TheNet X-1J

Configuration instructions

1. Introduction

This document describes the build process for creating a ROM image for TheNet X-1J. This process differs from the previous versions of TheNet-X in that it is delivered as two files rather than three. This is in response to a number of requests for a simpler process. In addition the patcher has been considerably changed and utilities for hex conversion are included.

2. Files.

The ROM image comes as two files,

THENET1.X1J
THENET2.X1J

These two files are loaded into memory as described below.
Before loading them however, both should be configured as described in section 3.

In addition, the following files are also provided :

PATCH.EXE
INTEL.EXE
MOTOROLA.EXE
INTEL.C
MOTOROLA.C

PATCH.EXE is the windowing patch utility for the ROM images. INTEL.EXE and MOTOROLA.EXE are utilities that are designed to convert binary files into hex notation, in the Intel Intellec and Motorola S formats.

The ROM image consists of two halves, one for the lower half of a 512K EPROM, and one for the upper half. The files are loaded as shown :

FILENAME	Load starting at hex
THENET1.X1J	**0000**
THENET2.X1J	**8000**

No information on how to load the files into a programmer is presented as all are different. Typical scenarios are however given in section 5.

3. Configuration

Each of the two halves of the ROM image contains two different parts, a common set of drivers & interrupt routines and part of the functionality of the node. Part 1 contains the level 2, 3 and 4 software. Part 2 contains the switch. Each must be patched in an identical way to reflect the desired operation as each part contains an identical section at the start of the file for configuration data. This patching may be done manually or it may be done with the patcher.

The first part of the ROM images is identical to TheNet 1.01 in its configuration. These parameters are followed by additional ones for the extended version as shown :

LOCN	HEX SIZE	FIELD DESCRIPTION
003B	6 BYTES	CALLSIGN OF THE NODE
0041	BYTE	SSID OF THE NODE CALLSIGN
0042	6 BYTES	ALIAS OF THE NODE
004A	WORD	MIN AUTO UPDATE QUALITY
004C	WORD	HDLC DEFAULT ROUTE QUALITY
004E	WORD	DEFAULT RS232 ROUTE QUALITY
0050	WORD	INITIAL NODE LIFETIME
0052	WORD	MIN LIFETIME TO BROADCAST
0054	WORD	BROADCAST INTERVAL IN SECONDS
0056	WORD	LEVEL 3 TIME-TO-LIVE INITIALISER
0058	WORD	LEVEL 4 TIMEOUT IN SECONDS
005A	WORD	LEVEL 4 RETRIES
005E	WORD	LEVEL 4 ACK DELAY IN SECONDS
0060	WORD	LEVEL 4 WINDOW SIZE
0062	WORD	NUMBER OF BUFFERED FRAMES PER CONNECTION
0064	WORD	NO ACTIVITY TIMEOUT IN SECONDS
0066	WORD	PERSISTENCE
0068	WORD	SLOT TIME IN TENS OF MILLISECONDS
006A	WORD	LEVEL 2 INITIAL T1 COUNTER IN SECONDS
006C	WORD	LEVEL 2 WINDOW SIZE
006E	WORD	LEVEL 2 RETRIES
0070	WORD	LEVEL 2 INITIAL T2 COUNTER
0072	WORD	LEVEL 2 INITIAL T3 COUNTER
0074	WORD	LEVEL 2 DIGIPEAT ENABLE FLAG 0 = DISABLED, 1 = ENABLED
0076	WORD	CALLSIGN VALIDATION, 0 = OFF, 1 = ON
0078	WORD	BEACON MODE, 0 = OFF, 1 = AFTER TRAFFIC, 2 = ALWAYS
007A	WORD	CQ ENABLE FLAG, 0 = DISABLED, 1 = ENABLED
007C	BYTE	FULL DUPLEX FLAG, 0=SIMPLEX, 1 = DUPLEX
007D	BYTE	SEND FLAGS IF NO DATA NEEDED, 1 = YES
007E	BYTE	COMMAND LEAD-IN CHARACTER (ESCAPE)
007F	BYTE	TRANSMIT KEY-UP DELAY, 10's OF MILLISECS
0080	80 BYTES	DEFAULT PASSWORD
00D0	BYTE	NULL BYTE AT END OF PASSWORD
00D1	80 BYTES	INFORMATION MESSAGE
0121	BYTE	NULL AT END OF INFORMATION STRING
0122	WORD	CW REPEAT RATE IN SECONDS. 0 DISABLES
0124	BYTE	CW BIT SPEED IN 10's OF MILLISECONDS 6 = 20 WPM
0125	BYTE	DEFAULT HOST MODE. 0 = NORMAL 1 = HARDWARE HANDSHAKE CONNECT CONTROL
0126	BYTE	CROSSLINK PROTOCOL MODE CONTROL 0 = TheNet NORMAL CROSSLINK PROTOCOL 1 = USE KISS INSTEAD OF CROSSLINK 2 = AS PER 1, ALSO COPY NON NODE PKTS

