB FFH ; end
Tbl2_say117
Tbl2_say41: ; SG DONE
  DB 46 ; speech speed
  DB Voice ; system pitch setting
  DB 10
  DB FFH ; end
Tbl2_say238: ; SG DONE
  DB 46 ; speech speed
  DB Voice ; system pitch setting
  DB 40
  DB FFH ; end
Tbl2_say239: ; SG DONE
  DB 46 ; speech speed
  DB Voice-5 ; system pitch setting
  DB 60
  DB FFH ; end
Tbl2_say240: ; SG DONE
  DB 50 ; speech speed
  DB Voice ; system pitch setting
  DB 30
  DB FFH ; end
Tbl2_say53
Tbl2_say45: ; SG DONE
  DB 46 ; speech speed
  DB Voice ; system pitch setting
  DB 52
  DB FFH ; end
Tbl2_say241: ; SG DONE
  DB 70 ; speech speed
  DB Voice-8 ; system pitch setting
  DB 17
  DB FFH ; end
Tbl2_say242: ; SG DONE
  DB 80 ; speech speed
  DB Voice-10 ; system pitch setting
  DB 46
  DB FFH ; end
Tbl2_say243: ; SG DONE
  DB 55 ; speech speed
  DB Voice-8 ; system pitch setting
DB 8
DB FFH ;end

: Tbl2_say244: ;SG DONE
DB 40 ;speech speed
DB Voice-8 ;system pitch setting
DB 73
db FFH ; end

: Tbl2_say245: ; SG DONE
DB 75 ; speech speed
DB Voice-5 ; system pitch setting
DB 80
DB FFH ; end

: Tbl2_say182

: Tbl1_say51: ; SG DONE
DB 55 ; speech speed
DB Voice-10 ; system pitch setting
DB 43
DB FFH ; end

: Tbl2_say246: ; SG DONE
DB 70 ; speech speed
DB Voice ; system pitch setting
DB 9
DB FFH ; end

: Tbl2_say247: ; SG DONE
DB 60 ; speech speed
DB Voice-12 ; system pitch setting
DB 90,165
DB FFH ; end

: Tbl2_say248: ; SG DONE
DB 100 ; speech speed
DB Voice ; system pitch setting
DB 140
DB FFH ; end

: Tbl2_say249: ; SG DONE
DB 40 ; speech speed
DB Voice-20 ; system pitch setting
DB 162,129
DB FFH ; end

: Tbl2_say250: ; SG DONE
DB 100 ; speech speed
DB Voice ; system pitch setting
DB 142
DB FFH ; end

: END GEORGE 07/06/98
: END SICK

: LIGHT
: GEORGE 07/08/98
: starts at 2
Tb12_say251:
DB 40 ; speech speed
DONE RB
BEGIN LIGHT

D. [BRIGHTER]
<table>
<thead>
<tr>
<th>Table</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tbl1_say252</td>
<td>DB 40 ;speech speed DO NOT USE</td>
</tr>
<tr>
<td></td>
<td>DB FFH ;end</td>
</tr>
<tr>
<td>Tbl1_say253</td>
<td>DB 75 ;speech speed done RB</td>
</tr>
<tr>
<td></td>
<td>DB Voice+5 ;system pitch setting</td>
</tr>
<tr>
<td></td>
<td>DB 142</td>
</tr>
<tr>
<td></td>
<td>DB FFH ;end</td>
</tr>
<tr>
<td>Tbl1_say254</td>
<td>DB 46 ;speech speed done RB</td>
</tr>
<tr>
<td></td>
<td>DB Voice+5 ;system pitch setting</td>
</tr>
<tr>
<td></td>
<td>DB 102,149</td>
</tr>
<tr>
<td></td>
<td>DB FFH ;end</td>
</tr>
<tr>
<td>Tbl1_say255</td>
<td>DB 46 ;speech speed done RB</td>
</tr>
<tr>
<td></td>
<td>DB Voice+5 ;system pitch setting</td>
</tr>
<tr>
<td></td>
<td>DB 210,35,165,165,14,6</td>
</tr>
<tr>
<td></td>
<td>DB FFH ;end</td>
</tr>
<tr>
<td>Tbl2_say256</td>
<td>DB 46 ;speech speed done RB</td>
</tr>
<tr>
<td></td>
<td>DB Voice+5 ;system pitch setting</td>
</tr>
<tr>
<td></td>
<td>DB 148,163,149</td>
</tr>
<tr>
<td></td>
<td>DB FFH ;end</td>
</tr>
<tr>
<td>Tbl2_say257</td>
<td>DB 46 ;speech speed done RB</td>
</tr>
<tr>
<td></td>
<td>DB Voice+5 ;system pitch setting</td>
</tr>
<tr>
<td></td>
<td>DB 131,164,95,149,123</td>
</tr>
<tr>
<td></td>
<td>DB FFH ;end</td>
</tr>
<tr>
<td>Tbl2_say258</td>
<td>DB 55 ;speech speed SEQ 4, AGE 2 DONE RB</td>
</tr>
<tr>
<td></td>
<td>DB Voice+8 ;system pitch setting</td>
</tr>
<tr>
<td></td>
<td>DB 158,163,8,6</td>
</tr>
<tr>
<td></td>
<td>DB FFH ;end</td>
</tr>
<tr>
<td>Tbl2_say259</td>
<td>DB 45 ;speech speed SEQ 6, AGE 2 DONE RB</td>
</tr>
<tr>
<td></td>
<td>DB Voice+8 ;system pitch setting</td>
</tr>
<tr>
<td></td>
<td>DB 119,35,70,81</td>
</tr>
<tr>
<td></td>
<td>DB FFH ;end</td>
</tr>
<tr>
<td>Tbl3_say260</td>
<td>DB 46 ;speech speed RB DONE</td>
</tr>
</tbody>
</table>
DB Voice-8 ;system pitch setting SEQ 1, AGE 3
DB 119,66
DB FFH ;end
;
Tbl3_say261:
DB 46 ;speech speed SEQ 4, AGE 3 RB DONE
DB Voice-3 ;system pitch setting
DB 158,14,42
DB FFH ;end
;
Tbl3_say262:
DB 46 ;speech speed SEQ 6 AGE 3 RB DONE
DB Voice-3 ;system pitch setting
DB 119,35,5,93
DB FFH ;end
;
Tbl3_say263:
DB 60 ;speech speed SEQ 2, AGE 1 RB DONE
DB Voice+8 ;system pitch setting
DB 131,95,149
DB FFH ;end
;
Tbl3_say264:
DB 46 ;speech speed RB DONE
DB Voice-4 ;system pitch setting
DB 158,8,42
DB FFH ;end
;
Tbl3_say265:
DB 46 ;speech speed RB DONE
DB Voice-4 ;system pitch setting
DB 119,35,93
DB FFH ;end
;
END GEORGE 07/08/98
END LIGHT
DARK
/GEORGE 07/08/98

Tbl3_say266:
DB 52 ;speech speed BEGIN LIGHT DARKER
DB Voice-8 ;system pitch setting SEQ 1 AGE 1 RB DONE
DB 119,10,162,6
DB FFH ;end
;
Tbl3_say267:
DB 46 ;speech speed SEQ 2 AGE 1 DONE RB
DB Voice+8 ;system pitch setting
DB 119,6,21
DB FFH ;end
;
Tbl3_say268:
DB 55 ;speech speed
DB Voice+6 ;system pitch setting SEQ 3 AGE 1 DONE RB
DB 119,6,163,82,163,23
DB FFH ;end
;
Tbl3_say269:
DB 40 ;speech speed
DB Voice+8 ;system pitch setting SEQ 4 AGE 1 DONE RB
DB 158,101,163,104
; Tbl3_say270:
    DB 70 ; speech speed
    DB Voice*8 ; system pitch setting
    DB 148,10,6,148
    DB FFH ; end

; Tbl3_say271:
    DB 59 ; speech speed
    DB Voice*4 ; system pitch setting
    DB 149,163,21,21 ; SEQ6 AGE4/SEQ14 AGE4 LIGHT js
    DB FFH ; end

; Tbl3_say272:
    DB 52 ; speech speed
    DB Voice*8 ; system pitch setting
    DB 119,35,162,10,581
    DB FFH ; end DONE RB

; Tbl3_say273:
    DB 60 ; speech speed
    DB Voice*8 ; pitch control ONE RB
    DB 63,163,149,163,163,51,35,152
    DB FFH ; end

; Tbl3_say274:
    DB 52 ; speech speed
    DB Voice*2 ; system pitch setting
    DB 119,60.6
    DB FFH ; end

; Tbl3_say275:
    DB 2 ; speech speed
    DB Voice*2 ; pitch control
    DB 119,60.55,85
    DB FFH ; end DONE RB

; Tbl3_say276:
    DB 60 ; speech speed
    DB Voice*2 ; system pitch setting DONE RB
    DB 119,42,62,23
    DB FFH ; end

; Tbl3_say277:
    DB 70 ; speech speed
    DB Voice*2 ; system pitch setting
    DB 148,60,6,148
    DB FFH ; end DONE RB

; Tbl3_say278:
    DB 52 ; speech speed
    DB Voice*2 ; system pitch setting DONE RB
    DB 119,52,60,70,81
    DB FFH ; end

; Tbl3_say279:
    DB 52 ; speech speed
    DB Voice*2 ; system pitch setting
    DB 119,10,42
DB FFH ; end

Tbl3_say280:
DB 52 ; speech speed
DB Voice ; system pitch setting DONE RB
DB 119,10,34,85
DB FFH ; end

Tbl3_say281:
DB 60 ; speech speed
DB Voice ; system pitch setting
DB 119,42,83,23
DB FFH ; end DONE RB

Tbl3_say282:
DB 52 ; speech speed
DB Voice ; system pitch setting
DB 119,52,60,5,93
DB FFH ; end DONE RB

Tbl3_say283:
DB 60 ; speech speed
DB Voice ; system pitch setting
!NOTE!! PRINTED TO Had
DB 119,149,162,38,35,152
DB FFH ; end DONE RB

Tbl3_say284:
DB 52 ; speech speed
DB Voice ; system pitch setting
DB 119,60,42
DB FFH ; end DONE RB

Tbl3_say285:
DB 52 ; speech speed
DB Voice-3 ; system pitch setting
DB 119,60,34,85
DB FFH ; end

Tbl3_say286:
DB 60 ; speech speed
DB Voice ; system pitch setting
DB 119,42,83,68
DB FFH ; end

Tbl3_say287:
DB 70 ; speech speed
DB Voice ; system pitch setting
DB 148,60,42,148
DB FFH ; end

Tbl3_say288:
DB 46 ; speech speed
DB Voice ; system pitch setting
DB 119,163,52,60,70,93 ; SEQ7 AGE4/SEQ15 AGE 4 LIGHT js
DB FFH ; end

Tbl3_say289:
DB 50 ; speech speed
DB Voice ; system pitch setting
DB 50 ;speech speed
DB 163,148,165,17 ;S1-A1, S2-A1 SOUND js
DB FFH ;end ;S9-A2/S1-A3/S9-A3 SOUND js
;
Tbl3_say291:
DB 46 ;speech speed
DB 85,165,165,165 ;S2-A1/S2-A1/S2-A1 SOUND js
DB FFH ;end ;S2-A4/S10-A4 SOUND js
;
Tbl3_say292:
DB 46 ;speech speed
DB 35,163,89 ;S5-A1/S5-A1 SOUND (with say/m2) js
DB FFH ;end
;
Tbl3_say293:
DB 53 ;speech speed
DB 163,148,163,36 ;S6-A1/S6-A1/S6-A2 SOUND js
DB FFH ;end ;S6-A2/S6-A3/S6-A3 SOUND js
;
Tbl3_say294:
DB 53 ;speech speed
DB 17 ;S7-A1/S7-A1 SOUND (with say/m2) js
DB FFH ;end
;
Tbl3_say295:
DB 60 ;speech speed
DB 122,164,21,164,21 ;S8-A1/S8-A1 SOUND js
DB FFH ;end ;S8-A3/S8-A3 SOUND js
;
Tbl3_say296:
DB 46 ;speech speed
DB Voice ;system pitch setting
DB 121,165,164,8,16',41,21 ;S3-A2/S11-A2 SOUND js
DB FFH ;end

Tbl3_say299:
DB 46 ;speech speed
DB Voice ;system pitch setting
DB 163,129,164,5,165,73 ;S4-A2/S12-A2 SOUND js
DB FFH ;end

Tbl3_say300:
DB 46 ;speech speed
DB Voice ;system pitch setting
DB 35,165,31 ;S5-A2/S13-A2/S5-A3 SOUND (with say/m2) js
DB FFH ;end ;S13-A3/S5-A4/S13-A4 SOUND (with say/m2) js

Tbl3_say301:
DB 46 ;speech speed
DB Voice ;system pitch setting
DB 8,162,41,163,85 ;S7-A2/S15-A2 SOUND (with say/m2) js
DB FFH ;end

Tbl3_say302:
DB 60 ;speech speed
DB Voice ;system pitch setting
DB 122,164,21 ;S8-A2/S16-A2 SOUND js
DB FFH ;end

Tbl3_say303:
DB 46 ;speech speed
DB Voice ;system pitch setting
DB 121,165,164,14,163,73,21 ;S3-A3/S11-A3 SOUND js
DB FFH ;end

Tbl3_say304:
DB 46 ;speech speed
DB Voice ;system pitch setting
DB 163,129,164,35,165,44 ;S4-A3/S12-A3 SOUND js
DB FFH ;end ;S1-A4/S12-A4 SOUND js

Tbl3_say305:
DB 46 ;speech speed
DB Voice ;system pitch setting
DB 8,73,164,85 ;S7-A3/S15-A3 SOUND (with say/m2) js
DB FFH ;end ;S7-A4/S15-A4 SOUND (with say/m2) js

Tbl3_say306:
DB 55 ;speech speed
DB Voice ;system pitch setting
DB 164,148,164,163,46 ;S1-A4/S9-A4 SOUND js
DB FFH ;end

