The BPQ Node in an Expanding Network

by Karl Medcalf WK5M, and Phil Anderson WØXI,
in cooperation with John Wiseman G8BPQ

July 20, 1990, Lawrence, Kansas

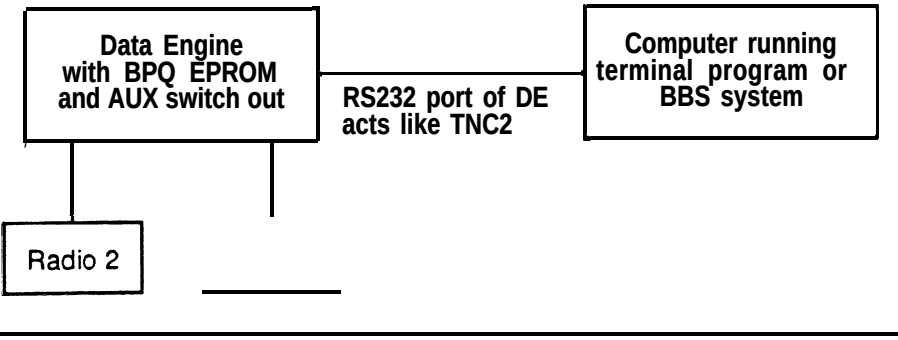
The release of the Kantronics Data Engine, along with the development of their 9600 baud (G3RUH compatible) modem has now added a new dimension to the meaning of a Network Node. With the open architecture of this packet controller, and the flexibility now possible, network expansion and high speed operation is now possible with future expansion in mind.

John Wiseman (G8BPQ) has completed development of his packet switch software in an EPROM version, suitable for installation in the Data Engine. The flexibility written into this EPROM allows you to configure your node to meet your specific requirements. First, let's look at some of the specific features of the BPQ node. This is not a new node, as the G8BPQ packet switch software has been running on numerous systems in its PC based version. Using it in this fashion allows any KISS mode TNC to be a Network Node, providing all the features of Net/Rom nodes, and a few new features.

One of the major advantages of the BPQ node over the other nodes is the "stay" option when connecting to another node or an end user. When this option is specified in the Connect command, the node will not cause a complete disassembly of the network link when the far end of the connection initiates a disconnect. As an example, if you connect to a node named "KSLAW", and then issue a connect of the form "C KSWCH S", this will activate the stay function and connect you to the KSWCH node. From this point you could connect to an end user, or a PBBS near the KSWCH node. Since the stay option is in effect, when you are finished with the PBBS you simply use its BYE command, causing the PBBS to disconnect. When that disconnect is received by the KSLAW node, you will not be disconnected, but you will be sent a message 'Returned to node KSLAW". You may now issue further commands to the node.

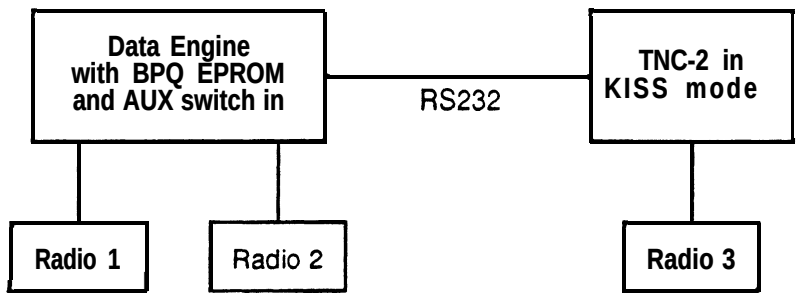
Since the BPQ node software has previously run only in a PC computer (normally co-located with a BBS system) this meant that the node required a computer to be connected and running at all times. Now that the code has been made available in an EPROM version for the Data Engine, the computer is no longer required. Another advantage of this implementation is that since the Data Engine is a dual-port unit, you can now operate a simple two-port Network Node using a single Data Engine with the BPQ EPROM code installed. The serial port of the Data Engine can be configured to appear like a TNC-2, allowing regular use with a simple terminal program, or access by a BBS system, just as though the BBS were connected to a TNC-2. (See figure 1)

Figure 1



Frequently, we have the need for more than two radios in a single node, and this has been cumbersome in past implementations of node firmware. With the BPQ code running in the Data Engine, a simple press of the front panel AUX switch changes the serial port from a TNC-2 port to a KISS link, which may be connected to any KISS mode TNC. In this configuration (see figure 2), you can have a three-port Network node with a single Data Engine using the BPQ code and a TNC-2 in KISS mode. No specific requirements exist for the TNC-2, except that it runs the KISS firmware. Since the Data Engine supports both 1200 and 9600 baud modems, as well as virtually any other known modem, you could configure this three-port node as a gateway for 1200 baud users onto a 9600 baud backbone, or even to a PSK modem for a satellite gateway.

