All About AdamNet – Reference

This is documentation gleaned from my experiences of developing the FujiNet for the Coleco Adam. This document attempts to describe the operation of the AdamNet Protocol, which is necessary for creating the various devices that comprise the FujiNet.

AdamNet Physical description

AdamNet is an asynchronous serial interface connecting the Adam to various peripherals, such as the printer, keyboard, and mass storage devices. It is asynchronous, having a single wire shared for both receipt and transmission of data, while also providing wires for resetting all connected devices, a positive 5 volt power source with a maximum current draw of 40 milliamps, and the rest of the wires being attached to ground. The physical connection for these wires is an RJ12 connector, and cables used in this network are expected to be crossed, that is, the order of the pins reversed on both sides of the cable.

Format of bits on the data wire

The AdamNet protocol, as mentioned previously, shares a single wire on each connected cable for both transmission and reception. On this wire is transmitted an asynchronous protocol where a binary 0 is transmitted by pulling the voltage to 5 volts, and a binary 1 is transmitted by pulling the signal back to ground. This makes the serial bus inverted compared to other contemporary serial buses such as TTL RS-232.

Each individual bit cell is timed at 16 microseconds in length.

Each complete byte of serial data consists of a start bit, represented as a zero, followed by eight data bits, followed by a stop bit, represented as a 1. This makes 10 total bits for each 8-bits of data, consistent with most contemporary serial interfaces, and makes it easy to use a standard serial shift register on a UART to read and write the resulting data.

With 10 bit cells, this puts the total transmission time of each individual byte at 160 microseconds, giving an effective transmission rate of 62,500 bits per second, or 62.5 kilobits per second.

A note on Bus contention

Since the receiption and transmission of data happens on a single shared wire between different devices, bus contention is possible. If two devices try to send data at the same time, the resulting contention of signal is OR'ed together. This most commonly shows up as a framing error, such as when the node responds with an acknowledgement too quickly, to resulting distortions of the received data by the master or peripheral if the contention occurs during the data portion of the frame.

Some basic terms

- **Packet** - A single piece of information sent from an end-point. Each packet contains its address/type byte, and its subsequent payload, followed by a checksum.
- **Master** - The Adam Computer is the master on the network. It is expected that the Adam initiate all network transactions, with peripherals acting as nodes, responding as needed.
- **Node** - each peripheral exists as a node on the network. Nodes do not initiate communication, and can only respond to communications from the Master.
- **Address** - each packet of information has a destination address for a peripheral node. There are fifteen possible addresses, from 1 to F in hexadecimal.
- **Type** - each packet of information has a type, indicating its function. There are fourteen possible types defined in the 6801 controller.
- **MN__** - This indicates that the master sends a given packet type to a node (a control packet).
- **NM__** - This indicates that a node sends a given packet type to the master (a response packet).

NOTE ABOUT NUMERIC NOTATION: Most numbers should be assumed to be hexadecimal, unless phonetically enumerated, or are timing values, in which case, decimal is used.

How packets are framed

The AdamNet was originally designed around a specific design feature present on the serial ports of the Motorola 6801 microcontroller, the serial status and control register (present at address 011H, see the Motorola 6801 Data Book) has a bit that when set, will ignore any further serial data being received on the line, until 10 consecutive 0 bits (start, data, stop bits) are received. This causes the 6801 to clear this register, and it is used to cause peripherals to ignore data not meant for them. Since FujiNet and most modern UARTs do not have the ability to emulate this feature, a simple idle timer of 160 microseconds is used. Packets can be assumed to be complete when this delay passes with no traffic on the bus.

Packet Checksum

If a packet has a payload, a single checksum byte is appended to the end of the payload data. This is calculated from the payload section of the packet, and does not include the command or message size bytes in its calculation.

It can be calculated with the following C code

```
uint8_t adamnet_checksum(uint8_t *buf, unsigned short len)
{
    uint8_t checksum = 0x00;
```

```
    for (unsigned short i = 0; i < len; i++)
        checksum ^= buf[i];

    return checksum;
}
```

How packets are differentiated and addressed

Each packet of data begins with a single byte, containing both the peripheral to be addressed in the bottom four bits of the nibble (bits 0-3), and the upper four bits of the nibble containing the packet type. This gives a total of fifteen possible devices (as 0 is master), and sixteen possible packet types, although the firmware in the 6801 only defines fourteen of them.

```
7    3    0
-----------
|TYPE|ADDR|
-----------
```

- 11 would indicate a CONTROL.STATUS command addressed to device 1, the keyboard.
- D4 would indicate a CONTROL.READY command addressed to device 4, the first disk drive.

