Hyper about Slow Load Times?

... KIM Hypertape is an alternative

Jim Butterfield
14 Brooklyn Ave.
Toronto Ontario M4M 2X5

Whenever I meet a bunch of KIM users in my travels, I’m likely to notice a couple of guys off to one side whispering and gesturing in my direction. This means either of two things: I’m wearing odd socks (again), or I’m being identified as the creator of KIM Hypertape.

Hypertape, often called super tape by KIM users, is indeed a good thing. It speeds up the standard KIM-1 cassette tape interface by a factor of six times. This gives a speed of roughly 50 bytes per second loading or dumping. It saves time and tape. And it’s completely compatible with the KIM cassette tape loader — no extra hardware or software is needed to read Hypertape.

But I must confess: I didn’t do it alone. I didn’t even plan to write Hypertape; it sort of happened. It’s not that I mind the fame. It’s kind of nice getting fan letters and acknowledgements in other people’s programs. And I have no objection to jubilant nymphae strewing rose petals in my path, either, although I haven’t had too many of those yet.

Now, it’s time to own up. I wasn’t a man with a vision struggling against innumerable setbacks. I fell into it on my way to something else. It’s like the story of Thomas Edison picking up the world’s first light bulb, admiring it, and then bringing it to his lips and hollering, “Hello? Hello?”

It all started last fall, when I was having lunch with Julien Dube, a friend and fellow KIM owner, and Rick Simpson, then manager of KIM-1 Product Support for MOS Technology, Inc. Rick was talking about the cassette interface. “Maybe we should have made it faster,” he mused. “It could be speeded up by a factor of three, but...” At that moment the chopped chicken livers arrived, and the sentence was never finished. But the phrase had caught my imagination. A speedup of three times! Wow! But how would it be done? To solve the mystery, I would have to look into the workings of the KIM cassette load/dump programs.

Recording Basics

The KIM User Manual describes the cassette recording principles quite clearly. The system uses frequency shift keying (FSK). The two tones used are at frequencies of 3700 Hertz and 2400 Hertz respectively. During a dump to cassette, the tones are generated directly from the microprocessor as square waves — no oscillators are involved. In reading back from cassette, the signal is fed to an LM565 phase lock loop used as frequency discriminator (see Fig. 1). Everything else is done in software — timing, assembling of characters, storage of data and checksum. Handy to know, but not enough.

The next step was to dig into the software. How are the bits represented on tape? Still not hard to find; KIM is well documented. The so-called 2/1 scheme is used: To record a logic zero, send 3700 Hertz for five milliseconds duration followed by 2400 Hz for 2.5 ms. To record a logic one, send 3700 Hz for 2.5 ms followed by 2400 Hz for 5 ms (see Fig. 2). Either way, it’s about 7.5 ms per bit, right? And each sequence commences with the higher frequency.

Now we’re getting somewhere. The next step is to look at the tape load monitor program and see how it gets those bits back off the tape. Aha! Here’s what KIM does: It compares the timing of the two parts, 3700 Hz versus 2400 Hz. If the 3700 Hz signal lasts longer, the bit must be zero; if the 2400 Hz is the long one, then the bit is logic one.

Now, pay attention, we’re almost there. If the KIM loader doesn’t care about the actual timing, but just wants to know which frequency lasts longer... we can speed the whole thing up! As long as we keep the right timing ratio between the two frequencies, the KIM monitor won’t worry whether it’s fast or slow. Since we’re dealing with input and output at the bit level, we don’t need to...

---

Fig. 1. KIM-1 audio tape input circuit.

Fig. 2. Timing of normal KIM tape signals.
cycles direct from software. So all you’d need to do is to change the loop counters and...hmm, it might work. Of course, Rick’s phrase was, “...factor of three, but...” But what? Would the phase lock loop be too sluggish to take the speed increase?

Did all these exciting discoveries send me rushing to the coding sheets to see if I could produce triple-speed tape? Nope. Next day, I was chatting to Julien Dubé again.

“Funny thing,” I said in my wise and knowing way. “I think there’s a way to increase tape speed by at least 50 percent — it might even go up to triple speed.” As usual, Julien made a good audience as I outlined my detective work of the previous evening.

“What’s more,” I concluded triumphantly, “you’d hardly need to write it. Just copy out the ROM programs to RAM, change the pulse counters, and you have it!”

I thought no more of the conversation until late that evening when Julien called me. “Works fine,” he reported. “Good at triple speed, too. Why did you suspect there might be a problem?” I found this disconcerting. Not only had Julien been listening to me earlier in the day, he’d gone right ahead and done it...I collected my thoughts.

“OK,” I said. “The problem is that the phase lock loop is likely coming to the limit of its tracking capability. Put a meter on the output of the LM311 comparator. Normally it’s 2.5 V, but at high speed it will start to pull because of bias distortion.”