Tbl3_say307:
DB 46 ;speech speed
DB Voice ;system pitch setting
DB 121,165,164,8,163,73,21 ;S3-A4/S11-A4 SOUND js
DB FFH ;end
; Tilt
; George 07/09/98

Tbl3_say308:
DB 55 ; speech speed
DB Voice ; system pitch setting
DB 164,146,164,163,54 ; S6-A4/S14-A4 Sound js
DB FFH ; end

Tbl3_say309:
DB 60 ; speech speed
DB Voice ; system pitch setting
DB 122,164,163,88,164,21 ; S8-A4/S16-A4 Sound js

; END SOUND
;

; Tilt

Tbl3_say310:
DB 56 ; speech speed
DB Voice*8 ; pitch control
DB 160 ; S1 A1 TILT/S4 A1 TILT/S14 A1 TILT js
DB FFH ; end

Tbl3_say311:
DB 46 ; speech speed
DB Voice ; pitch control
DB 157 36 ; S2 A1 TILT js
DB FFH ; end

Tbl3_say312:
DB 46 ; speech speed
DB Voice ; system pitch setting
DB 158,9 ; S3 A1 TILT js
DB FFH ; end

Tbl3_say313:
DB 46 ; speech speed
DB Voice*8 ; system pitch setting
DB 154 ; S5 A1/S4 A2/S2 A3/S2 A4 TILT js
DB FFH ; end

Tbl3_say314:
DB 46 ; speech speed
DB Voice ; system pitch setting
DB 159,82,39 ; S6 A1 TILT js
DB FFH ; end

Tbl3_say315:
DB 46 ; speech speed
DB Voice ; system pitch setting
DB 155,39,39 ; S7 A1 TILT/S6 A2 TILT ja
DB FFH ; end

Tbl3_say316:
DB 46 ; speech speed
DB Voice ; system pitch setting
DB 37,152 ; S8 A1 TILT (with say/m5) js
DB FFH ; end

Tbl3_say317:
DB 46 ;speech speed
DB Voice ;system pitch setting
DB 154,120 ;S9 A1 TILT/S9 A2 TILT js
DB FFH ;end

Tbl3_say318:
DB 46 ;speech speed
DB Voice ;system pitch setting
DB 155,120,120 ;S10 A1 TILT/S10 A2 TILT js
DB FFH ;end

Tbl3_say319:
DB 46 ;speech speed
DB Voice ;system pitch setting
DB 35,57 ;S11 A1 TILT (with say/m21) js
DB FFH ;end

Tbl3_say320:
DB 48 ;speech speed
DB Voice ;system pitch setting
DB 158,10,80 ;S12 A1 TILT js
DB FFH ;end

Tbl3_say321:
DB 46 ;speech speed
DB Voice ;system pitch setting
DB 119,150 ;S13 A1 / S15 A3 TILT js
DB FFH ;end

Tbl3_say322:
DB 46 ;speech speed
DB Voice ;system pitch setting
DB 160,9 ;S15 A1 TILT js
DB FFH ;end

Tbl3_say323:
DB 46 ;speech speed
DB Voice ;system pitch setting
DB 154,149 ;S16 A1 / S15 A2 / S13 A3 TILT js
DB FFH ;end

Tbl3_say324:
DB 46 ;speech speed
DB Voice ;system pitch setting
DB 160 ;S1 A2/S3 A2/S1 A3/S1 A4 TILT js
DB FFH ;end

Tbl3_say325:
DB 46 ;speech speed
DB Voice ;system pitch setting
DB 52,9 ;S2 A1 TILT (with say/m16) js
DB FFH ;end

Tbl3_say326:
DB 46 ;speech speed
DB Voice ;system pitch setting
DB 159,83,39 ;S5 A2 TILT js
DB FFH ;end

Tbl3_say327:
DB 46 ; speech speed
DB Voice ; system pitch setting
DB 52,43,81,152 ; S7 A2 TILT (with say/m5) js
DB FFH ; end

; Tbl3_say328:
DB 46 ; speech speed
DB Voice ; system pitch setting
DB 155 ; S8 A2 TILT (with say/m5) js
DB FFH ; end

; Tbl3_say329:
DB 46 ; speech speed
DB Voice ; system pitch setting
DB 52,57 ; S11 A2 TILT (with say/m2) js
DB FFH ; end

; Tbl3_say330:
DB 46 ; speech speed
DB Voice ; system pitch setting
DB 158,60,80 ; S12 A2 TILT js
DB FFH ; end

; Tbl3_say331:
DB 46 ; speech speed
DB Voice ; system pitch setting
DB 163,156 ; S13 A2 TILT (with say/m5) js
DB FFH ; end

; Tbl3_say332:
DB 46 ; speech speed
DB Voice ; system pitch setting
DB 8,22,85 ; S14 A2 TILT js
DB FFH ; end

; Tbl3_say333:
DB 46 ; speech speed
DB Voice ; pitch control
DB 154,118,163,145,165,162.118 ; S16 A2/S14 A3/S14 A4 TILT js
DB FFH ; end

; Tbl3_say334:
DB 46 ; speech speed
DB Voice ; system pitch setting
DB 159 ; S3 A3 TILT js
DB FFH ; end

; Tbl3_say335:
DB 46 ; speech speed
DB Voice ; pitch control
DB 83,1 ; S4 A3/S4 A4 TILT (with say/m26) js
DB FFH ; end

; Tbl3_say336:
DB 46 ; speech speed
DB Voice ; system pitch setting
DB 155,52,62,85 ; S5 A3 TILT js
DB FFH ; end

A-238
Tbl3_say337:
  DB  50 ;speech speed
  DB  Voice ;system pitch setting
  DB  52,48,93,152 ;S6 A3 TILT (with say/m5) js
  DB  FFH ;end

Tbl3_say338:
  DB  46 ;speech speed
  DB  Voice ;system pitch setting
  DB  155 ;S7 A3/S7 A4 TILT (with say/m5) js
  DB  FFH ;end

Tbl3_say339:
  DB  46 ;speech speed
  DB  Voice ;system pitch setting
  DB  155,120,163,149 ;S8 A3/S8 A4 TILT js
  DB  FFH ;end

Tbl3_say340:
  DB  46 ;speech speed
  DB  Voice ;system pitch setting
  DB  165,129 ;S9 A3/S9 A4 TILT (with say/m9) js
  DB  FFH ;end

Tbl3_say341:
  DB  46 ;speech speed
  DB  Voice ;system pitch setting
  DB  160,153,120,120 ;S10 A3/S10 A4 TILT (with say/m10) js
  DB  FFH ;end

Tbl3_say342:
  DB  46 ;speech speed
  DB  Voice ;system pitch setting
  DB  163,23 ;S11 A3/S11 A4 TILT (with say/m23) js
  DB  FFH ;end

Tbl3_say343:
  DB  55 ;speech speed
  DB  Voice ;system pitch setting
  DB  164,156 ;S12 A3 TILT (with say/m5) js
  DB  FFH ;end

Tbl3_say344:
  DB  46 ;speech speed
  DB  Voice ;system pitch setting
  DB  163,1,163,1,117 ;S16 A3 TILT (with say/m117) js
  DB  FFH ;end

Tbl3_say345:
  DB  46 ;speech speed
  DB  Voice ;system pitch setting
  DB  27,162,149 ;S1 A4 TILT (with say/m27) js
  DB  FFH ;end

Tbl3_say346:
  DB  46 ;speech speed
  DB  Voice ;system pitch setting
  DB  155,52,29,163,85 ;S5 A4 TILT js
  DB  FFH ;end
Tbl3_say347:
    DB 46 ;speech speed
    DB Voice ;system pitch setting
    DB 52,47,93,164,152 ;S6 A4 TILT (with say/m5) js
    DB FFH ;end

Tbl3_say348:
    DB 46 ;speech speed
    DB Voice ;system pitch setting
    DB 52,24,68 ;S11 A4 TILT (with say/m2) js
    DB FFH ;end

Tbl3_say349:
    DB 46 ;speech speed
    DB Voice ;system pitch setting
    DB 22,149 ;S13 A4 TILT (with say/m5) js
    DB FFH ;end

Tbl3_say350:
    DB 46 ;speech speed
    DB Voice ;system pitch setting
    DB 163,1,163,39,163,117 ;S16 A4 TILT (with say/m5) js
    DB FFH ;end

;END GEORGE 07/09/98
;
;GEORGE
;IR 07/09/98
Tbl3_say351:
    DB 46 ;speech speed
    DB Voice ;pitch control
    DB 40 ;SEQ1,seq2,seq3,seq4 ir age 1
    DB FFH ;end

Tbl3_say352:
    DB 46 ;speech speed
    DB Voice ;pitch control
    DB 66,162,85 ;seq5, ir age1
    DB FFH ;end

Tbl3_say353:
    DB 46 ;speech speed
    DB Voice ;system pitch setting
    DB 19,85 ;seq5, ir age1 DANCE WAH
    DB FFH ;end

Tbl3_say354:
    DB 46 ;speech speed
    DB Voice ;system pitch setting
    DB 162,164,134,134 ;seq6, ir age1 DO DO DO
    DB FFH ;end

Tbl3_say355:
    DB 46 ;speech speed
    DB Voice ;system pitch setting
    DB 134,134,25,19 ;seq7 ir age1
    DB FFH ;end

Tbl3_say356:
    DB 50 ;speech speed
DB  Voice*8  ;system pitch setting
DB  162
DB  FFH  ;end

Tbl3_say357:
DB  42  ;speech speed
DB  Voice  ;system pitch setting
DB  102,97,118,34  ;seq8 ir agel
DB  FFH ;end

Tbl3_say358:
DB  50  ;speech speed
DB  Voice  ;system pitch setting
DB  117,34,22  ;seq9 ir agel
DB  FFH ;end

Tbl3_say359:
DB  50  ;speech speed
DB  Voice  ;system pitch setting
DB  30,78,145,145  ;seq10,11 ir agel
DB  FFH ;end

Tbl3_say360:
DB  50  ;speech speed
DB  Voice  ;system pitch setting
DB  150,151,93,71  ;seq12 ir agel TWINKLE
DB  FFH ;end

Tbl3_say361:
DB  46  ;speech speed
DB  Voice  ;system pitch setting
DB  FFH ;end

Tbl3_say362:
DB  46  ;speech speed
DB  Voice  ;system pitch setting
DB  161,72,161  ;seq15 ir agel
DB  FFH ;end

Tbl3_say363:
DB  60  ;speech speed
DB  Voice  ;system pitch setting
DB  144,144,144,144  ;seq16 ir agel
DB  FFH ;end

Tbl3_say364:
DB  46  ;speech speed
DB  Voice*5  ;system pitch setting
DB  81,40  ;seq1,2,3 ir age2
DB  FFH ;end

Tbl3_say365:
DB  46  ;speech speed
DB  Voice*8  ;system pitch setting
DB  81,40  ;seq4,5 ir age2
DB  FFH ;end

Tbl3_say366:
DB 46 ; speech speed
DB Voice+6 ; system pitch setting
DB 66,159 ; seq6 ir age2
DB FFH ; end

Tbl3_say367:
DB 46 ; speech speed
DB Voice+7 ; system pitch setting
DB 19,165,165,165,164,134,165,135 ; seq7, 8 ir age2
DB FFH ; end

Tbl3_say368:
DB 46 ; speech speed
DB Voice+3 ; system pitch setting
DB 118,25,34 ; seq9 ir age2
DB FFH ; end

Tbl3_say369:
DB 51 ; speech speed
DB Voice+8 ; system pitch setting
DB 102,97,118 ; seq10 ir age2
DB FFH ; end

Tbl3_say370:
DB 46 ; speech speed
DB Voice+5 ; system pitch setting
DB 117,34,32 ; seq11 ir age2
DB FFH ; end

Tbl3_say371:
DB 48 ; speech speed
DB Voice ; system pitch setting
DB FFH ; end

Tbl3_say372:
DB 55 ; speech speed
DB Voice ; system pitch setting
DB 161,72,161 ; seq15 ir age2
DB FFH ; end

Tbl3_say373:
DB 50 ; speech speed
DB Voice ; system pitch setting
DB 143,144,143 ; seq16 ir age2
DB FFH ; end

Tbl3_say374:
DB 50 ; speech speed
DB Voice ; pitch control
DB 14,40 ; seq1, 2, 3, 4, 5 ir age3
DB FFH ; end

Tbl3_say375:
DB 46 ; speech speed
DB Voice+5 ; system pitch setting
DB 35,48,66 ; seq6 ir age3
DB FFH ; end
; Tbl3_say376:
  DB 50 ;speech speed
  DB Voice*8 ;pitch control
  DB 19,12,134,134 ;seq7,8 ir age3
  DB FFH ;end

; Tbl3_say377:
  DB 46 ;speech speed
  DB Voice*3 ;system pitch setting
  DB 34,85,99 ;seq9 ir age3
  DB FFH ;end

; Tbl3_say378:
  DB 46 ;speech speed
  DB Voice*2 ;system pitch setting
  DB 156,25,34 ;seq11 ir age3
  DB FFH ;end

; Tbl3_say379:
  DB 50 ;speech speed
  DB Voice*3 ;system pitch setting
  DB 63,165,165,165,165,165,124,31 ;seq13,14 ir age3
  DB FFH ;end

; Tbl3_say380:
  DB 70 ;speech speed
  DB Voice*4 ;system pitch setting
  DB 35,72,162,162,162,162,162,162,162,162,162,162,162,162,161
  DB FFH ;end

; Tbl3_say381:
  DB 58 ;speech speed
  DB Voice*5 ;system pitch setting
  DB 40,85 ;seq1,2,3,4,5 IR AGE4
  DB FFH ;end

; Tbl3_say382:
  DB 46 ;speech speed
  DB Voice*6 ;system pitch setting
  DB 81,66,21 ;seq6 ir age4
  DB FFH ;end