Responding to packets

Nodes are expected to respond with RESPONSE (NM) packets if they need to. The response time varies depending on the command, but the Adam typically expects to see the response within at least 1 full byte time after the CONTROL command is sent, a minimum of 80 to 150 microseconds.

List of Packet Types

There are 14 different packet types implemented in the 6801 firmware, which can be understood via AdamNet:

Upper Nibble	Description
1	CONTROL.STATUS (MN_STATUS)
2	CONTROL.ACK (MN_ACK)
3	CONTROL.CLR (MN_CLR)
4	CONTROL.RECEIVE (MN_RECEIVE)
5	CONTROL.CANCEL (MN_CANCEL)
6	CONTROL.SEND (MN_SEND)
7	CONTROL.NAK (MN_NAK)
8	RESPONSE.STATUS (NM_STATUS)
9	RESPONSE_ACK (NM_ACK)

Upper Nibble	Description
A	RESPONSE.CANCEL (NM_CANCEL)
B	RESPONSE.SEND (NM_SEND)
C	RESPONSE.NAK (NM_NAK)
D	CONTROL.READY (MN_READY)

The MN_ and NM_ designations come from the 6801 listings, and refer to the direction of said packet types. MN means Master-to-Node, and NM means Node-to-Master.

CONTROL.STATUS (1X)

- **From:** Adam
- **To:** Peripheral

This packet is sent when the Adam requests the status of a device. This packet has no payload, and thus is the single type/address byte.

For example:

- 11 - Ask for status from the Keyboard
- 12 - . . . from the printer

RESPONSE

The corresponding response packet is RESPONSE.STATUS (NM_STATUS) and it is expected within 80 to 200 microseconds of the CONTROL.STATUS packet. A successful response will result in a CONTROL.ACK packet. If it is not received within this time period, it is ignored.

WHEN

When the EOS Scan for Devices command is sent from the Adam (for example, when the system is reset), Adam will send CONTROL.STATUS for all 15 possible addresses on the AdamNet, listening for responses in between each one.

CONTROL.ACK (2X)

- **From:** Adam
- **To:** Peripheral

This packet is sent by Adam whenever the data returned by the node's response packet has a valid checksum. Like CONTROL.STATUS, there is no payload.

For example:

- 24 - ADAM Accepts the response from the first disk drive.
- 22 - ADAM accepts the response from the printer.

CONTROL.CLR (3X)

- **From:** Adam
- **To:** Peripheral

This packet is sent by the Adam, whenever it wants to signal to the node that it is clear to send content.

For example:

- 34 - ADAM is ready to receive the requested block of data from the disk drive.
- 31 - ADAM is ready to receive the most recent keypress from the keyboard.

WHEN

This is most commonly done when a CONTROL.RECEIVE packet is acknowledged by the peripheral, indicating that the Adam is now ready to receive the content for which it asked.

RESPONSE

A CONTROL.CLR response typically is followed immediately by a RESPONSE.SEND packet, containing the requested content. This response can occur immediately upon receipt of the CLR command. If the peripheral isn't ready, the ADAM will repeatedly send CONTROL.READY packets, every few milliseconds, until the a RESPONSE.ACK is sent, at which point, the peripheral can send the RESPONSE.SEND packet.

CONTROL.RECEIVE (4X)

- **From:** Adam
- **To:** Peripheral

For example: * 44 - ADAM is asking to receive the previously requested block (via a CONTROL.SEND packet) from the disk drive. * 41 - ADAM wants to receive a response from the keyboard.

WHEN

A CONTROL.RECEIVE packet is sent by the Adam whenever it wishes to receive data from the peripheral. This can be done by sending one of the EOS START READ commands.

This command has no payload.

