Lego Programming for the Apple //
Evan Koblentz @ Kansasfest 2017
Childhood

• Lego fan ever since I can remember

• Learned LOGO (only turtle graphics) on C-64 in 5th grade

• Learned Applesoft BASIC on ][+ in 6th grade

• Got a //e Enhanced for my bar mitzvah in 7th grade

• Special memory: told parents my wish that year, no luck :(

• Got out of computers after middle school
Adulthood

• Got into vintage computing around 2002
• Linux for modern computing, Platinum for vintage
• Co-founded MARCH in 2004
• Co-founded Vintage Computer Federation Inc. in 2015
• VCF is a 501(c)(3); VCForum; VCF East/West/more; etc.
• (Still) shun social interaction to play with Lego :)}
Enough background! Let’s discuss Lego Programming for the Apple //

• In 2016 VCF/Mid-Atlantic chapter member Ben G. loaned us NIB kit for HOPE & World Maker Faire

• It was the best medicine!!!

• Rabbit hole of 1980s technology led me to here :)

• Disclaimer: I don’t know much tech, but maybe you’ll dig it
All about the kit

- Lego 9700 Technic Control Center - 1986
- Card, interface box, motors + sensors + lights
- Choose your weapon: Apple II (with Applesoft firmware card), Apple ][+, Apple //e, Apple IIgs, etc.
But not Apple //c! Thus: Laser 128
Commercial interruption :) 

My friend Jon Chapman made the replica card using a prototype of his new Apple II board. Henry has them here at Kfest or visit www.Glitchwrks.com
LEGO Robotics TC Logo

Lego 9767

Transfers the programming data and the sensor data to the processor bus when accessing the memory address for the 9767 card.

1. Provides an integrated peripheral expansion
2. Offers 2x 8-bit I/O ports
3. Contains 2x 16-bit Timers
4. Interrupt control

Programmable ABORT signal same as STOP button

74LS245
8-bit Data Bus Transceiver

8-bit Data

6522 VIA
Versatile Interface Adapter

Control Logic
Port A Data Register

Port A bit.0

Port B bits 0-7

8-bit Data

6 Outputs
2 Inputs

74LS74
Clock Divider

Interrupt signal

Timing functions available via the expanded Lego TC Logo commands and other languages, eg. Apple Pascal

Provides the timing clock to allow the 9767 card to latch the data from your code and from the sensors

* More detailed information can be found in the datasheet
LEGO Robotics TC Logo

**Motor Output**

- **Data Port**
  - Data Bit# 0
  - Data Bit# 1
- **Opto Isolator**
- **Driver Interface**
- **Motor H-Bridge**
  - Port 0
  - Port 1
- **Bi-Directional output**
- **3 modes of operation**
  - #1 CW
  - #2 CCW
  - #3 FWD/RVS
- **MOTION**
- **MOTION**
- **MOTION**

*opposite direction is done by flipping the polarity on the motor cable*

---

**Sensor Input**

- **LIMIT SWITCH**
- **OPTO SENSOR**

---

**Operational Amplifiers**

- **Opto Isolator**
- **Data Port**
- **Data Bit# 0**
- **Data Bit# 1**

*Converts the Analog signal from the sensors into digital data*
LEGO Robotics TC Logo

Lego 9767
Slot 1 to 7

Lego 9750
6 Output Ports
2 Input Ports

Bi-Directional Option
On/Off Option
Test Port
STOP Button

OPTO SENSOR
LIMIT SWITCH
CCW MOTION*
CW MOTION
FWD/RVS MOTION
FWD/RVS MOTION
All about the kit

• Lego Technic Control Center - 1986

• Card, interface box, motors + sensors + lights

• Choose your weapon: Apple II (with Applesoft firmware card), Apple ][+, Apple //e, Apple IIgs, etc.

• Apple II + LOGO or IBM + BASIC
All about the kit

• Lego Technic Control Center – 1986

• Card, interface box, motors + sensors + lights

• Choose your weapon: Apple II (with Applesoft firmware card), Apple ][+, Apple //e, Apple IIgs, etc.