; Tbl3_say383:
  DB 46 ;speech speed
  DB Voice*7 ;system pitch setting
  DB 134,134,25,19 ;seq7,8 ir age4
  DB FFH ;end

; Tbl4_say384:
  DB 50 ;speech speed
  DB Voice*8 ;system pitch setting
  DB 34,78,145,145 ;seq9 ir age4
  DB FFH ;end

; Tbl4_say385:
  DB 50 ;speech speed
  DB Voice*8 ;system pitch setting
  DB 119,44,52,71,150 ;seq10 ir age4
  DB FFH ;end
DIALOGUE

; Tbl4_say386:
  DB 46 ; speech speed
  DB Voice+8 ; system pitch setting
  DB 34,85,99 seq11 ir age4
  DB FFH ; end

; Tbl4_say387:
  DB 50 ; speech speed
  DB Voice+1 ; system pitch setting
  DB 119,124,31 ; seq12 ir age4
  DB FFH ; end

; Tbl4_say388:
  DB 56 ; speech speed
  DB Voice+3 ; system pitch setting
  DB 162,63 ; seq14 ir age4
  DB FFH ; end

; Tbl4_say389:
  DB 60 ; speech speed
  DB Voice-8 ; system pitch setting
  DB 161,164,161 ; SEQ10 HANGING (YAWN)
  DB 46 ; speech speed
  DB Voice+3 ; system pitch setting
  DB 161,144,144 ; seq15 ir age4
  DB FFH ; end

; Tbl4_say390:
  DB 55 ; speech speed
  DB Voice+3 ; system pitch setting
  DB 165,165,164,165,164,165,164,164
  DB FFH ; end

; END IR
; END GEORGE

; ADDED BY DMH (FOR FURBY SAYS)
Tbl4_say391:
  DB 46 ; speech speed
  DB Voice+4 ; system pitch setting
  DB 142,144,143 ; seq16 ir age4
  DB FFH ; end

; Tbl4_say42:
  DB 46 ; speech speed
  DB Voice+4 ; system pitch setting
  DB 4
  DB FFH ; end
Tbl4_say392:
DB 52 ;speech speed
DB Voice ;system pitch setting
DB 60,42 ;no light
DB FFH ;end
;
Tbl4_say393:
DB 55 ;speech speed
DB Voice ;system pitch setting
DB 164,163,46 ;LOUD SOUND
DB FFH ;end
;
Tbl4_say394:
DB 46 ;speech speed
DB Voice ;system pitch setting
DB 164,163,44 ;LISTEN (FURBY SAYS)
DB FFH ;end
;
Tbl4_say395:
DB 46 ;speech speed
DB Voice ;system pitch setting
DB 52,163 ;(ME) with names (dmh)
DB FFH ;end
;
Tbl4_say396:
DB 56 ;speech speed
DB Voice ;system pitch setting
DB 162,55 ;name (MEE MEE) (dmh)
DB FFH ;end
;
Tbl4_say397:
DB 58 ;speech speed
DB Voice ;system pitch setting
DB 163,23 ;(DO MOH)
DB FFH ;end
;
Tbl4_say398:
DB 60 ;speech speed
DB Voice ;system pitch setting
DB 80 ;TOH-LOO
DB FFH ;end
;
Tbl4_say399:
DB 60 ;speech speed
DB Voice ;system pitch setting
DB 165 ;DELAY 1 SECOND DMH
DB FFH ;end
;
; start of diagnostic tables dmh
Tbl4_say400:
DB 0 ;speech speed
DB Voice+16 ;system pitch setting
DB 168,168,168 ;used at start of diagnostics
DB FFH ;end
;
Tbl4_say401:
DB 20 ;speech speed
DB Voice+13 ;system pitch setting
DB 169,165 ;key beep

A-245
Tbl4_say402:
  DB 20 ;speech speed
  DB Voice+5 ;system pitch setting
  DB 169,163,169,163,169 ;pass test
  DB 2
  DB FFH ;end

Tbl4_say403:
  DB 96 ;speech speed
  DB Voice-40 ;system pitch setting
  DB 169,163 ;fail test tone
  DB FFH ;end

Tbl4_say404:
  DB 46 ;speech speed
  DB Voice ;system pitch setting
  DB 169 ;speaker tone test
  DB FFH ;end

Tbl4_say405:
  DB 46 ;speech speed
  DB Voice ;system pitch setting
  DB 163 ;no sound for start of motor cal
  DB FFH ;end

Tbl4_say406:
  DB 20 ;speech speed
  DB Voice+5 ;system pitch setting
  DB 169,163,169,163,169 ;feed1
  DB FFH ;end

Tbl4_say407:
  DB 20 ;speech speed
  DB Voice+5 ;system pitch setting
  DB 169,163,169,163,169 ;pass feed sw
  DB FFH ;end

Tbl4_say408:
  DB 20 ;speech speed
  DB Voice+5 ;system pitch setting
  DB 169,163,169,163,169 ;pass light test
  DB FFH ;end

Tbl4_say409:
  DB 20 ;speech speed
  DB Voice+5 ;system pitch setting
  DB 169,163,169,163,169 ;pass sound test
  DB FFH ;end

Tbl4_say410:
  DB 20 ;speech speed
  DB Voice+5 ;system pitch setting
  DB 169,163,169,163,169 ;pass all test complete
  DB 159
  DB FFH ;end

Tbl4_say411:
DB 60 ; speech speed ; HIDE ME (HIDE AND SEEK) DHM
DB Voice+3 ; system pitch setting
DB 31,52 ; HIDE ME
DB FFH ; end

DB 100 ; speech speed
DB Voice ; system pitch setting
DB 167,167,167 ; SEQ1 FEED AGE1 (AAAA"")
DB FFH ; end
ON POWER UP, UNTIL WAKE-UP TABLE INSTALLED (Dave)

DB 45  ; speech speed
DB Voice
DB 165
DB FFH  ; end

; Motor tables

; Offset pointer:

Motor_grpl:

DW     Tbl1_M000
DW     Tbl1_M001,Tbl1_M002,Tbl1_M003,Tbl1_M004,Tbl1_M005
DW     Tbl1_M006,Tbl1_M007,Tbl1_M008,Tbl1_M009,Tbl1_M010
DW     Tbl1_M011,Tbl1_M012,Tbl1_M013,Tbl1_M014,Tbl1_M015
DW     Tbl1_M016,Tbl1_M017,Tbl1_M018,Tbl1_M019,Tbl1_M020
DW     Tbl1_M021,Tbl1_M022,Tbl1_M023,Tbl1_M024,Tbl1_M025
DW     Tbl1_M026,Tbl1_M027,Tbl1_M028,Tbl1_M029,Tbl1_M030
DW     Tbl1_M031,Tbl1_M032,Tbl1_M033,Tbl1_M034,Tbl1_M035
DW     Tbl1_M036,Tbl1_M037,Tbl1_M038,Tbl1_M039,Tbl1_M040
DW     Tbl1_M041,Tbl1_M042,Tbl1_M043,Tbl1_M044,Tbl1_M045
DW     Tbl1_M046,Tbl1_M047,Tbl1_M048,Tbl1_M049,Tbl1_M050
DW     Tbl1_M051,Tbl1_M052,Tbl1_M053,Tbl1_M054,Tbl1_M055
DW     Tbl1_M056,Tbl1_M057,Tbl1_M058,Tbl1_M059,Tbl1_M060
<table>
<thead>
<tr>
<th>Motor_grp2:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Tbl1_M061, Tbl1_M062, Tbl1_M063, Tbl1_M064, Tbl1_M065</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M066, Tbl1_M067, Tbl1_M068, Tbl1_M069, Tbl1_M070</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M071, Tbl1_M072, Tbl1_M073, Tbl1_M074, Tbl1_M075</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M076, Tbl1_M077, Tbl1_M078, Tbl1_M079, Tbl1_M080</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M081, Tbl1_M082, Tbl1_M083, Tbl1_M084, Tbl1_M085</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M086, Tbl1_M087, Tbl1_M088, Tbl1_M089, Tbl1_M090</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M091, Tbl1_M092, Tbl1_M093, Tbl1_M094, Tbl1_M095</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M096, Tbl1_M097, Tbl1_M098, Tbl1_M099</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M100, Tbl1_M101, Tbl1_M102, Tbl1_M103, Tbl1_M104</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M105, Tbl1_M106, Tbl1_M107, Tbl1_M108, Tbl1_M109</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M110, Tbl1_M111, Tbl1_M112, Tbl1_M113, Tbl1_M114</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M115, Tbl1_M116, Tbl1_M117, Tbl1_M118, Tbl1_M119</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M120, Tbl1_M121, Tbl1_M122, Tbl1_M123, Tbl1_M124</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M125, Tbl1_M126, Tbl1_M127</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Motor_grp3:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Tbl1_M128</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M129, Tbl1_M130, Tbl1_M131, Tbl1_M132, Tbl1_M133</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M134, Tbl1_M135, Tbl1_M136, Tbl1_M137, Tbl1_M138</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M139, Tbl1_M140, Tbl1_M141, Tbl1_M142, Tbl1_M143</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M144, Tbl1_M145, Tbl1_M146, Tbl1_M147, Tbl1_M148</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M149, Tbl1_M150, Tbl1_M151, Tbl1_M152, Tbl1_M153</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M154, Tbl1_M155, Tbl1_M156, Tbl1_M157, Tbl1_M158</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M159, Tbl1_M160, Tbl1_M161, Tbl1_M162, Tbl1_M163</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M164, Tbl1_M165, Tbl1_M166, Tbl1_M167, Tbl1_M168</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M169, Tbl1_M170, Tbl1_M171, Tbl1_M172, Tbl1_M173</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M174, Tbl1_M175, Tbl1_M176, Tbl1_M177, Tbl1_M178</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M179, Tbl1_M180, Tbl1_M181, Tbl1_M182, Tbl1_M183</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M184, Tbl1_M185, Tbl1_M186, Tbl1_M187, Tbl1_M188</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M189, Tbl1_M190, Tbl1_M191, Tbl1_M192, Tbl1_M193</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M194, Tbl1_M195, Tbl1_M196, Tbl1_M197, Tbl1_M198</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M199, Tbl1_M200, Tbl1_M201, Tbl1_M202, Tbl1_M203</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M204, Tbl1_M205, Tbl1_M206, Tbl1_M207, Tbl1_M208</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M209, Tbl1_M210, Tbl1_M211, Tbl1_M212, Tbl1_M213</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M214, Tbl1_M215, Tbl1_M216, Tbl1_M217, Tbl1_M218</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M219, Tbl1_M220, Tbl1_M221, Tbl1_M222, Tbl1_M223</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M224, Tbl1_M225, Tbl1_M226, Tbl1_M227, Tbl1_M228</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M229, Tbl1_M230, Tbl1_M231, Tbl1_M232, Tbl1_M233</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M234, Tbl1_M235, Tbl1_M236, Tbl1_M237, Tbl1_M238</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M239, Tbl1_M240, Tbl1_M241, Tbl1_M242, Tbl1_M243</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M244, Tbl1_M245, Tbl1_M246, Tbl1_M247, Tbl1_M248</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M249, Tbl1_M250, Tbl1_M251, Tbl1_M252, Tbl1_M253</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M254, Tbl1_M255</td>
<td></td>
</tr>
</tbody>
</table>
Each motor table has the following format:

- The first line is the delay between motor steps.
- The next group of lines are the motor steps.
- The last line is the terminator command.

Delay table - a number from 0 - 255. The entry is multiplied by a 2.9 mSec timer. Therefore 1=2.9mSec 2=5.8mSec 255=739mSec.

The motor step is entered as a decimal number of 10-190.

'00' is a PAUSE command base on the motor delay setting.
'FF' or '255' is the end of table command.

Tables with ending step not within required range (10-20), (132, 136)

M94, M127, M131, M139, M140, M143, M146

With duplicate steps put consecutively

M187, M193, M220, M229, M237, M241, M242

M250, M310, M321, M369

TBL1_M000:
   DB 50 ; motor delay between steps
   DB 10, 135
   DB FFH ; end

GEORGE 07/03/96

TBL1_M001:
   .DON START SEQ1 AGE1
   DB 1 ; motor delay between steps
   DB 190, 133
   DB FFH

TBL1_M002:
   .DON START SEQ2 AGE1
   DB 1 ; motor delay between steps
   DB 150, 145, 138, 120, 145, 133, 147, 133
   DB FFH ; end

TBL1_M003:
   10 ; motor delay between steps
   DB 90, 100, 0, 0, 100, 0, 0, 0, 0, 133 ; CONNECTED M23 ; DON START
   DB 145, 160, 0, 0, 0, 0, 0
   DB FFH ; end

TBL1_M004:
   1 ; motor delay between steps
   DB 200, 190, 160, 100, 133 ; CONNECTED M22 ; DON START
   DB FFH ; end

TBL1_M005:
   5 ; motor delay between steps
   DB 170, 130, 90, 100, 133 ; DONE connected M22 seq4 age1
   DB FFH ; end

TBL1_M006:
   10 ; motor delay between steps
   DB 150, 200, 0, 0, 150, 133 ; seq5 front1 age1
   DB FFH ; end

TBL1_M007:
   1 ; motor delay between steps
   DB 120, 150, 133 ; SEQ6 FRONT1 AGE1 HORSE LAUGH
   DB FFH ; end

TBL1_M008:
   10 ; motor delay between steps
   DB 150, 200, 150, 170, 133 ; SEQ7 FRONT AGE1
DB FFH ; end

; Tbl1_M009:
DB 10 ; motor delay between steps
DB 150,200,150,190,170,120,133 ; SEQ8, FRONT AGE1
DB FFH ; end

; Tbl1_M010:
DB 1 ; motor delay between steps
DB 180,100,133 ; SEQ9, FRONT AGE1
DB FFH ; end

; Tbl1_M011:
DB 1 ; motor delay between steps
DB 80,0,125,0,0,133 ; SEQ10, FRONT AGE1
DB FFH ; end