3 = AS PER 2 BUT COPY ALL PACKETS
0127 BYTE MHEARD LIST LENGTH. 0 = OFF, MAX = 100
0128 BYTE NODE BROADCAST CONTROL. 0 = NO BCAST
1 = HDLC PORT ONLY, 2 = RS232 PORT ONLY
3 = BOTH PORTS
0129 WORD RS232 NODE BROADCAST INTERVAL (SECS)
0 DISABLES
012B BYTE NODE BROADCAST ALGORITHM CONTROL.
BIT 0 = HDLC, BIT 1 = RS232
WHEN BIT SET, IMPLEMENT VARIANT ALGO.
012C 8 BYTES OPTIONAL BEACON DIGI LIST, NULL TERM.
0134 WORD DEFAULT BEACON INTERVAL IN SECONDS
0136 BYTE CONNECT REDIRECTION, 0=HOST 1=BBS 2=DXC
0137 BYTE HASH NODE CONTROL. Each bit controls whether
nodes whose alias starts with a '#' are included in node broadcasts on a specific port. Bit 0 determines port 0 (the radio), bit 1 controls the RS232 port. If a bit is set, hash nodes are not broadcast.
0138 4 BYTES THIS IS THE NODE'S AMPRNET ADDRESS.
It is a numeric address of 4 bytes. Each byte corresponds to one byte of the address, for example if the address is 44.131.16.31, then the data stored at each of the bytes 138, 139, 13A and 13B respectively will be 1F, 10, 83 and 2C. Contact your local co-ordinator for an address.
013C 4 BYTES THIS IS THE AMPRNET ADDRESS USED BY THE NODE TO
RECOGNISE BROADCASTS. The data is stored in the same way as for the node's address (as shown above). A typical address would be 44.131.0.0 for the UK.
0140 BYTE IP PORT MODE CONTROL.
This byte controls the default modes used for AX.25 frames on each port. It is bit mapped, with bit 0 controlling the radio port and bit 1 controlling the RS232 port. If a bit is set, the default mode for that channel is datagram (UI frame), if not it is virtual circuit.
0141 BYTE IP INITIAL TIME TO LIVE
0142 BYTE IP ENABLE FLAG.
If zero, the IP router is disabled.
If not zero the IP router is operational.
0143 BYTE HELP MESSAGES CONTROL BYTE.
Each bit enables or disables a different type of help message as follows :

Bit 0 - 'trying to connect' message
Bit 1 - sysop sees all commands in help
Bit 2 - give a 'good-bye' message to users
Bit 3 - enables the connect text message
Bit 4 - show nodes as alias:callsign
Bit 5 - pass 8 bit data in TALK if set
Bit 6 - Make node alias handling case sensitive
0144 WORD MTU_IP0
This is the MTU for port 0 Level 2 AX.25 (the radio port) for the IP router.
0146 WORD MTU_IP1
This is the MTU for port 1 Level 2 AX.25 (the RS232 port) for the IP router.
0148 WORD MTU_IPN

This is the MTU for the Net/Rom subnetwork layer. It should NOT exceed 236 for compatibility with Net/Rom

014A WORD MTU_I_MAX

This is the maximum number of data bytes, plus one, that will be accepted in a received L2 AX.25 packet. Above this will cause an error.

