Hayesduino

Payton Byrd

Contents

1. What Is It?
2. History
3. Implemented Hayes Command Set
    1. Standard Commands
    2. Extended Commands
    3. Factory Reset
    4. Non-Standard Commands
    5. Phonebook Usage
    6. Set MAC Address
    7. Manually Set IP Address
    8. Manually Set Gateway Address
    9. Manually Set DNS Address
4. Hardware Requirements
5. Platform-Specific Docs
    1. Commodore Userport
    2. RS232
6. License
7. Compiling Notes

1. What Is It?

Hayesduino is an Arduino sketch that provides a bridge between the world of the Internet and small devices that do not have built-in ethernet capabilities. Old computers, such as the Commodore 64, Apple II and Atari 800 have serial ports, but do not have readily available Internet solutions with wide software support. While specialized solutions do exist for these platforms, they all require specialized software to use them and do not lend themselves to more general usage such as simply opening a socket, sending some data, and/or receiving some data.

Hayesduino bridges this gap by emulating a Hayes compatible modem. This allows users to initiate Internet communications via sockets that are opened by "dialing" to a hostname and port. An example would be initiating a telnet session with a host by simply typing *atdt hostname:23* and waiting for the host to respond. Using this technique, any online socket can be reached and communicated with.

Hayesduino could have accomplished this without emulating a modem, but there needed to be a good way to allow the small machine to receive incoming connections. The three platforms listed above were all very popular systems for hosting BBS (bulletin board systems) which would accept calls over a telephone line via modem. Hayesduino simulates the incoming phone call whenever the software receives an inbound connection on port 23 (this is changeable in the code). When an incoming connection is detected, the Hayesduino will toggle the DCE-DCD line to trigger the remote software to answer the incoming "call". In this way a classic BBS can be hooked up directly to the Internet.

2. History

Hayesduino started in early 2013 as the firmware for the Comet BBS product which is to be sold as a commercial product. The owner of the Comet BBS project and Payton Byrd agreed that the firmware would be made open-source as the Comet BBS product is based on the Arduino Uno and thus easily reproducible as a hobby project. Although the Comet BBS hardware project appears to be stalled, the development of the firmware is not stalled and I, Payton Byrd, have decided to move forward with open-sourcing the firmware as the Hayesduino project.

3. **Implemented Hayes Command Set**

3.1 Standard Commands

All commands are prefixed with the letters "AT". Commands can be chained (but not queries).

Command	**Description**
/	Replay last command
A	Answer Call
D	Dial *
DT	Dial **
DP	Dial **
E0	Echo Off ***
E1	Echo On
H	Hang Up Call
H0	Hang Up Call
M0	Unused
Q0	Quiet Mode Off ***
Q1	Quiet Mode On ***
V0	Verbose Mode Off - Returns numeric result codes ***
V1	Verbose Mode On - Returns textual result codes ***
X0	Unused
X1	Unused
X2	Unused
&W	Write settings as default to EEPROM
Z	Reset to default settings

** Use without a space to dial a phonebook entry. ex: ATD1 would dial phonebook entry 1*
*** Use with a space to dial a hostname or IP address. ex: ATDT BBS.PAYTONBYRD.COM would dial the hostname on port 23*
**** Saves when AT&W is called*

3.2 Extended Commands

Hayesduino knows of the following S-Register extended commands:

** All S-Registers are preserved when performing AT&W.*

Command	Description	Notes
S0	Auto answer, 0 mean answer immediately, > 0 means to answer after the number of rings in S1.	
S1	Rings before auto answer	Value is saved, but currently the feature is not implemented.
S2	Escape Character	Defaults to '+'
S3	Carriage Return Character	Value is saved, but currently the feature is not implemented.
S4	Line Feed Character	Value is saved, but currently the feature is not implemented.
S5	Backspace Character	
S6	Wait Blind Dial	Not Applicable
S7	Wait for Carrier	Not Applicable
S8	Pause for Comma	Not Applicable
S9	CD Response Time	Value is saved, but currently the feature is not implemented.
S10	Delay Hangup	Value is saved, but currently the feature is not implemented.
S11	DTMF	Value is saved, but currently the feature is not implemented.
S12	ESC Guard Time	Value is saved, but currently the feature is not implemented.
S18	Test Timer	Value is saved, but currently the feature is not implemented.
S25	Delay DTR	Value is saved, but currently the feature is not implemented.
S26	Delay RTS to CTS	Value is saved, but currently the feature is not implemented.
S30	Inactivity Timer	Value is saved, but currently the feature is not implemented.
S37	Set Line Speed	See **Baud Table** below. Saves to EEPROM immediately.
S38	Delay Forced	Value is saved, but currently the feature is not implemented.

Baud Table

Code	Speed
0, 1, 2, 3	300 Baud
5	1200 Baud
6, or any number not listed here	2400 Baud
8	9600 Baud
10	14400 Baud
11	28800 Baud
12	57600 Baud
13	115,200 Baud

Example: **ats37=8** would set baud rate to 9600 and save automatically to EEPROM.