; Tbl1_M012:
DB 10 ; motor delay between steps
DB 125,0,0,0,0,0,0,133,80,133 ; SEQ11, FRONT AGE1
DB FFH ; end

; Tbl1_M013:
DB 20 ; motor delay between steps
DB 145,133,145,133,145,133,145
DB 125,0,0,0,0,130,0,90,133 ; SEQ12 FRONT AGE1 ADD
SAY20 TO FRONT
DB FFH ; end

; Tbl1_M014:
DB 10 ; motor delay between steps
DB 90,130,120,0,0,133 ; SEQ13 FRONT AGE1 ADD
SAY 22
DB FFH ; end

; Tbl1_M015:
DB 10 ; motor delay between steps
DB 125,110,133 ; SEQ14 FRONT AGE1 ADD
SAY 22
DB FFH ; end

; Tbl1_M016:
DB 1 ; motor delay between steps
DB 160,0,0,133,125,150,133 ; SEQ15 FRONT AGE1
DB FFH ; end

; Tbl1_M017:
DB 10 ; motor delay between steps
DB 120,133,125,150,120,0,0,0,0,0,0,0,0,133 ; SEQ16 FRONT AGE1 ADD 37
DB FFH ; end

; Tbl1_M018:
DB 1 ; motor delay between steps
DB 124,0,115,0,133,120,133 ; SEQ16 FRONT AGE1 ADD 37
DB FFH ; end
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TB11_M019</td>
<td>DB 10 ;motor delay between steps, 30,100,0,0,100,0,0,0,0,133 ;SEQ1 FRONT AGE2 175,160,0,0,160,0,0,0,0,133</td>
</tr>
<tr>
<td>TB11_M020</td>
<td>DB 10 ;motor delay between steps, 143,150,133,155,133 ;SEQ2 FRONT AGE2</td>
</tr>
<tr>
<td>TB11_M021</td>
<td>DB 1 ;motor delay between steps, 180,133,180,133 ;SEQ3 AGE2 FRONT ADD SEQ9AGE1</td>
</tr>
<tr>
<td>TB11_M022</td>
<td>DB 10 ;motor delay between steps, 140,150,133 ;SEQ4 AGE2 FRONT</td>
</tr>
<tr>
<td>TB11_M023</td>
<td>DB 1 ;motor delay between steps, 120,133,0,0,0,0,0,0,140,150,133</td>
</tr>
<tr>
<td>TB11_M024</td>
<td>DB 5 ;motor delay between steps, 150,140,138,120,145,133,0,147,133</td>
</tr>
<tr>
<td>TB11_M025</td>
<td>DB 1 ;motor delay between steps, 150,200,0,0,150,133,143,133,143</td>
</tr>
<tr>
<td>TB11_M026</td>
<td>DB 10 ;motor delay between steps, 142,150,133 ;SEQ7 AGE2 FRONT PART1</td>
</tr>
<tr>
<td>TB11_M027</td>
<td>DB 1 ;motor delay between steps, 150,145,160,133,145,133,145,133,133</td>
</tr>
</tbody>
</table>

Danger always followed by 003: dmh

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TB11_M028</td>
<td>DB 1 ;motor delay between steps, 30,70 ;&lt;-- OK ;SEQ8 MIDDLE OF 22, AND 4SOMETHING</td>
</tr>
<tr>
<td>TB11_M029</td>
<td>DB 1 ;motor delay between steps, 190,133 ;SEQ9 TITTER</td>
</tr>
</tbody>
</table>

A-255
// Tbl1_M030:
DB   1           ;motor delay between steps
DB   120,133,140,150,133   ; SEQ10 FRONT AGE2
DB   FFH   ;end

// Tbl1_M031:
DB   5           ;motor delay between steps
DB   180,160,133,115,105,133   ; SEQ11 FRONT AGE 2 ADD 41
DB   FFH   ;end

// Tbl1_M032:
DB   10           ;motor delay between steps
DB   145,133,145,133,145,133,0,120,115,133   ; SEQ12 FRONT AGE 2 ADD 20
DB   FFH   ;end

// Tbl1_M033:
DB   1           ;motor delay between steps
DB   150,170,190,133,120,133,135,133,150,0,0,133   ; SEQ14 FRONT
DB   FFH   ;end

// Tbl1_M034:
DB   10           ;motor delay between steps
DB   125,0,0,0,0,0,133,145,133   ; SEQ15 FRONT AGE2 ADD 20
DB   FFH   ;end

// Tbl1_M035:
DB   1           ;motor delay between steps
DB   120,0,0,0,0,0,0,133,145
DB   133,0,150,133,110,133,120,0,0,133   ; SEQ16 FRONT AGE2 ADD 20
DB   FFH   ;end

// Tbl1_M036:
DB   1           ;motor delay between steps
DB   155,0,0,0,133   ; SEQ1 FRONT AGE3
DB   FFH   ;end

// Tbl1_M037:
DB   1           ;motor delay between steps
DB   140,150,133,120,133,110,133   ; SEQ2 FRONT AGE3
DB   FFH   ;end

// Tbl1_M038:
DB   1           ;motor delay between steps
DB   155,0,0,0,133,155,0,0,0,133   ; SEQ3 FRONT AGE3
DB   FFH   ;end

// Tbl1_M039:
DB   1           ;motor delay between steps
DB   190,0,0,133   ; SEQ4 FRONT AGE3
DB   FFH   ;end

; ERROR
// Tbl1_M040:
DB   10           ;motor delay between steps
DB   140,150,133   ; SEQ5 FRONT AGE3 ADD SEQ14AGE1
DB   FFH   ;end
TABLE M040:

DB 10 ; motor delay between steps
DB 150,200,0,0,150,133,143,133
DB 143,133,110,0,0,133 ; SEQ6 FRONT AGE3
DB FFH ; end

TABLE M041:

DB 1 ; motor delay between steps
DB 160,140,0,150,133,160,140,133
DB 150,160,133 ; SEQ7 FRONT AGE3
DB FFH ; end

TABLE M042:

DB 1 ; motor delay between steps
DB 30,70,120 ; SEQ7
DB 160,140,0,150,133,160,140,133
DB FFH ; end

TABLE M043:

DB 10 ; motor delay between steps
DB 80,0,150,0,125,0,0,133 ; SEQ10 FRONT AGE3
DB FFH ; end

TABLE M044:

DB 1 ; motor delay between steps
DB 100,133,120,133 ; SEQ11
DB FFH ; end

TABLE M045:

DB 10 ; motor delay between steps
DB 150,0,0,133,120,100,113 ; SEQ12 FRONT AGE3
{HEEY, TICKLE ME} ADD20
DB 4
DB FFH ; end

TABLE M046:

DB 10 ; motor delay between steps
DB 145,133,145,133,145,133 ; SEQ13 FRONT AGE3
{NANNY, NANNY} ADD20
DB FFH ; end

TABLE M047:

DB 1 ; motor delay between steps
DB 125,0,130,0,0,90,133 ; SEQ13 FRONT AGE3 (RASBERRY, HE
HE HE )
DB ADD20
DB FFH ; end

TABLE M048:

DB 1 ; motor delay between steps
DB 200,0,0,133 ; SEQ16 FRONT AGE3
DB FFH ; end

TABLE M049:

DB 1 ; motor delay between steps
DB 120,110,133,115,133 ; SEQ16
DB FFH ; end

TABLE M050:

DB 10 ; motor delay between steps
DB 140,150,133 ; SEQ2 (TICKLE) FRONT AGE4
DB FFH ; end

; Tbl1_M051:
DB FFH ; end
DB 125,100,133 ; SEQ2 (AGAIN) FRONT AGE4

; Tbl1_M052:
DB 1 ; motor delay between steps
DB 120,133 ; SEQ3 (YOU) FRONT AGE4

; Tbl1_M053:
DB FFH ; end
DB 160,133 ; SEQ3 (ME) FRONT AGE4

; Tbl1_M054:
DB 20 ; motor delay between steps
DB 150,133 ; SEQ4 (LOVE) FRONT AGE4 ADD45 74 71 20

; Tbl1_M055:
DB FFH ; end
DB 125,133,150,0,0,133 ; SEQ5 (ME ME ME) FRONT AGE4

; Tbl1_M056:
DB 10 ; motor delay between steps
DB 154,133,115,0,0,0,0,0,0,133 ; SEQ5 (BIG FUN) FRONT AGE4 ADD26

; Tbl1_M057:
DB FFH ; end
DB 120,133 ; SEQ8 (NO) FRONT AGE4

; Tbl1_M058:
DB FFH ; end
DB 100,133 ; SEQ8 (PLEASE) FRONT AGE4

; Tbl1_M059:
DB FFH ; end
DB 150,0,0,0,133 ; SEQ9 (HEEY) FRONT AGE4 ADD71

; Tbl1_M060:
DB FFH ; end
DB 120,100,133 ; SEQ14 (PARTY) AGE4 ADD45

; Tbl1_M061:
DB FFH ; end
DB 143,150,170,133 ; SEQ15 (WA WA WA) FRONT AGE4 ADD22

; END GEORGE 07/03/98
<table>
<thead>
<tr>
<th>Table</th>
<th>DB</th>
<th>Motor Delay Between Steps</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>M062</td>
<td>20</td>
<td>150,0,0,0,133</td>
<td>Motor delay between steps; FORTUNE ASK</td>
</tr>
<tr>
<td></td>
<td>FFH</td>
<td>:end</td>
<td></td>
</tr>
<tr>
<td>M063</td>
<td>1</td>
<td>150,0,0,133</td>
<td>Motor delay between steps; FORTUNE ASK</td>
</tr>
<tr>
<td></td>
<td>FFH</td>
<td>:end</td>
<td></td>
</tr>
<tr>
<td>M064</td>
<td>1</td>
<td>150,0,0,0,133</td>
<td>Motor delay between steps; FORTUNE TELL (BIG)</td>
</tr>
<tr>
<td></td>
<td>FFH</td>
<td>:end</td>
<td></td>
</tr>
<tr>
<td>M065</td>
<td>10</td>
<td>190,150,0,0,133</td>
<td>Motor delay between steps; FORTUNE TELL (VERY, BIG)</td>
</tr>
<tr>
<td></td>
<td>FFH</td>
<td>:end</td>
<td></td>
</tr>
<tr>
<td>M066</td>
<td>1</td>
<td>120,0,0,0,0,0,0,0,133</td>
<td>Motor delay between steps; FORTUNE TELL (SEE)</td>
</tr>
<tr>
<td></td>
<td>FFH</td>
<td>:end</td>
<td></td>
</tr>
<tr>
<td>M067</td>
<td>10</td>
<td>30,10,30,10,30,10,30,70</td>
<td>Motor delay between steps; FORTUNE WHINE START; OK</td>
</tr>
<tr>
<td></td>
<td>FFH</td>
<td>:end</td>
<td></td>
</tr>
<tr>
<td>M068</td>
<td>1</td>
<td>100,133,150,133,150,133</td>
<td>Motor delay between steps; FORTUNE WHINE START</td>
</tr>
<tr>
<td></td>
<td>FFH</td>
<td>:end</td>
<td></td>
</tr>
<tr>
<td>M069</td>
<td>1</td>
<td>150,133</td>
<td>Motor delay between steps; FORTUNE TELL (NO)</td>
</tr>
<tr>
<td></td>
<td>FFH</td>
<td>:end</td>
<td></td>
</tr>
<tr>
<td>M070</td>
<td>1</td>
<td>125,100,133</td>
<td>Motor delay between steps; FORTUNE TELL (WORRY)</td>
</tr>
<tr>
<td></td>
<td>FFH</td>
<td>:end</td>
<td></td>
</tr>
<tr>
<td>M071</td>
<td>10</td>
<td>110,120,133</td>
<td>Motor delay between steps; FORTUNE (SOUND)</td>
</tr>
<tr>
<td></td>
<td>FFH</td>
<td>:end</td>
<td></td>
</tr>
<tr>
<td>M072</td>
<td>1</td>
<td>150,133</td>
<td>Motor delay between steps; FORTUNE (GOOD)</td>
</tr>
<tr>
<td></td>
<td>FFH</td>
<td>:end</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M073:</td>
<td>DB 1</td>
<td>;motor delay between steps</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DB 150,0,133</td>
<td>;FORTUNE TELL (VERY)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DB FFH</td>
<td>;end</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M074:</td>
<td>DB 1</td>
<td>;motor delay between steps</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DB 145,133,150,0,0,0,0,133</td>
<td>;FORTUNE (WHOOPPEE)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DB FFH</td>
<td>;end</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M075:</td>
<td>DB 1</td>
<td>;motor delay between steps</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DB 115,133</td>
<td>;FORTUNE (GOOD)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DB FFH</td>
<td>; end</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M076:</td>
<td>DB 1</td>
<td>;motor delay between steps</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DB 120,0,0,0,0,133</td>
<td>;FORTUNE (RASPBERRY)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DB FFH</td>
<td>;end</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M077:</td>
<td>DB 1</td>
<td>;motor delay between steps</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DB 150,115,133</td>
<td>;FORTUNE (OH OH)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DB FFH</td>
<td>;end</td>
<td></td>
</tr>
<tr>
<td>Tbl1_M078:</td>
<td>DB 1</td>
<td>;motor delay between steps</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DB 150,115,133</td>
<td>;FORTUNE (HAY BEE)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DB FFH</td>
<td>;end</td>
<td></td>
</tr>
</tbody>
</table>

;END GEORGE 07/04/98
;START HANGOUT
;GEORGE 07/04/98

| Tbl1_M079: | DB 1 | ;motor delay between steps |
|            | DB 150,133,135,150,133 | ;SEQ1 HANGING (DE DE DE DUM DUM DUM DUM)
| DB FFH | ;end |
| Tbl1_M080: | DB 1 | ;motor delay between steps |
|            | DB 190,133 | ;SEQ1 HANGING (DUM DUM DUM DUM DUM DUM) |
| AGE1 DB FFH | ;end |
| Tbl1_M081: | DB 1 | ;motor delay between steps |
|            | DB 120,100,133 | ;SEQ1 HANGING (beedo) |
|            | DB 120,100,133 | ;end |
| Tbl1_M082: | DB 1 | ;motor delay between steps |
|            | DB 141,150,170,0,0,0,0,190 | ;-133 |
|            | DB 120,100,160,133 | ;SEQ1 HANGING (YA DA DA)
| DB FFH | ;end |
| Tbl1_M083: | DB 1 | ;motor delay between steps |

