GMON – a G-TOR™ Monitoring Program for PC Compatibles

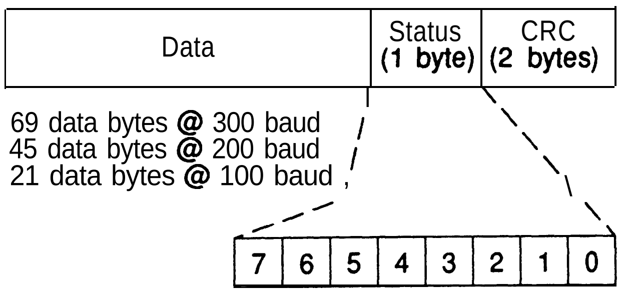
by Richard Huslig and Phil Anderson, WØXI

The G-TOR data communications protocol is an innovation of the technical staff of Kantronics Co., Inc. It was introduced in March 1994 as an inexpensive means of improving point-to-point digital communications in the HF radio bands. It has been implemented in the KAM Plus and KAM-E and is now offered for licensing to other manufacturers.

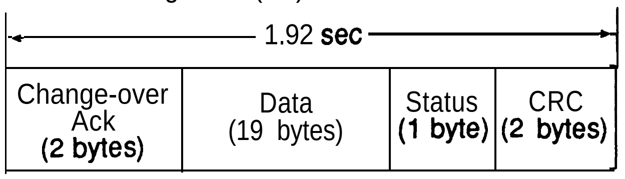
Monitoring of G-TOR frames is difficult since a variety of frame formats exist as shown in Figure 1. Although a data frame is always 1.92 seconds in duration, it might be sent at 100,200, or 300 baud; it might contain real data or Golay parity bits in ASCII or Huffman encoded form; and it may be received in Lower or Upper Side Band. Each data frame must be deinterleaved at the receiver and this consumes a large amount of processing time. Data may exist in BK (BreaK or changeover) frames too; although, they are sent at only 100 baud without interleaving. In addition, no synchronizing flags exist at the beginning or end of the frames. Each data frame contains data, a status byte, and two CRC bytes. The connect and disconnect frames, sent at only 100 baud, contain destination and source address fields in place of the data frame's data field.

Figure 1
G-TOR Frame Structures

G-TOR Frame Structure Before Interleaving

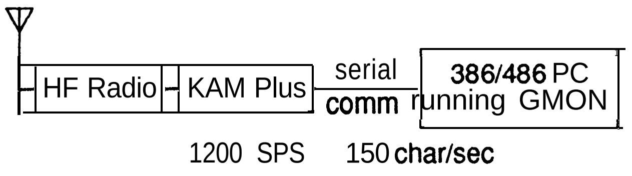


G-TOR Changeover (BK) Frame Structure



Hence, monitoring on the fly – in real time – is best accomplished using a brute force algorithm with the system shown in Figure 2.

Figure 2
G-TOR Monitoring Setup



Frames are received by an HF radio, demodulated by the KAM Plus (or TNC), and processed in 1.92 second segments by GMON, a PC-based basic terminal program that runs on a 386 or 486 compatible and includes G-TOR monitoring capability. The KAM Plus is configured via the GSCAN command to sample the receiver modem digital data 1200 times per second. These data are then shipped continuously via the serial port to the PC where GMON carries out the monitor processing, looking for valid frames at each sample. This paper will describe the development of GMON, provide a functional description of the protocol, and detail the theory of GMON operation.

Background

To realize our goal of developing a terminal program to monitor all G-TOR frames, we first developed GOFF, a program that would monitor G-TOR frames from a file of sampled data. This program, written in BORLAND C, operated offline from the TNC. GOFF performed 100,200, and 300 baud tests and a 100 baud BK test on 1.92 seconds of modem data samples. The baud tests consisted of deinterleaving, packing, CRC checking, Golay decoding, and again CRC checking. The 100 baud BK test consisted of shifting and CRC checking. If the CRC was valid, then the status byte was decoded; if the frame was a connect, disconnect, data, or BK frame, the frame was suitable for display. If the frame was a data frame and the status byte also indicated Huffman coding, then Huffman and RLEn decoding was performed. All other frames with invalid CRC or status byte were discarded.