• Apple II + LOGO or IBM + BASIC

• Experiment 1: IBM (sort of) + BASIC = Fail
17
9950 ' ---
9960 ' INIT
9970 ' ---
9980 ' ---
9990 ' 10000 P=925
10010 OUT P,21
10020 IF (INP(P) AND 63)=21 THEN OUT P,0 ELSE ERC=4 : GOTO 20000
10030 RETURN
10040 '
10050 '----------------------------------
10060 ' BITON ENTRY PAR: NUM%
10070 '----------------------------------
10080 ' 10090 ' 10090 ' 10100 IF NUM%>=0 AND NUM%<6 THEN 11020
11010 ERC=1: GOTO 20000
11020 OUT P,(INP(P) OR 2^NUM%)
11030 RETURN
11040 '
11050 '----------------------------------
11060 ' BITOFF ENTRY PAR: NUM%
11070 '----------------------------------
11080 ' 11090 ' 11090 ' 11100 IF NUM%>6 OR NUM%=-7 THEN 13020
11110 ERC=2: GOTO 20000
11120 Y%=((INP(P) AND 2^NUM%)/ 2^NUM%
11130 RETURN
11140 '
11150 '----------------------------------
11160 ' WAIT ENTRY PAR: TIM%
11170 '----------------------------------
11180 ' 11190 ' 11190 ' 11200 IF TIM%>=0 THEN 14020
11210 ERC3: GOTO 20000
11220 QT=TIMER + TIM%
11230 IF QT>TIMER THEN 14030
11240 RETURN
11250 '
11260 '----------------------------------
11270 ' ERROR HANDLING
11280 '----------------------------------
11290 ' 11300 ' 11300 ' 11310 CLS:COLOR 20,0 PRINT"PARAMETER ERROR":COLOR 7,0
11320 ' 11330 ' 11330 ' 11340 IF ERC=1 THEN PRINT "OUTPUT BITS MUST BE BETWEEN 0 AND 5":END
11350 ' 11360 ' 11360 ' 11370 IF ERC=2 THEN PRINT "INPUT BITS MUST BE 6 OR 7":END
11380 ' 11390 ' 11390 ' 11400 IF ERC=3 THEN PRINT "WAIT TIME MUST BE POSITIVE":END
11410 ' 11420 ' 11420 ' 11430 IF ERC=4 THEN PRINT "NO INTERFACE CARD IN COMPUTER AT ADDRESS 925":END
11440 ' 11450 ' 11450 ' 11460 END
All about the kit

- Lego Technic Control Center – 1986
- Card, interface box, motors + sensors + lights
- Choose your weapon: Apple II (with Applesoft firmware card), Apple ][+, Apple //e, Apple IIgs, etc.
- Apple II + LOGO or IBM + BASIC
- Experiment 1: IBM (sort of) + BASIC = Fail
- Experiment 2: Apple II + (sort of) LOGO = Pass/Fail
LEGO® TC logo
LEGO Systems Inc.

©Logo Computer Systems Inc. 1987
Press Return
TO TANK
IF BUTTON? 0 [LTREAD]
IF BUTTON? 1 [RTREAD]
ELSE IF NOT BUTTON? 0 [CTTO "A OFF"]
ELSE IF NOT BUTTON? 1 [CTTO "B OFF"]
TANK}
END

TO LTREAD
IF (PADDLE 1)<75 [CTTO "A SETEVEN ON"]
ELSE IF (PADDLE 1)>180 [CTTO "A SETODD ON"]
END

TO RTREAD
IF (PADDLE 1)<75 [CTTO "B SETEVEN ON"]
ELSE IF (PADDLE 1)>180 [CTTO "B SETODD ON"]
END
IF (PADDLE 1) > 170 [TTO 5 OFF]
IF (PADDLE 1) < 170 [TTO 5 SETPOWER 3] [ON]
IF (PADDLE 1) < 85 [TTO 5 SETPOWER 7]
V8
V8
All about the kit

• Lego Technic Control Center - 1987

• Card, interface box, motors + sensors + lights

• Choose your weapon: Apple II (with Applesoft firmware card), Apple ][+, Apple //e, Apple IIgs, etc.

• Apple II + LOGO or IBM + BASIC

• Experiment 1: IBM (sort of) + BASIC = Fail

• Experiment 2: Apple II + (sort of) LOGO = Pass/Fail

• Experiment 3: Lego Lines (1987) = Apple II + unsupported BASIC = Inconclusive (capable but over-engineered)
These routines assume that the LEGO Slot Card has been installed in slot 2.