A-260
DB 190, 120, 133 ; SEQ3 HANGING (ROPAH BRUMM BABAUM)
DB FFH ; end

Tbll_M084:
DB 10 ; motor delay between steps
DB 125, 120, 125, 115, 133 ; SEQ3 HANGING (BRRRUM BABAUM)
DB FFH ; end

Tbll_M085:
DB 1 ; motor delay between steps
DB 115, 125, 110, 125, 100, 133 ; SEQ4 HANGING (LA LA)
DB FFH ; end

Tbll_M086:
DB 1 ; motor delay between steps
DB 120, 130, 115 ; SEQ4 HANGING (LA LA)
DB 100, 125, 115, 125, 115, 125, 115, 125, 115, 133
DB FFH ; end

Tbll_M087:
DB 10 ; motor delay between steps
DB 115, 125, 130, 135, 140, 133 ; SEQ5 HANGING (HUMM BO DAH WAY-LOH)
DB FFH ; end

Tbll_M088:
DB 10 ; motor delay between steps
DB 115, 125, 130, 135, 140, 133 ; SEQ5 HANGING (HUMM BO DAH WAY-LOH)
DB 115, 133, 160, 133 ; SEQ5 HANGING (HUMM BO DAH WAY-LOH)
DB FFH ; end

Tbll_M089:
DB 60 ; motor delay between steps
DB 190, 170, 150, 133, 0, 0, 0, 0, 0, 0 ; SEQ5 HANGING (SNORE)
DB FFH ; end

Tbll_M090:
DB 10 ; motor delay between steps
DB 150, 133 ; SEQ6 HANGING (SHOUT)
DB FFH ; end

Tbll_M091:
DB 1 ; motor delay between steps
DB 143, 150, 140, 0, 150, 0, 0, 0, 133 ; SEQ6 HANGING (OK RAH)
DB FFH ; end

Tbll_M092:
DB 5 ; motor delay between steps
DB 110, 133 ; SEQ6 HANGING (U-TYE)
DB FFH ; end
Tbll_M093:
DB  60 ; motor delay between steps
DB  190,180,170,150,133 ; SEQ7 HANGING (SOFTER)
DB  FFH ; end
;
; danger sleep
Tbll_M094:
DB  50 ; motor delay between steps
DB  190,170,150,10 ; SEQ7 HANGING (SOFTER)
DB  FFH ; end
;
Tbll_M095:
DB  20 ; motor delay between steps
DB  145,133,115,133 ; SEQ8 HANGING ADD 76
DB  FFH ; end
;
Tbll_M096:
DB  1 ; motor delay between steps
DB  150,115,150,133 ; SEQ9 HANGING (DO BE DOBE DO)
DB  FFH ; end
;
Tbll_M097:
DB  46 ; motor delay between steps
DB  170,150,150,133 ; SEQ10 HANGING (YAWN)
DB  FFH ; end
;
Tbll_M098:
DB  25 ; motor delay between steps
DB  150,133 ; SEQ11 AND SEQ12 HANGING (SIGH)
DB  FFH ; end
;
Tbll_M099:
DB  1 ; motor delay between steps
DB  144,133 ; SEQ13 SEQ14 HANGING (HA)
DB  FFH ; end
;
Tbll_M100:
DB  10 ; motor delay between steps
DB  104,0,0,133 ; SEQ16 HANGING (PAUSE) ADD20
DB  FFH ; end
;
Tbll_M101:
DB  20 ; motor delay between steps
DB  100,133,0,0,100,133 ; SEQ16
DB  FFH ; end
;
anger, USED IN ONE CASE, HANGING OUT, FOLLOWED BY 101
Tbll_M102:
DB  10 ; motor delay between steps
DB  0 ; SEQ16 HANGING (PAUSE) ADD20
DB  FFH ; end
;
Tbll_M103:
DB  1 ; motor delay between steps
DB  114,133 ; SEQ6 HANGING (UP)
DB  FFH ; end
;
Tbll_M104:
DB  1 ; motor delay between steps
DB  1
DB 120,133 ;SEQ6 HANGING (ME)
DB FFH ;end

Tbl1_M105:
DB 1 ;motor delay between steps
DB 120,133 ;UP
DB FFH ;end

Tbl1_M106:
DB 10 ;motor delay between steps
DB 125,104,133 ;SEQ5 BORING
DB FFH ;end

END HANGOUT

; danger, OK PAUSE FOR FORTUNE TELLING
Tbl1_M107:
DB 1 ;motor delay between steps
DB FFH ;end ;Fortune pause

;END GEORGE 07/04/98
;FEED
;GEORGE 07/05/98
Tbl1_M108:
DB 10 ;motor delay between steps
DB 115,0,0,0,0,0,0,0,0,0,0,0,0,0,0
DB 133 ;SEQ1 FEED AGE1 (Ummmm)
DB FFH ;end

Tbl1_M109:
DB 1 ;motor delay between steps
DB 140
DB 165,0,0,0,0,0,0,150,0,0,165,0,0,0,0,0,0,0,133 ;SEQ1

FEED AGE1 (AAAAA)
DB FFH ;end

Tbl1_M110:
DB 1 ;motor delay between steps
DB 120,130,110,133 ;SEQ2 FEED AGE1 (KOH KOH)
DB FFH ;end

Tbl1_M111:
DB 1 ;motor delay between steps
DB 120,130,120,133 ;ME ME
DB FFH ;end

Tbl1_M112:
DB 1 ;motor delay between steps
DB 145,133,150,133 ;E-DAY
DB FFH ;end

Tbl1_M113:
DB 1 ;motor delay between steps
DB 115,130,110,133 ;DO MOH
DB FFH ;end

Tbl1_M114:
DB 1 ;motor delay between steps
DB 115,130,120,133 ;TOH DYE
DB FFH ;end

Tbll_M115:
DB 10 ;motor delay between steps
DB 110,133
DB FFH ;end

Tbll_M116:
DB 1 ;motor delay between steps
DB 145,133 ;SIGH
DB FFH ;end

Tbll_M117:
DB 10 ;motor delay between steps
DB 150,133
DB FFH ;end

Tbll_M118:
DB 10 ;motor delay between steps
DB 120,0,0,0,133
DB FFH ;end

Tbll_M119:
DB 1 ;motor delay between steps
DB 120,130,110,133 ;TOH LOO
DB FFH ;end

Tbll_M120:
DB 1 ;motor delay between steps
DB 120,133,120,133
DB FFH ;end

Tbll_M121:
DB 1 ;motor delay between steps
DB 145,130,120,133 ;HUNGRY
DB FFH ;end

Tbll_M122:
DB 1 ;motor delay between steps
DB 150,133 ;LIKE
DB FFH ;end

Tbll_M123:
DB 1 ;motor delay between steps
DB 150,0,0,133 ;seq4 feed done
DB FFH ;end

; END FEED
; END GEORGE 07/05/98
;
; WAKE
;GEORGE 07/06/98
Tbll_M124:
DB 255 ;SG DONE
DB 95,133
DB FFH
;
; danger
<table>
<thead>
<tr>
<th>Table</th>
<th>Motor Delay Between Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tbl1_M125</td>
<td>1 75,90 FFh</td>
</tr>
<tr>
<td>Tbl1_M126</td>
<td>1 135,120,135 FFh</td>
</tr>
<tr>
<td>Tbl1_M127</td>
<td>1 80,133 FFh</td>
</tr>
<tr>
<td>Tbl1_M128</td>
<td>1 75,90 FFh</td>
</tr>
<tr>
<td>Tbl1_M129</td>
<td>1 90,110,133 FFh</td>
</tr>
<tr>
<td>Tbl1_M130</td>
<td>1 115,133 FFh</td>
</tr>
<tr>
<td>Tbl1_M131</td>
<td>1 90,70 FFh</td>
</tr>
<tr>
<td>Tbl1_M132</td>
<td>1 95,133 FFh</td>
</tr>
<tr>
<td>Tbl1_M133</td>
<td>1 115,133 FFh</td>
</tr>
<tr>
<td>Tbl1_M134</td>
<td>1 185 FFh</td>
</tr>
<tr>
<td>Tbl1_M135</td>
<td>1 133 FFh</td>
</tr>
<tr>
<td>Tbl1_M136</td>
<td>1 133 FFh</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table</th>
<th>Motor Delay Between Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tbl2_M125</td>
<td>1 75,90</td>
</tr>
<tr>
<td>Tbl2_M126</td>
<td>1 135,120,135</td>
</tr>
<tr>
<td>Tbl2_M127</td>
<td>1 80,133</td>
</tr>
<tr>
<td>Tbl2_M128</td>
<td>1 75,90</td>
</tr>
<tr>
<td>Tbl2_M129</td>
<td>1 90,110,133</td>
</tr>
<tr>
<td>Tbl2_M130</td>
<td>1 115,133</td>
</tr>
<tr>
<td>Tbl2_M131</td>
<td>1 90,70</td>
</tr>
<tr>
<td>Tbl2_M132</td>
<td>1 95,133</td>
</tr>
<tr>
<td>Tbl2_M133</td>
<td>1 115,133</td>
</tr>
<tr>
<td>Tbl2_M134</td>
<td>1 185</td>
</tr>
<tr>
<td>Tbl2_M135</td>
<td>1 133</td>
</tr>
<tr>
<td>Tbl2_M136</td>
<td>1 133</td>
</tr>
</tbody>
</table>

A-265
; SG DONE
DB 1 ; motor delay between steps
DB 145
DB FFh
; danger
Tbl2_M138:
DB 1 ; motor delay between steps
DB 120,133,120,133,120,133,120,133,120,133,120,133,77,8
DB 0,0,70,0,0,0,0,0,0,0,0,0,0,0,0
DB FFh
; danger
Tbl2_M139:
DB 1 ; motor delay between steps
DB 82,70
DB FFh
; danger
Tbl2_M140:
DB 1 ; motor delay between steps
DB 120,115,130,120,70
DB FFH ; end
; danger
Tbl2_M141:
DB 1 ; motor delay between steps
DB 133
DB FFH ; end
; danger
Tbl2_M142:
DB 1 ; motor delay between steps
DB 75
DB FFH ; end
; danger
Tbl2_M143:
DB 1 ; motor delay between steps
DB 90,80,100,75
DB 90,80,100,133
DB FFH ; end
; danger
Tbl2_M144:
DB 1 ; motor delay between steps
DB 120
DB FFH ; end
; danger
Tbl2_M145:
DB 1 ; motor delay between steps
DB 110,75
DB FFH ; end
; danger
Tbl2_M146:
DB 1 ; motor delay between steps
DB 90,75
DB 90,133
DB FFH ; end
; danger
Tbl2_M147:
DB 1 ; motor delay between steps
DB 70,90,75
DB FFH ;end

Tbl2_M148: ;SG DONE
DB 1 ;motor delay between steps
DB 120,130,115,126,115,140,110,0,0,0,0,0,0,0,0,113
DB FFH ;end

; danger
Tbl2_M149: ;SG DONE
DB 1 ;motor delay between steps
DB 75
DB FFH ;end

Tbl2_M150: ;SG DONE
DB 1 ;motor delay between steps
DB 146,135
DB FFH ;end

Tbl2_M151: ;SG DONE
DB 1 ;motor delay between steps
DB 120,133,70,0,135
DB FFH ;end

; danger
Tbl2_M152: ;SG DONE
DB 1 ;motor delay between steps
DB 75
DB FFH ;end

Tbl2_M153: ;SG DONE
DB 1 ;motor delay between steps
DB 115,75
DB FFH ;end

; danger sleep
Tbl2_M154: ;SG DONE
DB 100 ;motor delay between steps
DB 0,0,0,85,30,0,20,0,85,30,0,20,0,85,30,0,20,0,75,0,0,0,0,85
DB 30,0,20,0,10
DB FFH ;end

; danger
Tbl2_M155: ;SG DONE
DB 1 ;motor delay between steps
DB 90,70
DB FFH ;end

; danger
Tbl2_M156: ;SG DONE
DB 1 ;motor delay between steps
DB 115,75
DB FFH ;end

; END WAKE
; END GEORGE 07/06/98

; HUNGER
; GEORGE 07/06/98

Tbl2_M157: ;SG DONE ; HUNGER
DB 50 ;motor delay between steps
;DB 120,120,133
DB 120,0,133
DB FFH ;end

; SG DONE
Tbl2_M158:
DB 1 ;motor delay between steps
DB 180,133
DB FFH ;end

; SG DONE
Tbl2_M159:
DB 1 ;motor delay between steps
DB 115,110,133
DB FFH ;end

; SG DONE
Tbl2_M160:
DB 1 ;motor delay between steps
DB 75,133
DB FFH ;end

; SG DONE
Tbl2_M161:
DB 1 ;motor delay between steps
DB 115,130,115,1
DB FFH ;end

; SG DONE
Tbl2_M162:
DB 1 ;motor delay between steps
DB 115,110,133
DB FFH ;end

; SG DONE
Tbl2_M163:
DB 50 ;motor delay between steps
DB 190,133
DB FFH ;end

; SG DONE
Tbl2_M164:
DB 50 ;motor delay between steps
;DB 148,148,133
DB 148,0,133
DB FFH ;end

; SG DONE
Tbl2_M165:
DB 50 ;motor delay between steps
;DB 150,150,150,133
DB 150,0,0,133
DB FFH ;end