GOFF began as a brute force algorithm which performed all tests on every sample at 600 SPS (Samples Per Second). A 7027 byte (7027*8/600 = 93.7 sec.) data file took 140 sec. to process and display on a 50 MHz 486DX machine. After analyzing and optimizing the loops with the highest iterations, we cut this time to 105 sec. Then, after optimizing the assembly code generated from the C code, we cut it to 80 sec. Next, we realized that if we cut the number of tests by processing only 2 samples of each bit at each baud rate, we could cut the number of tests per sample from 4 to a little more than 1, even though we had to increase the sample rate to 1200 SPS. By phasing the tests, we cut the time to 60 sec. Next we optimized the deinterleave process by storing the state of the interleave buffer at each baud rate and each bit phase, so that to deinterleave, we simply rotate the interleave buffer and shift in the new sample. This cut the time to 29 sec. Now, removing the CRC test of the inverted data and optimizing the assembly code by using string instructions, we cut the time to 16 sec. Further optimization yielded 14 sec. Processing the same file on a 386DX-33 took 39 sec., and after turning the turbo switch off (i.e. 386DX-8), processing took 85 sec. Processing on a 286-16 took 111 sec. Further speed enhancement is possible if the trace option is turned off. Or, if a valid G-TOR frame is detected by a valid CRC, most processing, except deinterleaving and shifting, on the next 1.92 sec. of data, can be skipped. Also if the same G-TOR frame is detected again by means of the second bit phase or a retransmitted frame which was not acknowledged, only the first frame is displayed.

After the GOFF program was debugged and optimized, then we developed the GMON program by integrating GOFF with a terminal program written in C. Later, we reworked GMON's modules, especially the G-TOR monitoring modules, to follow a specified calling convention so that the terminal program modules handled all data display and logging. The main G-TOR monitoring module was written in BORLAND C, and, therefore, follows the BORLAND C calling convention. The other 2 modules were written in assembly. These 3 object modules were combined into a library file called GMONTER.LIB, which can be linked with any other terminal program which adheres to the BORLAND C calling convention.