3.3 Factory Reset

To perform a "Factory Reset", you must issue *AT&F* to the modem. The first time you do this it will prompt you to enter *AT&F* again to confirm your intent to reset the modem to default. Entering any other command will abort the reset.

3.4 Non-Standard Commands

Command	Description	Notes
S90	DCD Inverted	0 = Normal, 1 = Invert. Saves to EEPROM immediately.
S100	Set Phonebook 0	Sets a hostname or IP address to phonebook entry 0.
S101	Set Phonebook 1	Sets a hostname or IP address to phonebook entry 1.
S102	Set Phonebook 2	Sets a hostname or IP address to phonebook entry 2.
S103	Set Phonebook 3	Sets a hostname or IP address to phonebook entry 3.
S104	Set Phonebook 4	Sets a hostname or IP address to phonebook entry 4.
S105	Set Phonebook 5	Sets a hostname or IP address to phonebook entry 5.
S106	Set Phonebook 6	Sets a hostname or IP address to phonebook entry 6.
S107	Set Phonebook 7	Sets a hostname or IP address to phonebook entry 7.
S108	Set Phonebook 8	Sets a hostname or IP address to phonebook entry 8.
S109	Set Phonebook 9	Sets a hostname or IP address to phonebook entry 9.
S110	Set Phonebook 0	Sets spoof phone number for phonebook entry 0.
S111	Set Phonebook 1	Sets spoof phone number for phonebook entry 1.
S112	Set Phonebook 2	Sets spoof phone number for phonebook entry 2.
S113	Set Phonebook 3	Sets spoof phone number for phonebook entry 3.
S114	Set Phonebook 4	Sets spoof phone number for phonebook entry 4.
S115	Set Phonebook 5	Sets spoof phone number for phonebook entry 5.
S116	Set Phonebook 6	Sets spoof phone number for phonebook entry 6.
S117	Set Phonebook 7	Sets spoof phone number for phonebook entry 7.
S118	Set Phonebook 8	Sets spoof phone number for phonebook entry 8.
S119	Set Phonebook 9	Sets spoof phone number for phonebook entry 9.
S200	Set Baud Rate	Accepts baud rate as whole number, but does not save to EEPROM
S300 - S305	Bytes of MAC address	Saves immediately.
S306 - S309	Bytes of IP Address	Saves immediately.
S310 - S313	Bytes of Gateway Address	Saves immediately.
S314 - S317	Bytes of DNS Address	Saves immediately.
S318	Use DHCP	0 = No, 1 = Yes, Saves immediately.

3.5 Phonebook Usage

Example: **ats104=derp.com:3600** will store hostname or IP address to phonebook entry 4.
ats114=5551234 will associate a made-up phone number with hostname or IP address stored in phonebook entry 4.
This allows you to “dial” a phone number rather than an IP address, i.e **atdt 5551234** instead of **atdt derp.com:3600**. This is handy where a program or a game will only accept a phone number.

3.6 Set MAC Address

To set a MAC address for your Hayesduino, each byte of the address must be entered in decimal and stored at a separate location from S300 to S305. Hexadecimal to decimal calculators can be found by searching online.

Example: To set a MAC address of **90:A4:E1:42:35:2D** enter the following in a terminal:

ats300=144 (where 144 is the decimal equivalent of hexidecimal value of 90)
ats301=164
ats302=225
ats303=66
ats304=53
ats305=45

3.7 Manually Set IP Address

As with setting the MAC address, when manually setting an IP address for your device, each byte of the IP address must be entered in decimal and stored at a separate location from S306 to S309.

Example: To set an IP address of 192.168.1.30 enter the following in a terminal:

ats306=192
ats307=168
ats308=1
ats309=30

3.8 Manually Set Gateway Address

Same as setting the IP address except each byte is stored in locations S310 to S313.

3.9 Manually Set DNS Address

Same as setting the IP address except each byte is stored in locations S314 to S317

4. Hardware Requirements

1. Arduino Mega 2560
2. Ethernet Shield (specifically shields that use the WIZNet W5100 chip)
3. RS232 to TTL converter (for computers with non-TTL RS232 serial ports such as the Amiga. Please see point 5.2)
4. Plug to socket jumper wires (please see wiring under point 5.2)

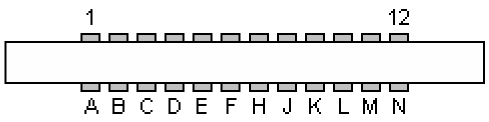
NOTE: The Hayesduino code has grown considerably and will no longer fit on an Arduino Uno.
Code has been added to the Hayesduino project to use a 240x320 LCD touch screen but this currently incomplete and non-functional.