RESPONSE

It is expected that the peripheral send a RESPONSE.ACK packet when the peripheral can send back a response to the data. If a RESPONSE.ACK isn't received

within 2 milliseconds, the Adam will continue to send CONTROL.RECEIVE commands until acknowledged, or the controlling program sends an EOS STOP READ command

If the peripheral does not wish to provide a response (such as if there is nothing to report), the peripheral can simply respond with a RESPONSE.NAK packet, and the Adam will stop trying to solicit a response.

CONTROL.CANCEL (5X)

- **From:** Adam
- **To:** Peripheral

For example:

- 54 - Adam requests that the current disk operation be cancelled.
- 58 - Adam requests that the current tape operation be cancelled.

This packet is sent when the Adam requests the current I/O operation to be cancelled. At least, that's how it's defined in the documentation. This packet type, while enumerated in the various 6801 listings, isn't actually used (verified by cross reference), and therefore is considered unimplemented.

(If this is in error, let's get it corrected. Maybe HME has some insight here?)

CONTROL.SEND (6X)

- **FROM:** Adam
- **To:** Peripheral

WHEN

This packet is used to send content to a peripheral, such as requesting a block number, or data from the Adam to be sent to a peripheral for output. It is most commonly triggered via an EOS START WRITE CHARACTER command for character devices, or via an EOS START WRITE or READ BLOCK commands for block devices (which cause it it to emit a send packet with the requested block number).

DETAILS

The Format of a send packet is:

Offset	Description
0	The Address/type (6x)
1-2	16-bit number in little endian form specifying the size of the payload, should be no larger than max size reported by RESPONSE.STATUS

Offset	Description
n	n number of bytes comprising payload
c	8-bit XOR checksum calculated from previous n bytes.

RESPONSE

A peripheral may either respond with a RESPONSE.ACK if the send command packet is valid, or with a RESPONSE.NAK if the send packet is invalid.

EXAMPLE

This is the Adam requesting block 0 from disk drive device 4.

```
64 00 05 00 00 00 00 00 00 94
```

Which is broken down into:

Bytes	Description
64	CONTROL.SEND for device 4
00 05	This send packet is 5 bytes long.
00 00 00 00	The block number
00	Not used (reserved)
94	Checksum

CONTROL.NAK (7X)

A non-acknowledgement. No payload.

- **From:** Adam
- **To:** Peripheral

WHEN

This packet is sent whenever a data transfer from the peripheral is invalid, such as when there is a checksum error or framing error.

RESPONSE.STATUS (8X)

This is the response packet, sent in response to a CONTROL.STATUS command.

- **From:** Peripheral
- **To:** Adam

WHEN

This is the response packet sent by the peripheral in response to a CONTROL.STATUS command. It should be sent no later than 200 microseconds after the receipt of the CONTROL.STATUS packet. A CONTROL.ACK is sent by Adam upon successful checksum of the RESPONSE.STATUS packet. A CONTROL.NAK is sent by Adam if the checksum doesn't match. If the RESPONSE.STATUS packet is sent too late, no response is sent from the Adam.

DETAILS

The format of a RESPONSE.STATUS packet is:

Offset	Description
0	The address type (8x)
1-2	The maximum message size
3	The type of device 0 = Character, 1 = Block
4	Device-specific Status Byte
5	Checksum calculated from maximum message size, device type, and device specific status bytes.

EXAMPLE

This is an example status response from device 4, the first disk drive.

```
84 00 04 01 40 45
```

Which breaks down into:

Bytes | Description
84 | RESPONSE.STATUS for device 4
00 04 | maximum message size of 1024 bytes
01 | This is a block device
40 | Device specific status byte, (40 = no errors)
45 | Checksum calculated from message size, device type, and device specific status

RESPONSE.ACK (9X)

- **From:** Peripheral
- **To:** Adam

An acknowledgement of a corresponding CONTROL packet from the Adam.

This packet has no payload.

WHEN

This packet should be sent by the peripheral when the full control packet is received from the Adam and deemed valid in both content and checksum. The acknowledgement should be sent no later than 400 microseconds after the receipt of a packet from Adam, otherwise Adam may try to re-send the command.

EXAMPLE

- 94 indicates an acknowledgement by disk drive 1 of the previous packet to disk drive 1 from Adam.

RESPONSE.CANCEL (AX)

The complement to CONTROL.CANCEL for the peripheral side. Like CONTROL.CANCEL, I can't find any implementation in the cross references of any of the peripherals or the master 6801 listing, so am assuming this command is also unimplemented.