; SG DONE
Tbl2_M166:
DB 1 ;motor delay between steps
DB 120,133
DB FFH ;end

; SG DONE
Tbl2_M167:
DB 1 ;motor delay between steps
DB 115,133
DB FFH ;end

; SG DONE
Tbl2_M168:
DB 1
DB 115, 133
DB FFh

;END GEORGE 07/06/98
;END HUNGER

; INVERT
; GEORGE 07/07/98
Tbl2_M169: ; SG DONE ; INVERT
DB 1 ; motor delay between steps
DB 110, 122, 75, 130, 117, 133
DB FFh ; end

Tbl2_M170: ; SG DONE
DB 10 ; motor delay between steps
DB 165, 165, 133
DB 165, 0, 133
DB FFh ; end

Tbl2_M171: ; SG DONE
DB 10 ; motor delay between steps
DB 105, 133
DB FFh ; end

Tbl2_M172: ; SG DONE
DB 1 ; motor delay between steps
DB 150, 133
DB FFh ; end

Tbl2_M173: ; SG DONE
DB 1 ; motor delay between steps
DB 155, 190, 133
DB FFh ; end

Tbl2_M174: ; SG DONE
DB 1 ; motor delay between steps
DB 145, 133
DB FFh ; end

Tbl2_M175: ; SG DONE
DB 1 ; motor delay between steps
DB 150, 135, 145, 133
DB FFh ; end

Tbl2_M176: ; SG DONE
DB 1 ; motor delay between steps
DB 75, 133
DB FFh ; end

Tbl2_M177: ; SG DONE
DB 1 ; motor delay between steps
DB 110, 133, 115, 133
DB FFh ; end

Tbl2_M178: ; SG DONE
DB 1 ; motor delay between steps
DB 115, 133
DB FFh ; end
; Tbl2_M179:
  DB 1 ;motor delay between steps
  DB 115,133
  DB FFH ;end

; Tbl2_M180:
  DB 1 ;motor delay between steps
  DB 110,125,115,133
  DB FFH ;end

; Tbl2_M181:
  DB 1 ;motor delay between steps
  DB 150,133
  DB FFH ;end

; Tbl2_M182:
  DB 1 ;motor delay between steps
  DB 115,133
  DB FFH ;end

; Tbl2_M183:
  DB 1 ;motor delay between steps
  DB 115,130,110,133
  DB FFH ;end

; Tbl2_M184:
  DB 1 ;motor delay between steps
  DB 150,133
  DB FFH ;end

; Tbl2_M185:
  DB 1 ;motor delay between steps
  DB 150,150,133
  DB FFH ;end

; Tbl2_M186:
  DB 1 ;motor delay between steps
  DB 115,130,115,133
  DB FFH ;end

; Tbl2_M187:
  DB 1 ;motor delay between steps
  DB 115,130,115,133
  DB FFH ;end

; Tbl2_M188:
  DB 1 ;motor delay between steps
  DB 145,135,145,133
  DB FFH ;end

; Tbl2_M189:
  DB 1 ;motor delay between steps
  DB 120,105,133
  DB FFH ;end

; Tbl2_M190:
DB 1 ;motor delay between steps
DB 155,190,133
DB FFH ;end

Tbl2_M191:
DB 1 ;motor delay between steps
DB 155,190,133
DB FFH ;end

Tbl2_M192:
DB 1 ;motor delay between steps
DB 155,190,133
DB FFH ;end

;END GEORGE 07/07/98
;END INVERT

;start at 202
Tbl2_M193: ;BACKSG ;SG DONE
DB 100 ;motor delay between steps
DB 200,200,200,200,133
DB 200,0,0,0,133
DB FFH ;end

Tbl2_M194:
DB 1 ;SG DONE
DB 75,133
DB FFH ;end

Tbl2_M195:
DB 1 ;SG DONE
DB 115,125,115,133
DB FFH ;end

Tbl2_M196:
DB 10 ;motor delay between steps
DB 145,133
DB FFH ;end

Tbl2_M197:
DB 1 ;SG DONE
DB 115,125,115,133
DB FFH ;end

Tbl2_M198:
DB 100 ;motor delay between steps
DB 145,0,0,133
DB FFH ;end

Tbl2_M199:
DB 10 ;motor delay between steps
DB 110,133
DB FFH ;end

Tbl2_M200:
DB 1 ;motor delay between steps
DB 75,133
DB FFH ;end

Tbl2_M201:
DB 10 ;motor delay between steps
DB 115,125,115,133
DB FFH ;end

Tbl2_M202:
DB 75,133
DB FFH ;end
;

danger
Tbl2_M203:
DB 120,128,79,133,146,0,0,0,133,145
DB FFH ;end

Tbl2_M204:
DB 190,0,133
DB FFH ;end

Tbl2_M205:
DB 115,133
DB FFH ;end
;

danger
Tbl2_M206:
DB 75
DB FFH ;end

Tbl2_M207:
DB 150
DB FFH ;end

Tbl2_M208:
DB 75,133
DB FFH ;end

Tbl2_M209:
DB 150,0,0,0,133
DB FFH ;end

Tbl2_M210:
DB 123,110,75,133,115,133
DB FFH ;end
;

danger
Tbl2_M211:
DB 1
DB FFH ;end
;

danger
Tbl2_M212:
DB 133
DB FFH ;end
Tbl2_M213:
DB 10 ;SG DONE
DB 115,150,133 ;motor delay between steps
DB FFH ;end

Tbl2_M214:
DB 1 ;SG DONE
DB 80,133 ;motor delay between steps
DB FFH ;end

; danger
Tbl2_M215:
DB 100 ;motor delay between steps
DB 138
DB FFH ;end

Tbl2_M216:
DB 10 ;motor delay between steps
DB 75,133
DB FFH ;end

Tbl2_M217:
DB 1 ;motor delay between steps
DB 115,130,115,133
DB FFH ;end

Tbl2_M218:
DB 50 ;motor delay between steps
DB 114,133
DB FFH ;end

Tbl2_M219:
DB 10 ;motor delay between steps
DB 120,130,120,130,120,130,120,130,120,130,120,130,120,130,120,130,115,115,115,133
DB 120,130,120,130,120,130,120,130,120,130,120,130,120,130,120,130,115,0,133
DB FFH ;end

Tbl2_M220:
DB 10 ;motor delay between steps
DB 120,130,120,130,120,130,120,130,120,130,120,130,120,130,120,130,115,0,133
DB FFH ;end

Tbl2_M221:
DB 10 ;motor delay between steps
DB 145,133
DB FFH ;end

Tbl2_M222:
DB 50 ;motor delay between steps
DB 0,0,0,0,115,133
DB FFH ;end

Tbl2_M223:
DB 1 ;motor delay between steps
DB 115,125,115,133

A-273
; danger
Tbl2_M233:
  DB 1
  DB 145
  DB FFH ;end

; danger sleep
Tbl2_M234:
  DB 10
  DB 10
  DB FFH ;end

Tbl2_M235:
  DB FFH ;end
motor delay between steps
motor delay between steps
motor delay between steps
motor delay between steps
motor delay between steps
motor delay between steps
danger
motor delay between steps
motor delay between steps
motor delay between steps
;DB 70,133
;DB FFH ;end
;
Tbl2_M247:
DB 1^ ;motor delay between steps
DB 110,133,0,0
DB FFH ;end
;
Tbl2_M248:
DB 10 ;motor delay between steps
DB 145,0,0,0,133
DB FFH ;end
;
Tbl2_M249:
DB 1 ;motor delay between steps
DB 115,0,0,0,133
DB FFH ;end
;
Tbl2_M250:
DB 10 ;motor delay between steps
;DB 150,150,150,190,0,133
DB 150,0,0,190,0,133
DB FFH ;end
;GEORGE 07/08/98
;LIGHT
;
Tbl2_M251:
DB 5 ;motor delay between steps SGTEST
DB 115,132,125,110,132
DB FFH
;
Tbl2_M252:
DB 1 ;motor delay between steps
DB 190,133
DB FFH
;
Tbl2_M253:
DB 1 ;motor delay between steps
DB 10,152,133,160,0,133
DB FFH
;
Tbl2_M254:
DB 1 ;motor delay between steps
;DB 143,137,137,137,150,133,155,133
DB 143,137,143,137,150,0,0,0,133,155,133
DB FFH
;
Tbl2_M255:
DB 1 ;motor delay between steps
DB 60,90,60,85,90,60,90,133
DB FFH
;
Tbl3_M256:
DB 10 ;motor delay between st DONE RB
DB 180,165,165,133
DB FFH
;
Tbl3_M257:
DB 10 ;motor delay between steps
DB 190,133,105,133,105,160,133 ;WOW DONE
DB FFH
;
Tbl3_M258:
DB 4 ;motor delay between steps DONE
DB 60,133,0,0,0,0,0,155,133,145,133
Tbl3_M259:
  DB 1 ;motor delay between steps
  DB 160,133,180,133,147,160,133
  DB FFh

Tbl3_M260:
  DB 1 ;motor delay between steps
  DB 160,133,90,133
  DB FFh

Tbl3_M261:
  DB 7 ;motor delay between steps
  DB 190,133,100,133
  DB FFh

Tbl3_M262:
  DB 7 ;motor delay between steps
  DB 60,133,160,153,0,0,133,150,133
  DB FFh

Tbl3_M263:
  DB 1 ;MOTOR DELAY BETWEEN STEPS
  DB 155,133,160,133,120,110,133
  DB FFh

Tbl3_M264:
  DB 10 ;motor delay between steps
  DB 190,133,0,0,0,0,0,0,133
  DB FFh

Tbl3_M265:
  DB 1 ;motor delay between steps
  DB 60,133,180,133
  DB FFh

;END LIGHT
;END GEORGE 07/06/98

;DARK
;GEORGE 07/08/98

Tbl3_M266:
  DB 1 ;motor delay between steps
  DB 150,133,160,133,120,0,0,0,0,0,0,133
  DB FFh

Tbl3_M267:
  DB 1 ;motor delay between steps DONE RB
  DB 150,133,120,112,0,0,0,0,0,133,149,0,0,133
  DB FFh

Tbl3_M268:
  DB 10 ;motor delay between steps
  DB 150,133,112,133,120,133,148,133,118,0,0,0,0,0,0,133
  DB 147,0,0,0,0,0,133
  DB FFH ;end DONE RB

Tbl3_M269:
  DB 1 ;motor delay between steps DONE RB
  DB 10,20,123,115,123,115,123,115,133
  DB FFH ;end

Tbl3_M270:
  DB 1 ;motor delay between steps DONE
  DB 190,133,120,133,112,0,0,0,0,0,0,130,112,133
  DB FFH ;end
; Tbl3_M271:
  DB 1 ; motor delay between steps
  DB 147,155,139,149 ; SEQ6 AGE4/SEQ14 AGE 4 LIGHT
  DB 133,149,0,0,0,133 ;
  DB FFH ; end

; Tbl3_M272:
  DB 1 ; motor delay between steps
  DB 150,133,0,0,0,159,133,150,0,0,133
  DB 145,137,144,133,117,125,117,133
  DB FFH ; end DONE

; Tbl3_M273:
  DB 1 ; motor delay between steps
  DB 145,155,133,120,115,133,190,133
  DB 0,0,0,150,0,0,0,0,0,0,0,133
  DB 0,0,0,0,0,0,0,0,0,0,0,115,133
  DB FFH ; end

; Tbl3_M274:
  DB 1 ; motor delay between steps
  DB 150,133,150,0,0,0,133,0,0,0,120,115,0,0,0,0,0,133
  DB FFH ; end

; Tbl3_M275:
  DB 10 ; motor delay between steps
  DB 150,133,0,0,0,150,0,0,133,0,120,133,120,133,155,0,0,0,0,133
  DB FFH ; end

; Tbl3_M276:
  DB 1 ; motor delay between steps
  DB 190,0,0,0,0,133,0,0,0,0,0,148,133,118,133,0,0,0
  DB 146,133,147,0,0,0,0,0,0,133
  DB FFH ; end

; Tbl3_M277:
  DB 1 ; motor delay between steps
  DB 190,133,120,133,112,0,0,0,0,0,130,112,133
  DB FFH ; end

; Tbl3_M278:
  DB 1 ; motor delay between steps
  DB 60,133,60,133,146,154,133
  DB FFH ; end

; Tbl3_M279:
  DB 1 ; motor delay between steps
  DB 190,133,0,0,0,110,0,0,0,0,0,133
  DB FFH ; end

; Tbl3_M280:
  DB 10 ; motor delay between steps
  DB 150,133,0,0,0,116,0,0,0,133,190,155,0,0,133
  DB FFH ; end

; Tbl3_M281:
  DB 1 ; motor delay between steps
  DB 190,155,0,0,0,133,119,0,0,0,0,0,133
DB 146, 133, 147, 0, 0, 0, 0, 0, 133
DB FFH ; end

Tbl3_M282:
DB 1 ; motor delay between steps
DB 60, 133, 75, 83, 78, 83, 78, 133
DB FFH ; end

Tbl3_M283:
DB 1 ; motor delay between steps
DB 145, 155, 133, 120, 115, 133, 72, 0, 0, 0, 0, 92, 133, 190, 133
DB FFH ; end

Tbl3_M284:
DB 1 ; motor delay between steps
DB 190, 133, 0, 0, 0, 110, 0, 0, 0, 0, 133
DB FFH ; end

Tbl3_M285:
DB 10 ; motor delay between steps
DB 150, 133, 0, 0, 0, 116, 0, 0, 0, 133, 190, 155, 0, 0, 0, 133
DB FFH ; end

Tbl3_M286:
DB 1 ; motor delay between steps
DB 190, 155, 0, 0, 0, 133, 119, 0, 0, 0, 0, 0, 0, 133
DB 147, 0, 0, 0, 0, 0, 133
DB FFH ; end

Tbl3_M287:
DB 1 ; motor delay between steps
DB 190, 133, 110, 0, 0, 0, 0, 133, 112, 0, 0, 133
DB FFH ; end

Tbl3_M288:
DB 1 ; motor delay between steps
DB 110, 0, 0, 0, 133, 115, 133, 147
DB 133, 190, 133 ; SEQ7 AGE4/SEQ15 AGE 4 LIGHT js
DB FFH ; end