5. Platform Specific Docs

5.1 Commodore User Port

The Commodore userport is not a true RS232 device, but RS232 communication is implemented in software in the Kernal of the computer. The userport provides all of the lines necessary for RS232 communications, but it is all TTL-level.

Here is a breakdown of the userport pins used to emulate RS232 communications.



Commodore Func.	Direction	Modem Func.	Pin	Name	Description	Mega 2560
GND		GND	A	GND	Protective Ground	GND
RxD	<--	TxD	B + C	FLAG2 + PB0	Receive Data (Must be applied to both pins)	19
RTS	-->	CTS	D	PB1	Ready to Send	22
DTR	-->	DCD	E	PB2	Data Terminal Ready	23
RI	<--	RI	F	PB3	Ring Indicator	24
DCD	<--	DTR	H	PB4	Data Carrier Detect	25
CTS	<--	RTS	K	PB6	Clear to Send	26
DSR	Unused	DSR	L	PB7	Data Set Ready	27
TxD	-->	RxD	M	PA2	Transmit Data	18
GRND		GRND	N	GRND	Signal Ground	GND

credit: http://www.gamesx.com/hwb/co_C64Rs232UserPort.html

5.2 RS232

The Arduino AVR series boards use TTL-level RS232 communications (Arduino ARM series use 3.3V). Some computers, such as the Commodore 8-bit line of computers, use TTL-Level serial, but most use RS232-level serial communications. If your target platform uses RS232-level serial you will need to construct a proper RS232 circuit. A MAX237 chip can be used as it allows all 7 used lines to be level-shifted with a single chip.

Alternatively you can use MDFLY's "RS232 Serial Port to TTL Converter Module LED DB9 Breakout Full Signals" which uses the SIPEX SP3238 chip.

This can be bought here:
https://www.amazon.com/Serial-Converter-Module-Breakout-Signals/dp/B07TCN5419

Wiring connections between the Arduino Mega 2560 and MDFLY's RS232 to TTL converter is as follows:

Mega 2560	**RS232/TTL Converter**
Pin 18	RxD
Pin 19	TxD
Pin 22	RTS (mislabelled as RST)
Pin 23	DTR
Pin 24	RI
Pin 25	DCD
Pin 27	CTS
5V	VCC
GND	GND

6. License

Hayesduino is released under the LGPL. Why? Because EthernetClient and EthernetServer both had to be modified because they essentially suck, and these files are under the LGPL. I started to write my own drivers but ran out of time (and will) to do it right, so I decided to go this route.

7. Compiling Notes

1. The Hayesduino project can be compiled and uploaded to the Mega 2560 using the official Arduino IDE software: https://www.arduino.cc/en/software
   Contents of the Hayesduino project must be in a folder named "Hayesduino" (case sensitive) otherwise it will not compile correctly.

2. This project specifically uses Ethernet Library 1.x.x and will NOT work with v2.0.0 or later. The latest version 1.1.2 can be downloaded through the IDE by going to Tools → Manage Libraries… and searching for "Ethernet". In Windows the Ethernet Library files will be downloaded to your Documents folder under Arduino\libraries\Ethernet.
   Full path: C:\Users\username\Documents\Arduino\libraries\Ethernet

3. You will need to copy the customized Ethernet library files from the Hayesduino project folder (under libraries\Ethernet\src) and replace the official Arduino Ethernet library in your Documents folder under Arduino\libraries\Ethernet\src.

4. Buffer size is currently set to 1024 bytes for each of the 4 sockets. This can be changed to either 2048 or 4096 bytes by changing the value in line 28 in w5100.h
   i.e BUFFER_2048
   NOTE: Because of the total 8KB limitation of the W5100 chip, setting the buffer to 4096 bytes will only be assigned to the first 2 sockets.

5. Hayesduino is currently set to use port **6300**. You may need to forward this port in your router. The port number can be changed to anything you like in line 10 of the Global.h file.

To upload the Hayesduino project to your Arduino Mega 2560:

1. Connect the Arduino Mega 2560 to your PC via USB

2. Start the Arduino IDE software and go to File → Open and locate the Hayesduino.ino file. Alternatively, go to the Hayesduino project folder and double click on the Hayesduino.ino file and it will auto run the Arduino IDE.

3. Go to Tools and make sure that that the “Arduino Mega or Mega 2560” board is selected. Processor should be “ATmega2560”.
   Under Tools → Port, tick the COM port that you Mega 2560 is connected to.

4. Press the Verify button (the “tick” button in top left hand button) to verify that there are no errors.
   If all is good then the following message (or something like it) will appear in white in the console at the bottom:

   “Sketch uses 36880 bytes (14%) of program storage space. Maximum is 253952 bytes.
   Global variables use 2501 bytes (30%) of dynamic memory, leaving 5691 bytes for local variables. Maximum is 8192 bytes.”

   If you see any errors (usually in red) make sure you have followed steps 1 to 3 closely in the Compiling Notes and then try again.

5. Press the Upload button (button with right arrow) and wait until finished.