RESPONSE.SEND (BX)

This is the response packet sent by the peripheral after a CONTROL.CLR has been received. If the checksum and data is correct, then the Adam will respond with a CONTROL.ACK packet, otherwise a CONTROL.NAK packet is sent.

- **From:** Peripheral
- **To:** Adam

WHEN

This is sent immediately after the receipt of a CONTROL.CLR packet from the Adam.

DETAILS

The format of a RESPONSE.SEND packet is

Offset	Description
0	The address/type (Bx)
1-2	The size of the message, should be no larger than the max message size reported by RESPONSE.STATUS
n	The content to send back to the Adam.
c	The checksum of the preceeding n bytes

Example

An example of a RESPONSE.SEND packet from the keyboard indicating a keypress of the RETURN key:

```
B1 00 01 0D 0D
```

Which breaks down into:

Bytes | Description
B1 | RESPONSE.SEND for Device 1
00 01 | Message size is one byte
0D | keyboard code for RETURN (which happens to be ASCII)
0D | Checksum of the one byte.

RESPONSE.NAK (CX)

The RESPONSE.NAK packet is sent by the peripheral when either the checksum or data from the Adam isn't valid, or whenever there is nothing of substance to report (such as when there is no key pressed on keyboard.)

WHEN

In response to either invalid or no data.

EXAMPLE

RESPONSE.NAK is sent back within 160 microseconds of a CONTROL.RECEIVE for the keyboard device (1), if there are no keys pressed.

```
41
... 160 microseconds later...
C1
```

CONTROL.READY (DX)

CONTROL.READY is sent by the Adam to query the peripheral for readiness for the proceeding command. It is expected that the device either return RESPONSE.ACK if it is ready, or RESPONSE.NAK if it is unable to fulfill a ready request.

WHEN

CONTROL.READY is sent periodically by the Adam when it wants to talk to a peripheral, and it's ready to perform the next command. The Adam will keep sending this command with a delay of a few milliseconds, until the target device responds with a RESPONSE.ACK or RESPONSE.NAK command.

This command was added to deal with the issues of slow devices, such as the digital data pack drives, which take time to find blocks to return to the Adam.

EXAMPLE

An example of a CONTROL.READY command is when the Adam is waiting for readiness by the first Adam tape drive, device 8:

```
D8
```

Protocol Examples

In these examples, M: refers to a packet initiated by the master, P: means a packet response from the peripheral (node)

STATUS Request, roll call.

After a reset, the Adam will do a roll call of all devices, setting up device control blocks for each device that returns a valid RESPONSE.STATUS packet.

Time	Byte	Description
0us		Reset happens, RESET line brought low for 2500us
160,000us	M: 11	CONTROL.STATUS for device 1 (keyboard)
160,160us	P: 81 01 00 00 00 01	Response packet
161,192us	M: 21	ACK from Adam. Device accepted.
164,475us	M: 12	CONTROL.STATUS for device 2 (printer)
164,785us	P: 82 10 00 00 00 10	Response packet
166,041us	M: 22	ACK from Adam. Device accepted
169,977us	M: 13	CONTROL.STATUS for device 3 (not there, so 4ms delay, no response)
174,682us	M: 14	CONTROL.STATUS for device 4 (disk drive 1)
174,900us	P: 84 00 04 01 40 45	Response packet
176,154us	M: 24	ACK from Adam. Device accepted.
...	...	Repeat above three steps for all other devices 5 to 15.

Reading Block 0 from a DDP in device 4 (in this case the fujinet boot ddp)

This is the entire conversation of the first block being read from boot.ddp into the Adam. Blocks are expressed as a 32-bit quantity by EOS. The fifth byte is reserved. This also shows the Adam retrying the transaction, as the response to the receive packet takes too long.