**Initialising**

This routine must be called first to set up the LEGO Interface correctly:

```plaintext
1000 REM INITIALISE INTERFACE
1001 S=5: L=49280+S+16
1002 POKE L+3, 1
1003 POKE L+2, 63
1004 POKE L+1, 0
1005 POKE L, 0
1006 RETURN
```

**Reading data**

Below is a subroutine which will read data from the interface, and store this in an array DB(.). This starts off as a single decimal value, but will here be converted into elements DB(6) and DB(7), containing 1’s or 0’s to represent on or off.

```plaintext
1200 REM INPUT DATA
1201 DB=PEEK(L) :REM Read data
1202 DB(7)=(DB>127) :REM Convert binary data
1203 DB=DB-128+DB(7)
1204 DB(6)=(DB>63)
1205 RETURN
```

This next routine uses the above to test whether certain bits are on or off, as required. It also allows you to set either test bit as any value. To use it, you must first set the test bits T(7) and T(6) (Temporary 7 and Temporary 6) to 0, 1 or -1 (for any value). The results will return as T=1 for true, and T=0 for false.

```plaintext
1210 REM TEST INPUT BITS
1211 GOSUB 1200 :REM Read Input Bits
1212 T=(T(7)=DB(7) OR T(7)=-1) AND (T(6)=DB(6) OR T(6)=-1)
1213 RETURN
```

For example, suppose you wish to test whether bit 7 was on, while bit 6 could be any value. Then you could write:

```plaintext
T7=1: T6=1: GOSUB 1210: IF T THEN (etc)
```

**Sending data**

Below is a subroutine which will send the data stored in an array DB(5) to the interface. It is assumed that the elements DB(0) through DB(5) contain 1’s and 0’s to turn the bits on or off. These data elements are then combined into a single decimal value to be sent.

```plaintext
1110 REM OUTPUT DATA
1111 DB=5
1112 FOR I=0 TO 5
1113 DB=DB+DB(I)*2^I
1114 NEXT I
1115 POKE L, DB :REM Send data to interface
1116 RETURN
```

The following routine will turn on all the bits specified. If you call it at the beginning (line 1110), you will turn off all the other bits. If you call it at line 1115, you will leave the other bits alone.

To use this routine, you must first set the required bits (T(0) to T(5)) to 1.

```plaintext
1117 REM TURN ON SPECIFIC BITS
1118 FOR I=0 TO 5
1119 DB(I)=0
1120 NEXT I
1121 GOSUB 1110 :REM Turn off all bits first
1122 FOR I=0 TO 5
1123 DB(I)=DB(I) OR T(I)
1124 NEXT I
1125 RETURN
```

For example, to turn on bits 3 and 4, without changing the other bits, you could use the following line:

```plaintext
T(3)=1: T(4)=1: GOSUB 1115
```

To turn on bits 3 and 4, and the rest off, use:

```plaintext
T(3)=1: T(4)=1: GOSUB 1110
```

Finally, the following routine will turn the desired bits on and off for a set period of time, testing for the ESC key while waiting. Note that, like LEGO Lines, it will not turn them off at the end of the routine. The data for this routine must be set as in the above routine at line 1115. In addition, the Output Time (OT) must be set (to 1 decimal place).

```plaintext
1120 REM TIMED OUTPUT
1121 ES=0
1122 FOR J=1 TO OT*25
1123 GOSUB 1115
1124 IF PEEK(49152)=155 THEN ES=1: GOTO 1129
1125 NEXT J
1126 RETURN
```
Discovered “Lego Lines” and this:

“...designed to allow the programmer to experiment further with the Lego interface”
The Slot Card

The Apple LEGO Slot Card is based on a Mostek 6522 VIA chip. The chip is communicated with through the I/O addresses, calculated as follows:

\[ L = 49280 + S \times 16 \]

where \( S \) is the slot number, and \( L \) the resulting address.

All input/output will come through the address \( L \), although the next three addresses are used during setup. A typical setup sequence runs as follows:

POKE L+3, 1
POKE L+2, 63
POKE L+1, 0
POKE L, 0

This sets up the 6522 registers so that bits 0-5 are output bits, and bits 6 and 7 are input bits. All I/O is then done through address \( L \).
PEEK and POKE are our friends.
• Lesson taught to me by Dan Roganti: Each “port” is a bit in the byte, and so…

• If it weighs the same as a duck… :)

• All we have to do is POKE the address of the device with the decimal total of the “ports” that we want to enable!
All about the kit

• Lego 9700 Technic Control Center - 1986

• Card, interface box, motors + sensors + lights

• Choose your weapon: Apple II (with Applesoft firmware card), Apple ][+, Apple //e, Apple IIgs, etc.