Tbl3_M289:
DB 1 ; motor delay between steps
DB 145, 155, 133, 0, 0, 0, 0, 0, 120, 115, 133, 350, 133
DB 160, 0, 0, 0, 0, 190, 0, 0, 0, 0, 0, 0, 0, 0, 133
DB 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 133 ; SEQ8 AGE4/SEQ 16 AGE 4

INVERT js
DB FFH ; end

; END GEORGE 07/08/98
; END DARK

; SOUND
Tbl3_M290:
DB 1 ; motor delay between steps
DB 155, 133, 0, 0, 0, 0, 125
DB 115, 145, 155, 133 ; S1-A1/S9-A1/S1-A2 SOUND js
DB FFH ; end ; S9-A2/S1-A3/S9-A3 SOUND js

Tbl3_M291:
DB 1 ; motor delay between steps
DB 100, 0, 0, 0, 10
| DB | 0,0,0,0,0,0,0,0 | ;S2-A1/S10-A1/S2-A2 SOUND js |
| DB | 0,0,0,70,0,0,0,0,0 | ;S10-A2/S2-A3/S10-A3 SOUND js |
| DB | 0,0,100,0,0,0,133 | ;S2-A4 SOUND js |
| DB | FFH ; end | ;S2-A4 SOUND js |

Tbl3_M292:

| DB | 1 | ;motor delay between steps |
| DB | 110,0,0,133,0,0,0,0 | |
| DB | 0,0,155,0,0,0,0 | |
| DB | 133,120,0,112,0 | |
| DB | 146,0,0,0,0,0,133 | ;S3-A1/S11-A1 SOUND js |
| DB | FFH ; end | ;S3-A1/S11-A1 SOUND js |

Tbl3_M293:

| DB | 15 | ;motor delay between steps |
| DB | 110,0,120,0,0,0,0,0 | |
| DB | 145,0,0,0,155,115 | |
| DB | FFH ; end | ;S4-A1/S12-A1 SOUND js |

Tbl3_M294:

| DB | 1 | ;motor delay between steps |
| DB | 115,0,0,0,148 | |
| DB | 115,0,0,133 | ;S5-A1/S13-A1 LIGHT (with say/m2) js |
| DB | FFH ; end | ;S5-A1/S13-A1 LIGHT (with say/m2) js |

Tbl3_M295:

| DB | 1 | ;motor delay between steps |
| DB | 155,133,122,0 | ;S6-A1/S14-A1/S6-A2 SOUND js |
| DB | 115,145,120,0,0,133 | ;S14-A2/S6-A3/S14-A3 SOUND js |
| DB | FFH ; end | ;S14-A2/S6-A3/S14-A3 SOUND js |

Tbl3_M296:

| DB | 1 | ;motor delay between steps |
| DB | 14,150 | |
| DB | 125,115 | |
| DB | 0,0,0,0,133 | ;S7-A1/S15-A1 SOUND (with say/m2) js |
| DB | FFH ; end | ;S7-A1/S15-A1 SOUND (with say/m2) js |

Tbl3_M297:

| DB | 1 | ;motor delay between steps |
| DB | 115,0,0,148,0,0,0,0 | |
| DB | 136,0,0,0,148,0,0,0 | |
| DB | 0,0,0,0,133 | ;S8-A1/S16-A1/S8-A3/S16-A3 SOUND js |
| DB | FFH ; end | ;S8-A1/S16-A1/S8-A3/S16-A3 SOUND js |

Tbl3_M298:

| DB | 1 | ;motor delay between steps |
| DB | 110,0,0,133,0,0,0,0 | |
| DB | 0,0,155,0,0,0,0 | |
| DB | 133,120,0,112,0 | |
| DB | 148,0,0,0,0,0,133 | ;S3-A2/S11-A2 SOUND js |
| DB | FFH ; end | ;S3-A2/S11-A2 SOUND js |

Tbl3_M299:

| DB | 1 | ;motor delay between steps |
| DB | 110,0,120,0,0,0,0,0 | |
| DB | 145,0,0,0,155,190 | |
| DB | 0,0,0,0,0,160,0,133 | ;S4-A2/S12-A2 SOUND js |
| DB | FFH ; end | ;S4-A2/S12-A2 SOUND js |
; Tbl3_M300:
  DB 1 ;motor delay between steps
  DB 165,0,0,190,0,0 ;S5-A2/S13-A2 SOUND (with
  say/m2) js
  DB 0,0,165,0,0,0,133 ;S5-A3/S13-A3 SOUND (with
  say/m2) js
  DB FFH ;end ;S5-A4 SOUND (with say/m2) js
; Tbl3_M301:
  DB 1 ;motor delay between steps
  DB 115,0,0,0,145,0,0,165 ;S7-A2/S15-A2 SOUND (with
  say/m2) js
  DB 0,0,190,165,0,0,0,133
  DB FFH ;end
; Tbl3_M302:
  DB 1 ;motor delay between steps
  DB 115,0,0,148,0,0,0,0
  DB 0,0,0,133 ;S8-A2/S16-A2 SOUND js
  DB FFH ;end
; Tbl3_M303:
  DB 1 ;motor delay between steps
  DB 110,0,0,133,0,0,0,0
  DB 0,0,155,0,0
  DB 133,0,112,0
  DB 148,0,0,0,0,133 ;S3-A3/S11-A3 SOUND js
  DB FFH ;end
; Tbl3_M304:
  DB 1 ;motor delay between steps
  DB 110,0,120,0,0,0,0,0,0,160;
  DB 160,0,0,0,190
  DB 160,0,0,0,0,133 ;S4-A3/S12-A3 SOUND js
  DB FFH ;end ;S4-A4 SOUND js
; Tbl3_M305:
  DB 1 ;motor delay between steps
  DB 115,0,0,0,0,0,160
  DB 0,0,190,0,0,0,0
  DB 0,165,133 ;S7-A3/S15-A3 SOUND (with say/m2) js
  DB FFH ;end ;S7-A4 SOUND (with say/m2) js
; Tbl3_M306:
  DB 1 ;motor delay between steps
  DB 157,0,0,0,133
  DB 0,0,120,0,0,0
  DB 133,150,0,0,0,0,133 ;S1-A4 SOUND js
  DB FFH ;end
; Tbl3_M307:
  DB 1 ;motor delay between steps
  DB 110,0,0,133,0,0,0,0
  DB 0,0,155,0,0
  DB 133,0,112,0,0,0
  DB 148,0,0,0,0,0,0,0,0,133 ;S3-A4 SOUND js
  DB FFH ;end
; Tbl3_M308:
DB 1 ;motor delay between steps
DB 157,0,0,133
DB 0,120,0,0
DB 133,150,0,0,0,0,0,133 ;S6-A4 SOUND js
DB FFh ;end

Tables 3 (M309):
DB 1 ;motor delay between steps
DB 115,0,0,148,0,0,0,0,0,0,0,0
DB 138,0,0,0,0,0,148,0,0,0 ;S8-A1/S4 A1/S2 A4 TILT js
DB FFh

Tables 3 (M310):
DB 1 ;motor delay between steps
DB 170,170,0,0,0
DB 170,0,0,0,0
DB 0,0,0,0,133 ;S1 A1/S4 A1/S2 A4 TILT js
DB FFh

Tables 3 (M311):
DB 1 ;motor delay between steps
DB 125,0,0,133,120,145,110,133 ;S2 A1 TILT js
DB FFh

Tables 3 (M312):
DB 1 ;motor delay between steps
DB 150,133,145,120,133 ;S3 A1 TILT js
DB FFh

Tables 3 (M313):
DB 1 ;motor delay between steps
DB 100,0,0,0,0
DB 0,0,0,0,133 ;S5 A1/S4 A2/S2 A3/S2 A4 TILT js
DB FFh

Tables 3 (M314):
DB 1 ;motor delay between steps
DB 120,100,0,0,0,0,0,70,80,90
DB 70,85,100,0,0,133 ;S6 A1 TILT js
DB FFh

Tables 3 (M315):
DB 1 ;motor delay between steps
DB 125,133,100,133,145,0,0,160
DB 190,0,0,175,160,133 ;S7 A1 TILT/S6 A2 TILT js
DB FFh

Tables 3 (M316):
DB 1 ;motor delay between steps
DB 145,133,145,160,145,160
DB 0,0,0,0,0,190,0,0,0
DB 0,0,0,0,0,0,150,133 ;S8 A1 TILT (with say/m5)
DB FFh

Tables 3 (M317):
DB 10 ;motor delay between steps
DB 160,0,0,0,0,0,0,190,133 ;S9 A1 TILT/S9 A2 TILT
DB FFh

Tables 3 (M318):
DB 10 ;motor delay between steps
DB 145,165,0,0,0,0,0,0,0,0,0,0
DB 190,0,0,160,190,133 ;S10 A1 TILT/S10 A2 TILT js
DB FFh

Tbl3_M319:
DB 1 ;motor delay between steps
DB 0,120,0,0,133,141
DB 133,120,0,0,0,133 ;S11 A1 TILT (with say/m2) js
DB FFh

Tbl3_M320:
DB 1 ;motor delay between steps
DB 150,133,123,0,0,133,142
DB 0,0,150,0,0,0,0,133 ;S12 A1 TILT js
DB FFh

Tbl3_M321:
DB 1 ;motor delay between steps
DB 200,170,170,0,0,0,0,133 ;S13 A1 / S15 A3 TILT js
DB 200,170,0,0,0,0,0,133 ;S13 A1 / S15 A3 TILT js
DB FFh

Tbl3_M322:
DB 1 ;motor delay between steps
DB 170,0,0,0,0,133,126,130,118,133 ;S15 A1 TILT js
DB FFh

Tbl3_M323:
DB 1 ;motor delay between steps
DB 155,0,0,0,155
DB 160,0,0,133 ;S16 A1 / S15 A2 / S13 A3 TILT js
DB FFh

Tbl3_M324:
DB 1 ;motor delay between steps
DB 170,160,0,0,0,0,133 ;S1 A2/S3 A2/S1 A3/S1 A4 TILT
DB FFh

Tbl3_M325:
DB 10 ;motor delay between steps
DB 120,145,110,133 ;S2 A2 TILT (with say/m16) js
DB FFh

Tbl3_M326:
DB 10 ;motor delay between steps
DB 120,100,0,0,0,0,0,0,133
DB 148,133,142,115,0,0,133 ;S5 A2 TILT js
DB FFh

Tbl3_M327:
DB 1 ;motor delay between steps
DB 145,133,145,160,145,160,0,0,0,0
DB 190,0,0,0,0,0,0,0,0
DB 150,133 ;S7 A2 TILT (witl. say/m5) js
DB FFh

Tbl3_M328:
DB 1 ;motor delay between steps
DB 145,0,0,160,0,0,0,0
DB 0,0,0,0,0,0,133 ;S8 A2 TILT (with say/m5) js
DB FFH ;end

Tbl3_M329:
DB 1 ;motor delay between steps
0,120,133,143
118,0,0,0,133 ;S11 A2 TILT (with say/m2) js
FFH ;end

1,150,133,123,0,0,133,142
0,0,150,0,0,0,0,133 ;S12 A2 TILT js
FFH ;end

1,120,150,133 ;S13 A2 TILT (with say/m5) js
FFH ;end

10,155,0,0,0,0,190,0,0,183,0,0,0
175,0,0,162,0,0,0,0,0,0,0,0,133
0,0,120,115,110,110,105,133
145,155,165,0,0,0,0,0,0,0,0,133
0,0,0,0,0,0,133 ;S16 A2/S14 A3/S14 A4 TILT js
FFH ;end

10,120,100,0,0,0,0,0,133 ;S3 A3 TILT js
FFH ;end

1,145,133,120,117
110,0,0,133 ;S4 A3/S4 A4 TILT (with say/m26) js
FFH ;end

1,145,165,0,0,0,0,0,0,0,0,0,133
120,133,145,155,0,0,0,133,115,0,0,0,133
FFH ;end

1,145,133,122,147,139,160
190,0,0,0,0
0,0,0,0,155,133 ;S6 A3 TILT (with say/m5) js
FFH ;end

1,145,165,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,133 ;S7 A3/S7 A4 TILT (with say/m5) js
FFH ;end
Tbl3_M339:
DB 1 ;motor delay between steps
DB 145, 165, 0, 0, 0, 0, 0, 0, 190, 133, 155, 133 ;S8 A3/S8 A4 TILT js
DB FFH ;end

Tbl3_M340:
DB 1 ;motor delay between steps
DB 115, 0, 110, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 133 ;S9 A3/S9 A4 TILT.js
DB FFH ;end

Tbl3_M341:
DB 10 ;motor delay between steps
DB 165, 0, 0, 0, 0, 0, 0, 0, 138, 148, 155
DB 0, 0, 0, 0, 0, 0, 0, 0, 0, 133, 125, 120, 115, 133 ;S16 A3 TILT (with say/m5)
DB FFH ;end

Tbl3_M342:
DB 1 ;motor delay between steps
DB 143, 118, 0, 0, 0, 0, 0, 133 ;S11 A3/S15 A4 TILT (with say/m26 and m34)
DB FFH ;end

Tbl3_M343:
DB 1 ;motor delay between steps
DB 145, 150, 145, 160, 133 ;S12 A3 TILT (with say/m5)
DB FFH ;end

Tbl3_M344:
DB 10 ;motor delay between steps
DB 148, 155, 0, 0, 0, 0, 0, 138, 148, 155
DB 0, 0, 0, 0, 133, 125, 120, 115, 133 ;S16 A3 TILT (with say/m5)
DB FFH ;end

Tbl3_M345:
DB 1 ;motor delay between steps
DB 155, 0, 120, 0, 0, 0, 0, 133 ;S3 A4 TILT (with say/m26)
DB FFH ;end