Figure 2

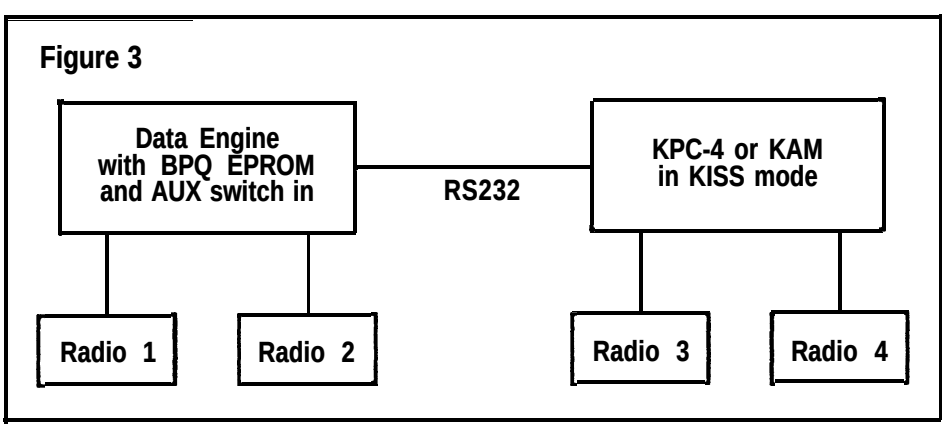


Need more than three ports? Not a big problem with the BPQ/Data Engine combination. A simple 4-port node can be configured using the Data Engine/BPQ combination with a KPC-4 or KAM (KISS mode) connected to the Data Engine serial port. Since the BPQ code allows for dual-port KISS, this provides a low-cost, no-diode implementation of a 4 port configuration. The four ports could all be different radios, bands, speeds and so forth. (See figure 3).

There is another implementation possible due to the incorporation of a new method called "multi-drop" kiss. John Wiseman implemented dual port support using the Kantronics scheme. Since the KISS implementation as originally specified only uses the low nibble (4 bits) of the control byte to specify the function, the high nibble was used by Kantronics in version 2.84 and later of their firmware to address the two ports of the KAM and the KPC-4. If the high nibble was zero, then the unit would address one port, and if the high nibble of this byte was non-zero, it would address the other port.

John has extended this implementation to allow all four bits of the upper nibble to be significant, thus permitting up to 16 different KISS addresses to be defined. He has supplied in his distribution, versions of the KISS code for the various TAPR type TNCs, and documents which byte of this code to change in order to tell each TNC which address it is assigned. Kantronics is also making an EPROM image available for free, non-commercial use, which contains this capability, along with instructions for changing the correct address information. This is available for all Kantronics TNCs direct from the manufacturer's phone-line BBS (913-842-4678).

Figure 3



When using this method, all of the TNCs can be attached to a single serial port, such as the serial port of the Data Engine. A single diode is connected in the RXD line of each TNC, with the anode connected to the TNC and the cathode connected to the serial port of the com-

puter or Data Engine. The BPQ code in the computer or Data Engine then polls each TNC in turn (by address) for information, and the TNC then responds with any data that it has received from its associated radio. No data is sent by the TNCs to the host computer or Data Engine without being polled, so there is no conflict in data on the serial line.

Using this multi-drop kiss method, a possible configuration for a multi-port Network Node could look similar to the one shown in figure 4. Using this method, it is theoretically possible to have one Data Engine with BPQ code (2 radio ports) connected to as many as 16 other TNCs using the multi-drop KISS code, each connected to its own radio. You could conceivable mix and match all makes of TNCs, including the KAM and KPC-4 (which each would use 2 addresses and connect to 2 radios).

In this type implementation however, I suggest connecting the busiest radio channels directly to the Data Engine radio ports, since all other TNCs are polled for data, thus possibly slowing them down slightly.

This seems like a possible way to finally begin to improve our Network efficiency at a reasonable cost, and maintain the flexibility to expand to even higher speeds or new modems as they become available. The open architecture of the Data Engine will certainly lead to the development of other firmware/hardware, and perhaps we can again make packet a viable means for communication in real-time as well as a messaging system.