014C WORD MTU_L2_MAX

This is the absolute limit on the number of bytes in a received AX.25 packet, counting the data, control, and address information. It is set to 328 usually (256 data bytes, 2 control and 70 address).

014E BYTE Auto re-connect to node on remote disconnect if 1

014F BYTE NoSlime -

Bit 0, if set, stops 'slime trails' being displayed in the nodes table. Bit 1, if set, stops slime trails being accepted by the node.

0150 BYTE Bit 0 if set bars digi L2 connects to the node. Bit 1 if set bars digi downlinking from the node.

0151 BYTE DEVIATION METER DEFAULT.

This is the default value for the RX Deviation meter.

The patch utility will not assist in changing the help text. That text is positioned at the end of THENET2.X1J. It is a null terminated string of characters. Newlines are represented by the value 0xd (decimal 13). It can be as long or as short as you like, but don't forget that it causes the node to be a source of data and if very long could crash the node. (Not likely in this version given the space available!).

Any problems, give me a ring !

4. The PATCH utility

The patch utility is designed to help configure the two ROM images in a manner that is not as user hostile as hand crafting a binary image. It is invoked as follows :

PATCH [file1 file2]

If no parameters are given, it will look for files THENET1.X1J and THENET2.X1J in the current directory. It will stop if it cannot load them. If the images are renamed, they may be given as parameters. If this is done, both files must be given, with file1 corresponding to part 1 and file2 corresponding to part 2. The program is menu driven, with extensive help provided on the operation of the program and each parameter. It also tries to make sure that only valid data is entered.

The program may also be instructed to load and save textual representations of the parameters. These consist of ASCII files, with one parameter per line. Each parameter consists of the name of the parameter, and equals sign, and the value for that parameter. The values are mainly numeric, with the obvious exceptions of things like the callsign, alias, password, info message etc. To get an example of the format, use the patcher to dump a file and look at it. The idea of this is not simply to load and dump whole images, but to load partial configurations such as passwords & info messages only or parameters only. The file may be edited to remove or add lines as desired. Note that each parameter MUST only occupy one line. For the information message, whitespace before the first printable character is ignored by the program, and if a newline is to be included, it is denoted by the sequence \m (i.e. backslash followed by the letter m). Similarly, to include the backslash character itself, a double backslash must be entered, i.e. \\.

5. Programming examples

There are two utilities included to facilitate conversion to hex for use in programming eproms. The source of both is also included if anyone wants a different file type. The programs have been compiled with Turbo C++.

Each has the same method of invocation,

INTEL infile outfile [address]

or

MOTOROLA infile outfile [address]

These create INTELLEC or S1 type records respectively. Each reads an input binary file and outputs a hex version. The starting address assumed for the file will be 0000 unless specified otherwise in the command line.

5.1 Intel format, loading as two halves

1. Use the patch program to create the desired image.
2. execute :
   INTEL TheNet1.x1j part1
   INTEL TheNet2.x1j part2

3. load part1 into the programmer and program the lower half of the ROM.
   load part2 into the upper half.

5.2 Motorola format, loading as one image

1. Use the patch program to create the desired image.
2. Execute :
   MOTOROLA TheNet1.x1j part1
   MOTOROLA TheNet2.x1j part2 8000
   COPY part1+part2 romimage

3. Edit romimage with a text editor to remove the intermediate end of file (S0...) marker.
4. Load romimage into the EPROM in one go & program it.

6. Acknowledgements

Intel and Intellec are trademarks of the Intel corporation
Motorola is a trade mark of the Motorola company.

My thanks to KH6ILT for the bug fixes to MOTOROLA.C