Tbl3_M346:
DB 1 ;motor delay between steps
DB 145, 165, 0, 0, 0, 0, 0, 0, 0, 133
DB 120, 133, 145, 125, 0, 0, 0
DB 133, 115, 0, 0, 0, 133 ;S5 A4 TILT js
DB FFH ;end

Tbl3_M347:
DB 10 ;motor delay between steps
DB 115, 133, 120, 160
DB 0, 0, 0, 0, 190, 0, 0, 0, 0
DB 0, 0, 0, 0, 0, 0, 0, 155, 133 ;S6 A4 TILT (with say/m5) js
DB FFH ;end

Tbl3_M348:
DB 1 ;motor delay between steps
DB 120,133,115,133,155
DB 0,0,0,0,0,0,133 ;Sll A4 TILT (with say/m2) js
DB FFH ;end

;Tbl3_M349:
DB 1 ;motor delay between steps
DB 145,155,115,133 ;Sl3 A4 TILT (with say/m5) js
DB FFH ;end

;Tbl3_M350:
DB 5 ;motor delay between steps
DB 145,158,0,0,0,0,138,147,155
DB 0,0,0,0,0,0,133
DB 125,120,115,133 ;Sl6 A4 TILT (with say/m5) js
DB FFH ;end

;END TILT
;END GEORG
;GEORGE
;IR 07/09/98

Tbl3_M351:
DB 20 ;motor delay between steps SGTEST
DB 120,100,133 ;seq1,seq2,seq3,seq4 IR age 1
DB FFh

Tbl3_M352:
DB 46 ;motor delay between steps SGTEST
DB 115,100,75,133 ;seq5 ir age 1
DB FFh

; DANGER

Tbl3_M353:
DB 30 ;motor delay between steps
DB 115,130,100,70 ;seq6 (DANCE,WAH) ir AGel
DB FFh

;Tbl3_M354:
DB 1 ;motor delay between steps
DB 133,145,155,190,133,155,175,145,133 ;seq6 (DO DO DO) ir AGel

AGel
DB FFh

Tbl3_M355:
DB 8 ;motor delay between steps
DB 145,115,145,133,145,115,145,133,0,0,0,0,0
DB 125,110,133,0,160,0,0,0,133
DB FFh ;end

Tbl3_M356:
DB 1 ;motor delay between steps
DB 0
DB FFh ;empty space

Tbl3_M357:
DB 1 ;motor delay between steps
DB 120,115,110,105,100,80,100,120,115,100,45,133 ;seq8 ir age 1
DB FFh

Tbl3_M358:
DB 10 ;motor delay between steps
DB 120,115,100,80,133,145,160,133 ;seq9 ir age 1
DB FFh

Tbl3_M359:
DB 1 ;motor delay between steps
DB 115,133,140,145,133,160,180
agel DB FFh
Tbl3_M360:
DB 1 ;motor delay between steps
DB 120,107,122,113,100,75,90,80,88,100,0,0,133
DB 120,107,122,113,100,75,90,80,88,100,133
DB 146,140,155,133 ;seq12 ir
agel DB FFh
Tbl3_M361:
DB 5 ;motor delay between steps
DB 115,125,100,10,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
DB 110,118,100,0,133 ;seq13,14 ir
agel DB FFh
Tbl3_M362:
DB 10 ;motor delay between steps
DB 160,0,0,190,160,0,0,133,100,0,0,0,133 ;seq15 ir
agel DB FFh
; DANGER SLEEP
Tbl3_M363:
DB 90 ;10 ;motor delay between steps
DB 85,40,30,85,40,30,0,85,40,30,0,85,40,30,10 ;seq16 ir
agel DB FFh
Tbl3_M364:
DB 1 ;motor delay between steps
DB 125,113,125,118,105,133 ;seq1,2,3 ir age2
agel DB FFh
Tbl3_M365:
DB 10 ;motor delay between steps
DB 125,113,125,118,105,133 ;SEQ4,5 IR AGE2
DB FFh
Tbl3_M366:
DB 10 ;motor delay between steps
DB 145,155,140,145,142,150,0,0,0,0,0,133 ;seq6
ir age2 DB FFh
Tbl3_M367:
DB 5 ;motor delay between steps
DB 10,40,10,40,133,143,140,145,143,145
DB 133
DB 100,133
DB 125,113,133 ;seq7,8 ir age 2
DB 125,113,133 ;seq7,8 ir age 2
DB FFh
Tbl3_M368:
DB 10 ;motor delay between steps
<table>
<thead>
<tr>
<th>Table ID</th>
<th>Description</th>
<th>Sequence</th>
<th>Delay Between Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tbl3_M369</td>
<td>1 :motor delay between steps</td>
<td>seq10</td>
<td>age2 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tbl3_M370</td>
<td>1 :motor delay between steps</td>
<td>seq11</td>
<td>age2 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tbl3_M371</td>
<td>1 :motor delay between steps</td>
<td>seq13.14</td>
<td>age2 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tbl3_M372</td>
<td>43 :motor delay between steps</td>
<td>seq15</td>
<td>age2 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tbl3_M373</td>
<td>90 :motor delay between steps</td>
<td>seq16</td>
<td>age2 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tbl3_M374</td>
<td>1 :motor delay between steps</td>
<td>seq1.2.3.4.5</td>
<td>age3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tbl3_M375</td>
<td>1 :motor delay between steps</td>
<td>seq6</td>
<td>age3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tbl3_M376</td>
<td>1 :motor delay between steps</td>
<td>seq7.8</td>
<td>age3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tbl3_M377</td>
<td>1 :motor delay between steps</td>
<td>seq9</td>
<td>age3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tbl3_M378</td>
<td>1 :motor delay between steps</td>
<td>seq9</td>
<td>age3</td>
</tr>
</tbody>
</table>
DB 120,122,115,125,112,150,0,0,0,133 ;seq11 ir age3
DB FFH ;end

;Tbl3_M379:
DB 1 ;motor delay between steps
DB 115,10,0,10,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
DB 0,0,0,0,0,0,0,0,0,0
DB 145,110,0,0,0,0,0,0,0,133 ;seq12,14 ir age3
DB FFH ;end

;Tbl3_M380:
DB 12 ;motor delay between steps
DB 117,0,0,0,0,0,0,0,0,0,0,0,0,100,0,0,0,0,0,0,0
DB 100,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
DB FFH ;end

;Tbl3_M381:
DB 5 ;motor delay between steps
DB 120,150,110,0,0,0,133 ;seq1,2,3,4,5 ir age4
DB FFH ;end

;Tbl3_M382:
DB 10 ;motor delay between steps
DB 120,110,145,155,100,133 ;seq6 ir age4
DB FFH ;end

;Tbl3_M383:
DB 8 ;motor delay between steps
DB 145,115,145,133,145,115,145,133,0,0,0,0,0
DB 125,110,133,0,160,0,0,0,133
DB FFH ;end

;Tbl4_M384:
DB 1 ;motor delay between steps
DB 115,133,143,148,136,160,180
DB 173,167,160,180,173,167,160,140,145,133 ;seq9 ir age4
DB FFH ;end

;Tbl4_M385:
DB 1 ;motor delay between steps
DB 118,0,0,155,0,0,133,0,0,118,0,133,0,0,0,0,0,0,110
DB 0,0,0,133,120,107,122,113,100,75,90,80,88,100,133
DB FFH ;end SAY NUMBERS MODIFIED TO MATCH CORRECT

DIALOGUE

;Tbl4_M386:
DB 1 ;motor delay between steps
DB 120,123,112,133,143,151,160,133
DB FFH ;end

;Tbl4_M387:
DB 1 ;motor delay between steps
DB 120,0,0,145,110,145,110,0,0,0,0,0,133
DB FFH ;end

;Tbl4_M388:
DB 1 ;motor delay between steps
DB 120,110,133 ;OK ;seq14 ir age4
Tbl4_M389:
DB 90 ;motor delay between steps
DB 150,0,130,0,100,0,133 ;YAAN
DB FFH ;end

Tbl4_M390:
DB 90 ;motor delay between steps
DB 0,0,85,30,0,20,0,85,30,0,20,0,85,0,20,0,85,10
DB FFH ;end

END GEORGE 07/09/98
END IR

; FURBY SAYS: (LIGHT) DMH
Tbl4_M391:
DB 10 ;motor delay between steps
DB 110,133 ;LIGHT (furby says)
DB 110,120,133 ;LIGHT (furby says)
DB FFH ;end

Tbl4_M392:
DB 1 ;dmh no light
DB 150,0,0,0,115,0,0,0,0,133
DB FFH ;end

Tbl4_M393:
DB 30 ;dmh loud sound
DB 150,0,0,0,115,0,0,0,0,133
DB FFH ;end

Tbl4_M394:
DB 10 ;motor delay between steps
DB 140,150,0,0,133
DB FFH

Tbl4_M395:
DB 10 ;motor delay between steps
DB 160,133 ;[ME]
DB FFH ;end

Tbl4_M396:
DB 1 ;motor delay between steps
DB 120,130,120,133 ;ME ME
DB FFH ;end

Tbl4_M397:
DB 1 ;motor delay between steps
DB 115,130,110,133 ;DO MCH
DB FFH ;end

Tbl4_M398:
DB 1 ;motor delay between steps
DB 120,130,110,133 ;TOH LOC
DB FFH ;end

A-290
Tbl4_M399:
DB 1 ;motor delay between steps
DB FFH ;end

Tbl4_M400:
DB 1 ;motor delay between steps
DB FFH ;end ; start diagnostic

Tbl4_M401:
DB 1 ;motor delay between steps
DB FFH ;end ; key press beep

Tbl4_M402:
DB 1 ;motor delay between steps
DB FFH ;end ; pass beep

Tbl4_M403:
DB 1 ;motor delay between steps
DB FFH ;end ; fail beep

Tbl4_M404:
DB 1 ;motor delay between steps
DB FFH ;end

Tbl4_M405:
DB 1 ;motor delay between steps
DB 10,200,10,134 ; motor cal
DB FFH ;end

Tbl4_M406:
DB 1 ;motor delay between steps
DB 120 ; feed 1
DB FFH ;end

Tbl4_M407:
DB 255 ;motor delay between steps
DB 0,134 ; feed 2
DB FFH ;end

Tbl4_M408:
DB 1 ;motor delay between steps
DB 30 ; light pass
DB FFH ;end

Tbl4_M409:
DB 1 ;motor delay between steps
DB 160 ; sound pass
DB FFH ;end

Tbl4_M410:
DB 1 ;motor delay between steps
DB 10 ; sleep
DB FFH ;end

Tbl4_M411:
DB 20 ; PEEK-BOO (HIDE AND SEEK) DEM
DB 155,133,0,0,147,133
DB FFH
Tbl4_M412: ; feed dmh
  DB 1 ;motor delay between steps
  DB 165,0,0,0,0,0,0,0,0,0,150,0,0,0,0,150 ;(AAAH)
  DB 0,0,165;0,0,0,0,0,133 ;(AAAH)
  DB FFH ;end

; DB FFH ;end

Tbl4_M413:
  DB 1 ;motor delay between steps
  DB FFH ;end

Tbl4_M414:
  DB 1 ;motor delay between steps
  DB FFH ;end

Tbl4_M415:
  DB 1 ;motor delay between steps
  DB FFH ;end

Tbl4_M416:
  DB 1 ;motor delay between steps
  DB FFH ;end

Tbl4_M417:
  DB 1 ;motor delay between steps
  DB FFH ;end

Tbl4_M418:
  DB 1 ;motor delay between steps
  DB FFH ;end

Tbl4_M419:
  DB 1 ;motor delay between steps
  DB FFH ;end

Tbl4_M420:
  DB 1 ;motor delay between steps
  DB FFH ;end

Tbl4_M421:
  DB 1 ;motor delay between steps
  DB FFH ;end

Tbl4_M422:
  DB 1 ;motor delay between steps
  DB FFH ;end

Tbl4_M423:
  DB 1 ;motor delay between steps
  DB FFH ;end

Tbl4_M424:
  DB 1 ;motor delay between steps
  DB FFH ;end

Tbl4_M425:
  DB 1 ;motor delay between steps
Bl14_M426:
DB 1 ;motor delay between steps
DB FFH ;end

Bl14_M427:
DB 1 ;motor delay between steps
DB FFH ;end

Bl14_M428:
DB 1 ;motor delay between steps
DB FFH ;end

Bl14_M429:
DB 1 ;motor delay between steps
DB FFH ;end

Bl14_M430:
DB 1 ;motor delay between steps
DB FFH ;end

Bl14_M431:
DB FFH ;end

Bl14_M432:

Bl14_M433:

Bl14_M434:
DB 1 ;motor delay between steps
DB 0
DB FFH ;end

Bl14_M435:
DB 1 ;motor delay between steps
DB 0
DB FFH ;end

Bl14_M436:
DB 1 ;motor delay between steps
DB 0
DB FFH ;end

Bl14_M437:
DB 1 ;motor delay between steps
DB 0
DB FFH ;end

Bl14_M438:
DB 1 ;motor delay between steps
DB 0
DB FFH ;end

Bl14_M439:
DB 1 ;motor delay between steps
DB 0
DB FFH ;end

Bl14_M440:
DB 1 ;motor delay between steps
DB 0
DB FFH ; end

Tbl4_M441;
Tbl4_M442;
Tbl4_M443;
Tbl4_M444;
Tbl4_M445;
Tbl4_M446;
Tbl4_M447;
Tbl4_M448;
Tbl4_M449;
Tbl4_M450;
Tbl4_M451;
Tbl4_M452;
Tbl4_M453;
Tbl4_M454;
Tbl4_M455;
Tbl4_M456;
Tbl4_M457;
Tbl4_M458;
Tbl4_M459;
Tbl4_M460;
Tbl4_M461;
Tbl4_M462;
Tbl4_M463;
Tbl4_M464;
Tbl4_M465;
Tbl4_M466;
Tbl4_M467;
Tbl4_M468;
Tbl4_M469;
DB 10, 200, 134
DB FFH

motor delay between steps

DB 10, 200, 100
DB FFH

motor delay between steps

.*.