• Apple II + LOGO or IBM + BASIC

• Experiment 1: IBM (sort of) + BASIC = Fail

• Experiment 2: Apple II + (sort of) LOGO = Pass/Fail

• Experiment 3: Apple II + unsupported BASIC = Inconclusive

• Experiment 4: Apple II + hacked BASIC = Success!
10    REM   THIS IS THE MAIN PROGRAM
20    GOSUB 1000: REM   INITIALIZE LEGO INTERFACE
30    GOSUB 2000: REM   DISPLAY INSTRUCTIONS
40    GOSUB 3000: REM   NAVIGATION CONTROL
50    GOSUB 4000: REM   FORKLIFT CONTROL
60    GOTO 40: REM   LOOP OPERATION
70    END : REM   JUST FOR GOOD GRAMMAR :) 

1000   REM INIT LEGO CARD+INTERFACE
1010   REM   SET SLOT AND MEMORY LOCATION
1020   S = 7:L = 49280 + S * 16
1030   REM   PREPARE INTERFACE CHIPS
1040   POKE L + 3,1: POKE L + 2,63: POKE L + 1,0
1050   POKE L,0: REM   CLEAR ALL PORTS
1060   RETURN : REM   GO!

2000   REM   DISPLAY INSTRUCTIONS
2010   HOME
2020   PRINT "Say hello to Leinad Legobot!"
2030   VTAB 3: PRINT "Use the joystick to make him go forward, backward, left, and right."
2040   VTAB 6: PRINT "Press top button to raise his forklift."
2050   VTAB 8: PRINT "Press left button to lower his forklift."
2060   VTAB 10: PRINT "Want to know how he works? Ask us!"
2070   VTAB 12: PRINT "******************************************************************"
2080   VTAB 14: PRINT "Lego design and construction by Evan."
2090   VTAB 16: PRINT "Software by Evan, Dan, and Paul."
2100   VTAB 18: PRINT "Interface card by Jonathan."
2110   VTAB 20: PRINT "Inspiration by Ben, kit donated by Eric."
2120   VTAB 22: PRINT "Learn more! www.vcfed.org"
2130   RETURN
3000 REM NAVIGATION
3010 FB = PDL (1):LR = PDL (0):M = 0: REM SET VARIABLES
3020 IF FB < 75 THEN M = 5
3030 IF FB > 180 THEN M = 10
3040 IF LR < 75 THEN M = 9
3050 IF LR > 180 THEN M = 6
3060 POKE L,M: REM SEND COMMANDS
3070 IF M = 10 THEN CALL - 198: FOR W = 1 TO 500: NEXT W: REM BACKUP ALERT
3080 RETURN

4000 REM FORKLIFT
4010 REM CHECK LIMIT SWITCHES
4020 IF PEEK (L) = 64 THEN GOTO 4130: REM CHECK LOWER SWITCH, GO UP
4030 IF PEEK (L) = 128 THEN GOTO 4110: REM CHECK UPPER SWITCH, GO DOWN
4040 GOTO 4200: REM NO ACTIVE SWITCHES, DO ANYTHING

4100 REM IF ACTIVE SWITCH, DO OPPOSITE
4110 IF PEEK (49249) > 127 THEN POKE L,16: GOTO 4400: REM KEEP LOWERING
4120 RETURN : REM NO BUTTON, EXIT
4130 IF PEEK (49250) > 127 THEN POKE L,32: GOTO 4500: REM KEEP RAISING
4140 RETURN : REM NO BUTTON, EXIT

4200 REM NO ACTIVE SWITCH, CHECK BUTTON 0
4210 IF PEEK (49249) > 127 THEN POKE L,16: GOTO 4400
4220 REM NO BUTTON, FALL THROUGH

4300 REM NO ACTIVE SWITCH, CHECK BUTTON 1
4310 IF PEEK (49250) > 127 THEN POKE L,32: GOTO 4500: REM KEEP RAISING
4320 RETURN : REM NO BUTTON, EXIT

4400 REM LOOP ON DOWN MOTOR WHILE BUTTON 0 AND INACTIVE LOWER SWITCH
4410 IF PEEK (L) = 16 OR PEEK (L) = 144 THEN GOTO 4210: REM KEEP LOWERING
4420 IF PEEK (L) = 80 THEN POKE L,0: RETURN : REM CHECK LOWER SWITCH, STOP, EXIT
4430 POKE L,0: RETURN : REM NOTHING HAPPENING, EXIT

4500 REM LOOP ON UP MOTOR WHILE BUTTON 1 AND INACTIVE UPPER SWITCH
4510 IF PEEK (L) = 32 OR PEEK (L) = 96 THEN GOTO 4310: REM KEEP RAISING
4520 IF PEEK (L) = 160 THEN POKE L,0: RETURN : REM CHECK UPPER SWITCH, STOP, EXIT
4530 POKE L,0: RETURN : REM NOTHING HAPPENING, EXIT
Pause to wave to Steve Jobs, wearing his Lego turtleneck and blue jeans.
All about the kit

• Stronger than DOS, Windows, and OS/2 combined!