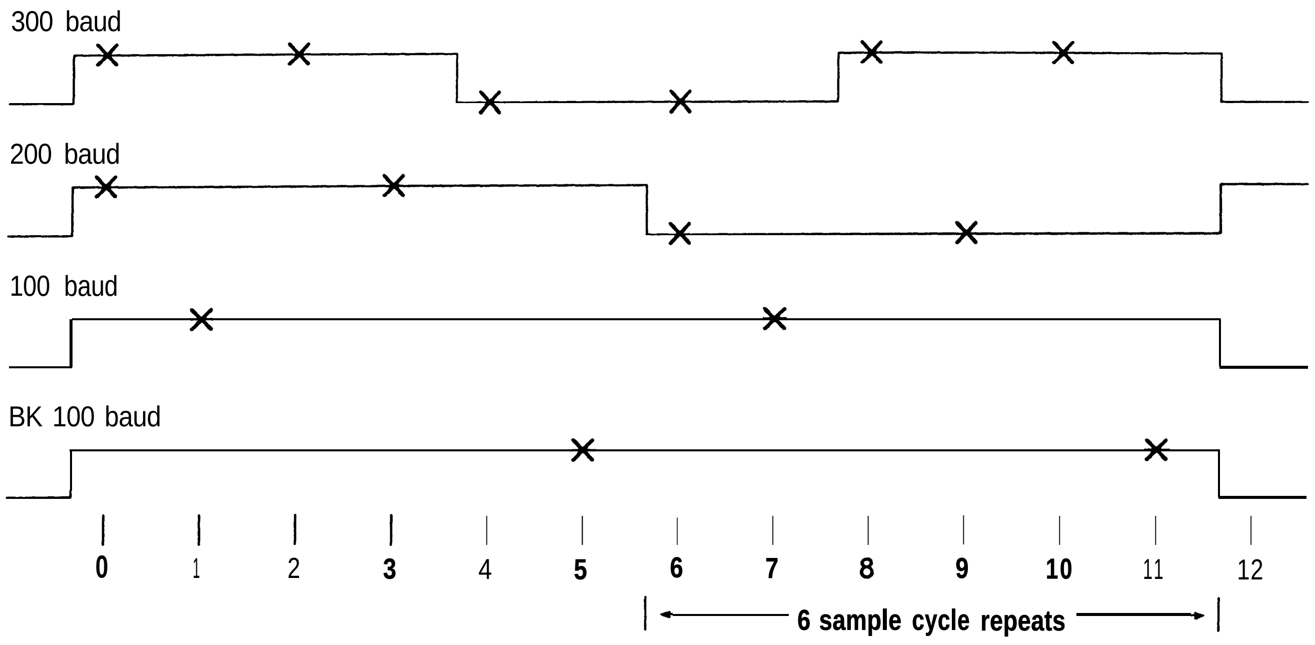
Functional Description of GMON

Sampling is initiated by issuing the GSCAN 1200 command to the KAM. When 8 samples are collected, the character is sent over the RS232 communication port at a constant 150 characters per second. Again, Figure 2 illustrates the G-TOR Monitoring Setup.

GMON receives the character in its interrupt driven receive buffer. If the receive buffer is not empty, the 8 samples are demultiplexed and stored in the sample buffer – one sample per byte.

The sampling and phasing process, illustrated in Figure 3, demonstrates that only 2 samples of the bit are tested at each baud rate and that, typically, only 1 test is performed per sample. The samples are either shifted and stored in one of two BK frame buffers or deinterleaved at 100, 200, or 300 baud and stored in one of 6 interleave buffers depending on the baud rate and bit sample phase. GMON does a 100 baud test every 6th sample (sample phase 1), a 200 baud test every 3rd sample (sample phases 0 and 3), a 300 baud test every other sample (sample phases 0, 2, and 4), and a 100 baud BK test every 6th sample (sample phase 5).

Figure 3
GMON Sampling and Phasing (x marks test performed at sample specified)



This results in a 6 sample cycle. After 1.92 seconds of data have been collected, the samples are either shifted and stored (if sample phase = 5) or deinterleaved and stored in their respective buffers. The order and steps of processing these frames are displayed in Figure 4, the GMON flow chart.

If the sample phase is not 5, then deinterleaved data is packed into a frame buffer. The CRC is calculated from the frame and compared with the CRC stored in the frame. If the CRCs match, then the frame buffer is Huffman decoded if the status byte indicates Huffman encoding. Now the frame buffer is ready to be sent to the video monitor. If the CRCs do not match, then the frame buffer is Golay decoded, stored in the Golay buffer, and CRC tested. If the CRCs match, then the Golay buffer is Huffman decoded if the status byte indicates Huffman encoding. Now the Golay buffer is ready to be sent to the video monitor. If the CRCs do not match, GMON discards the frame. GMON does not attempt to do Golay error correction.

If the sample phase is 5, then the CRC of one of the BK frame buffers (selected by the bit phase) is calculated from the frame and compared with the CRC stored in the frame. If the CRCs match, then the frame buffer is Huffman decoded if the status byte indicates Huffman encoding. Now the frame buffer is ready to be sent to the video monitor. If the CRCs do not match, GMON discards the frame.

Theory of Operation

In this section, the ideas outlined above are expanded in detail.