Time	Byte	Description
254,114us	M: D4	Is Device 4 ready?
254,447us	P: 94	Device 4 acknowledges
254,738us	M: 64 00 05 00 00 00 00 00 94	Adam asks for block 0000
256,196us	P: 94	Device 4 acknowledges
257,698us	M: 44	Adam asks for a receive of data
259,050us	P: 94	Peripheral acknowledge too late...
264,547us	M: D4	RETRY: Is device 4 ready?
264,888us	P: 94	Peripheral acknowledge in time...
265,187us	M: 64 00 05 00 00 00 00 00 94	Adam asks for block 0000
266,640us	P: 94	Peripheral acknowledge in time...
268,131us	M: 44	Adam asks for a receive of data
268,594us	P: 94	We acknowledge just in time...
268,899us	M: 34	Adam sends CONTROL.CLR to indicate it wants the data
269,144us	P: B4 04 00 78 32 6F FD 3E ..	Peripheral sends RESPONSE.SEND packet with 1024 bytes of data (1028 bytes total)
433,637us	P: .. 19	Peripheral sends last byte (checksum)
433,940us	M: 24	Adam acknowledges block. End of transaction.

Writing block 0 to a DDP in device 4 (in this case the fujinet boot ddp)

Writing a block is similar to reading one:

Time	Byte	Description
+0us	M: D4	Is device 4 ready?
+160us	P: 94	Device 4 acknowledges
+320us	M: 64 00 05 00 00 00 00 00 94	Adam asks to write block 0000
+480us	P: 94	Device 4 acknowledges
+640us	M: 64 04 00 11 22 33 44 55 ..	Adam sends 1028 bytes size block data; checksum to peripheral
+1600us	P: 94	Device 4 acknowledges. End of transaction.

Scanning the keyboard, no keypress

Using EOS START READ KEYBOARD and EOS END READ KEYBOARD cause status requests to periodically be sent via AdamNet to check the keyboard.

Time	Byte	Description
+0us	M: 41	CONTROL.RECEIVE of keyboard
+160us	P: C1	RESPONSE NAK - No Keypress

Reading a keyboard keypress

If a keypress is returned by EOS END READ KEYBOARD, the bus trace looks like this:

Time	Byte	Description
+0us	M: 41	Status of keyboard
+160us	P: 91	RESPONSE ACK - Keypress
+320us	M: 31	CONTROL.CLR - Adam ready for data
+480us	P: B1 00 01 0D 0D	RESPONSE.SEND 1 byte, 0D, checksum
+640us	M: 21	Adam acknowledges character transfer, end of transaction.

Getting the Network Adapter configuration from FujiNet

The FujiNet control device is device F (15), it is a character device, with a maximum message length of 1024 bytes.

The control protocol is very simple, you send the command you want to do, and if the command issues a response (as per documentation) ask for the response.

In this case, the Read Adapter Information command (0xE8) sends the adapter information on the next read character device.

Time	Byte	Description
+0us	M: DF	Is FujiNet device ready?
+160us	P: 9F	Yes, FujiNet device is ready.
+320us	M: 6F 00 01 E8 E8	Send command E8 to fujinet, single byte command.
+480us	P: 9F	FujiNet acknowledges
+960us	M: 4F	Adam asks for a receive
+1120us	P: 9F	FujiNet acknowledges
+1280us	M: 3F	Adam says its ready for data (CLR)
+1440us	P: BF 00 8B 43 68 65 72 72 79 ..	Fujinet sends RESPONSE.DATA packet with 139 bytes of data containing network adapter config as shown in struct below.
+24000us	M: 24	Adam acknowledges data is ok. Transaction done.

```
/**
 * The current network adapter configuration
 */
typedef union {
  struct
  {
    char ssid[32];
    char hostname[64];
    unsigned char localIP[4];
    unsigned char gateway[4];
    unsigned char netmask[4];
    unsigned char dnsIP[4];
    unsigned char macAddress[6];
    unsigned char bssid[6];
    char fn_version[15];
  };
  unsigned char rawData[139];
```

```
} AdapterConfig;
```

A note about block vs character devices

Fundamentally, there is little difference between block and character devices. Block devices simply add an additional SEND and ACK so that the Adam can tell the peripheral which 32-bit block number to deal with.

A note about maximum message size and performing READ

The maximum message size is important. It must be specified as part of the CONTROL.STATUS request, and the same value must be used when setting the maximum message size for an EOS READ BLOCK or CHARACTER DEVICE CALL. If they do not match, the request will be IGNORED. This means that you must have a data structure set up to hold the maximum amount that you have set up for the maximum message size for a given device.

Author

Thomas Cherryhomes, Firmware architect for FujiNet thom dot cherryhomes at gmail dot com