• But can it lift the **WORLD’S LARGEST*** Apple //e…?

• * by scale :)
More about the kit

• There is other experimental documentation for 6502 assembly

• Didn’t try it: “You know, for kids!” - The Hudsucker Proxy

• Other reason: time to confess, please don’t judge me :)

• It’s time to show but not much “tell”...
Programming in machine language

Although the LEGO Interface can be programmed in BASIC, some of the LEGO *Lines* program is written in 6502 machine code, to improve performance.

This section is intended only for experienced machine language programmers who wish to write their own routines for controlling the LEGO Interface. It only discusses how to initialise the interface, and how to get data to and from it.

All hexadecimal values are denoted by the prefix $ (for example, $2A), while binary values are denoted with the prefix % (for example, %101100).

The address of the interface I/O port is given by the following:

```
LEGO    EQU    $C080+S10+S
```

where $S$ is the slot number of the LEGO Interface. Normally, where the LEGO Interface is in slot 2, the address is given by:

```
LEGO    EQU    $C0A0
```
Initialisation

The correct initialisation sequence for the interface is as follows:

LEGO INIT
EQU $C0A0
LDA #1
STA $C0A3
LDA #$3F
STA $C0A2
LDA #$0
STA $C0A1
STA $C0A0
RTS
**Reading data**

Reading the data is more complex. First, the data must be read from the interface. Then the correct bits must be filtered out, if necessary. Finally, the appropriate bits must be examined.

To read the current status of the output bits (to read which bits are currently on):

<table>
<thead>
<tr>
<th>STATUS</th>
<th>LDA</th>
<th>$\text{C0A0}$ AND</th>
<th>#$3F$</th>
<th>:Filter out bits 7 &amp; 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEST</td>
<td>JSR</td>
<td>STATUS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STA</td>
<td>STA</td>
<td>TEMP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDA</td>
<td>LDA</td>
<td>#0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STA</td>
<td>STA</td>
<td>COUNT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T0</td>
<td>ROR</td>
<td>:Rotate last bit into Carry</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCC</td>
<td>T1</td>
<td>:No bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JSR</td>
<td>ACT</td>
<td>:Else act on it</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1</td>
<td>INC</td>
<td>COUNT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDA</td>
<td>LDA</td>
<td>COUNT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CMP</td>
<td>#6</td>
<td>:Up to 6?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BNE</td>
<td>T0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Writing data

All data is written to address $C0A0. The six data bits on the interface correspond exactly to the six data least significant bits written to the address. So, for example, to turn on bits 1 and 3 only (and the rest off), it is necessary to load the accumulator with binary 001010 ($0A).

Below is the routine to send the bits in the accumulator to the interface:

SEND      AND      #$3F     :Mask with 00111111
STA $C0A0  :to filter out bits 7 & 6
RTS
More software thoughts

• Still other experimental support for LOGO II
• 1988: LogoWriter Robotics (LCSI + Lego TC in one)
• Any language really: VB (been done!), Java, etc.
• Logo uses clock on the card; custom in others?
• How to correlate robot movement with on-screen sprite
• Apple II control of modern Mindstorms???
More hardware thoughts

• Supposedly there were C-64 and BBC Micro versions

• In theory Commodore 64 wouldn’t need a card

• Interface with set #8094 Plotter (part of 1989 Control Center which uses push-button programming, unclear about software)

• Multiple interface boxes = do more stuff!

• Modern computers only need the parallel port
Next-to-last slide!

- Where to get your own kit? Ebay, Bricklink.com, more kit data at Brickopedia, Brickowl, Technicopedia, individual blogs (Google is your friend)… or DIY using online schematics (limited)!

- Online software (limited) / Online docs (very limited!)


- Next step: Leinad* game development ( * for my friend Dan)

- Child/parent Lego learning station @ VCF Museum

- Other thanks: Paul Hagstrom, Michael Mulhern (& many others via Apple II Enthusiasts Facebook group, VCForum, Applefritter)

- Just one more slide to go...
The Last Slide

- 2016 World (NYC) Maker Faire: Make Magazine Editor’s Choice blue ribbon (for LOGO-programmed simple robotic car along with Jeff Brace’s BASIC-powered Capsella/C-64 robot)


- One day I will learn 6502 assembly

- Come talk to me this week or email me: evan@vcfed.org

- Ideas for programming (and Lego building!) welcome

- The end / Q&A