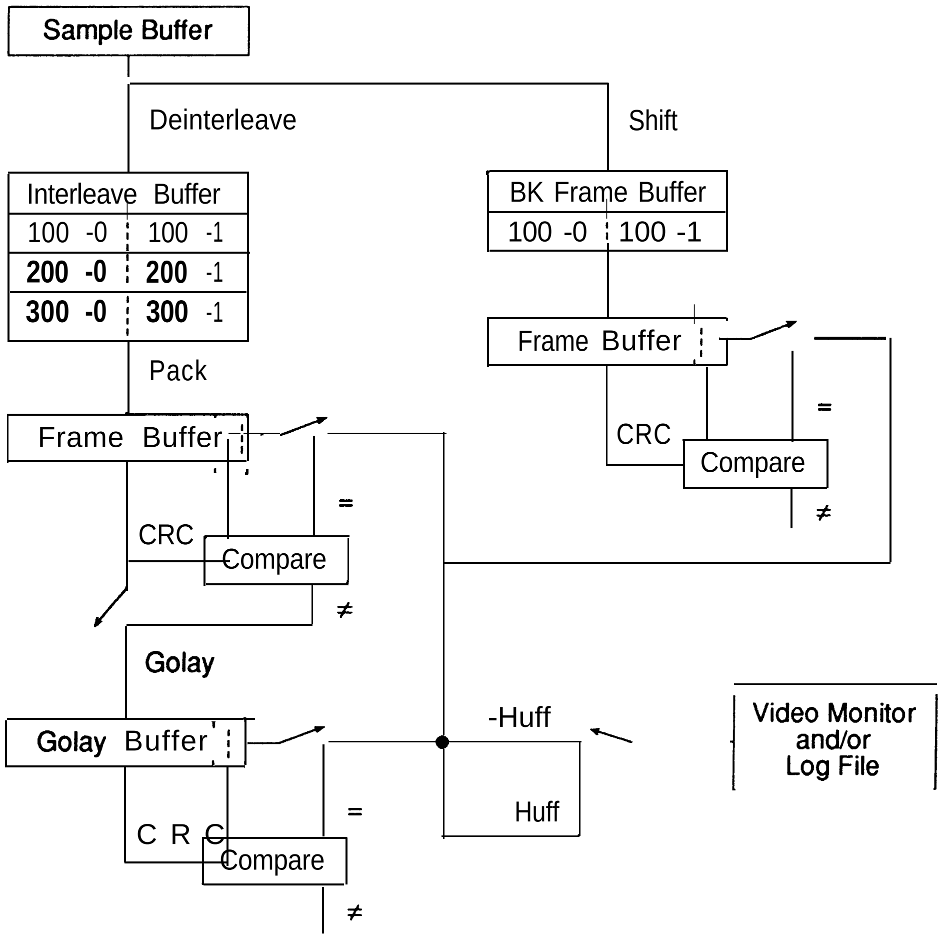
Sampling

GMON receives a character of 8 samples of the modem's data in its interrupt driven receive buffer at 150 characters per second. The 8 samples are demultiplexed and stored in the sample buffer – one sample per byte.

Phasing

Phasing reduces the number of tests per sample and, therefore, increases GMON's speed. GMON processes only 2 samples of the bit at each baud rate. Therefore of the 12 samples every 10 ms, GMON does a 100 baud test on

Figure 4
GMON Flow Chart



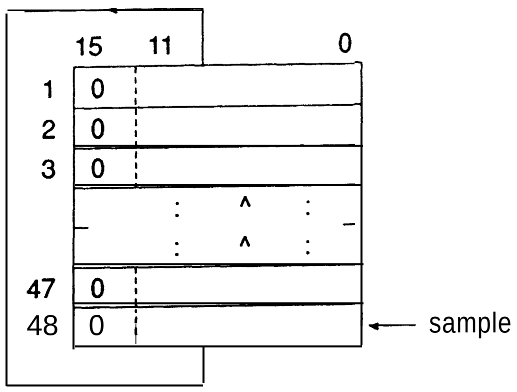
2 of the 12 samples, a 200 baud test on 4 of the 12 samples, a 300 baud test on 6 of the 12 samples, and a 100 baud BK test on 2 of the 12 samples. This establishes a 6 sample cycle:

Sample	Baud Test
6*n+0	300 and 200
6*n+1	100
6*n+2	300
6*n+3	200
6*n+4	300
6*n+5	100 BK

Deinterleaving

GMON's deinterleaving scheme significantly reduces the time needed to deinterleave 1.92 seconds of data. Instead of shifting all samples (576,384,192 for 300,200,100 baud respectively) for each new sample, this scheme, illustrated in Figure 5, rotates the interleave buffer and shifts in only the new sample; each 12-bit word shift is equivalent to 12 bit shifts. The samples are deinterleaved at 100, 200, or 300 baud and stored in one of 6 interleave buffers depending on the baud rate and bit sample phase. Each 300 baud interleave buffer contains 48 12-bit words, while each 200 baud interleave buffer contains 32 12-bit words, and each 100 baud interleave buffer contains 16

Figure 5
Interleave Buffer

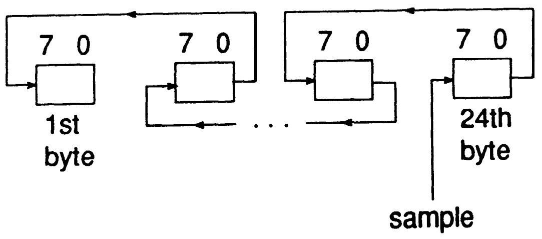


12-bit words. The interleave buffering scheme arranges the samples, which are exactly one bit time away, to be adjacent to one another in the same buffer. This scheme simplifies the deinterleaving task by simply shifting the words up one, bringing the top word down to the bottom and left shifting only the new sample into the LSB of the last word.

Shifting

The bit shifting task for BK frames, illustrated in Figure 6, simply shifts samples into one of two frame buffers depending on the bit sample phase. Since the LSB of the BK frame's first byte is transmitted first, the new sample is right shifted into the MSB of the 24th byte. After 192 bits are shifted in, the first bit occurs at the LSB of the first word.

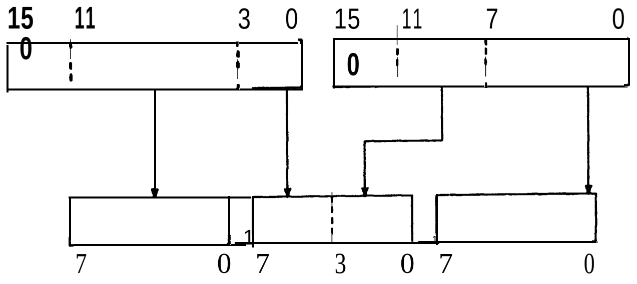
Figure 6
BK Frame Buffer



Packing

The packing task, illustrated in Figure 7, merely packs or formats each pair of 12-bit interleaved data words into three 8 bit bytes in the frame buffer. The 48 12-bit words of the 300 baud interleave buffers pack into 72 bytes of the frame buffer. The 32 12-bit words of the 200 baud interleave buffers pack into 48 bytes of the frame buffer. The 16 12-bit words of the 100 baud interleave buffers pack into 24 bytes of the frame buffer.

Figure 7
Packing 2 12-bit interleave words to 3 frame bytes



CRC Checking

Next the CRC is calculated from the frame and compared with the CRC stored in the frame. Much of the CRC calculation has been reduced to simple table lookup. If the CRCs match, then the frame buffer is tested for Huffman encoding. If the CRCs do not match, then the frame buffer is Golay decoded, stored in the Golay buffer, and CRC tested. If the CRCs match, then the Golay buffer is tested for Huffman encoding. If the CRCs do not match, GMON discards the frame.

Huffman Decoding

The frame or Golay buffer is Huffman decoded if the status byte indicates Huffman encoding. Huffman compressed data frames may contain RLEn (Run Length Encoding) codes. An RLEn code is a 19 bit code made up of a unique 14 bit Huffman code followed by 5 bits which represent a number n, O-31. When an RLEn code is encountered in a data frame, the previous character decoded in the frame should be repeated an additional N times where N is a number which depends on n and the number of bits used by the previous Huffman character. Refer to the G-TOR PROTOCOL for the Huffman table as well as the RLEn coding table.

Golay Decoding

If the CRCs do not match, then the frame buffer is Golay decoded and stored in the Golay buffer. The Golay decoding has been reduced to table lookup for quicker processing. The 4K 12-bit words of this table were generated by matrix multiplication of all combinations of 12-bit word inputs by the Golay Generator Matrix:

```
FFE
6E3
B71
5B9
2DD
16F
8B7
C5B
E2D
717
B8B
DC5
```

Summary

In conclusion the GMON terminal program efficiently monitors G-TOR frames using the fastest assembly language techniques. It also reduces the number of tests by testing only 2 samples of each bit. The deinterleave process is simplified by storing the state of the interleave buffer at each of 3 baud rates and each of 2 bit phases, and therefore only the current sample is shifted in after the interleave buffer is rotated. The CRC and Golay decoding is simplified by table lookup.

GMON's companion offline G-TOR monitoring program, GOFF, can also monitor G-TOR frames from sampled data stored in a disk file. This program is needed for slower PC compatible machines like the PC-XT. Kantronics Co., Inc. also provides GMONTER.LIB, a library of BORLAND C++ compatible object modules, which an external program can call to do the G-TOR monitoring process.