Julien called back very quickly. “It’s solid on 2.5 V...worry about details of tape formats: special characters, checksums and items like that. They will all eventually be sent as bits — and it’s the format of the bits we’re dealing with. To telescope those bits signals, we must return to the audio dump program. So long as we write them properly onto tape, we know that the load program will track them correctly.

Counting Cycles

A little arithmetic, or failing that, a look at the KIM manual, shows that logic zero consists of 18 pulses at the higher frequency followed by six pulses at the lower. For logic one the numbers become nine and 12 (see Fig. 3). As I noted these numbers, the words kept echoing through my head, “...factor of three...” Suddenly the penny dropped. All the above pulses are multiples of three — so you can reduce the number of cycles to 2/3 or to 1/3 without getting into fractions. KIM sends all its
at the highest speed," he reported. "Loads without error, too. Funny thing — I've listened to the tape itself and it sounds totally different than ordinary KIM tape."

On to the Coding Sheets

I thought it was that last comment that got me. How can it sound different when it's the same two frequencies? Besides, the phase lock loop behavior intrigued me: How could it track on only two frequencies? How much further could it go? At this stage, triple speed, we were sending a minimum of only two pulses at one frequency and three at the other. Could I speed that up without getting into fractions? I couldn't see how. Can you send half a pulse? That sounds like the paradoxe of one hand clapping. I was unable to see it.

Let's pick over the original KIM monitor coding for sending the two frequencies. It's shown in Program A. Well written, but since we need to change it anyway, let's see what we can do with it. Subroutines ONE and ZERO are almost identical. They differ in only two items: nine cycles versus six cycles, and 126 microseconds of delay in the timer versus 195 microseconds. Automatic programming reflex number one: Consolidate them and put the two variables in a table.

Assuming we have that squared away (see Program B), there's another piece of duplicate coding: The sequences at ONE1 and ONE2 (and their counterparts in ZRO) are almost the same. This time, the difference is in hexadecimal 27 versus A7. These values are sent to the output register to make bit 7 (the tape output) go on and off, generating the square wave that we record on tape. Automatic programming reflex number two: when you have a bit going back and forth like that, use an EOR (Exclusive OR) instruction to flip it over and back.

That last part is more than just efficient coding; it has important consequences for us to follow through. Previously, we generated a square wave by having a piece of program to turn the bit on, followed by a piece of program to turn it off again. That makes one full cycle of the square wave. But if we go the EOR route (see Program C), we'll flip the bit over and generate one-half of the square wave. That's what we've been looking for: a way to generate half a pulse. We've opened the door to sixfold speedup.

Now all the pieces have come together, and the coding comes easily. We have the number of half-cycles for each frequency in a table, so we can easily adjust the program for other speeds. At maximum — Hypertape — speeds, we'll be sending as little as one pulse at the lower frequency and 1.5 pulses at the higher frequency (Fig. 4). Can the phase lock loop track it? You bet it can — and the 2.5 V test point stays steady as a rock.

Wrapping It Up

The test runs were a bit eerie. Even when you do the arithmetic, it doesn't seem right for a 30-second program to load in five seconds. At first, it all happened so quickly that I was sure there was something wrong. But it checked out OK, and Hypertape became a reality.

Tests of various tape recorders revealed that a few of them won't carry Hypertape, apparently because their frequency response is too poor to carry the high sidebands of the signal. A related problem occurs in exchanging tapes from one cassette unit to another: Slight head misalignment causes those vital high frequencies to be lost. It's a good practice for KIM tape swappers to drop their speed to a paltry three times normal to eliminate this potential problem. Of course, the documented and tidied up program (Program D) was

```
0100 A9 AD DUMP
0101 8D EC 1F
0102 15 20 32 19
0104 08 A9 27
0105 0A 85 0F
0106 0C A9 BF
0107 0E 8D 43 17
0111 02 64
0113 0A 16
0115 20 61 01
0118 A9 2A
011A 20 88 01
011C AD F9 17
011E 10 20 70 01
0122 1D F5 A7
0124 0D 01
0120 AD F6 17
011C 20 6D 01
0122 F0 EC 17
0132 20 6D 01
0135 20 EA 19
0138 AD ED 17
013B CD F7 17
013E AD EE 17
0140 EF F8 17
0144 80 E9
0146 A9 2F
0148 20 88 01
014A AD E7 17
014E 10 20 70 01
0151 AD E8 17
0154 20 01
0157 A2 02
0159 A9 04
015B 20 61 01
015E 4C 5C 18

subroutines

0161 B6 F1
0163 48
0164 20 88 01
0166 07 68
0168 0E C6 F1
016A D0 F7
016C 60
016D 20 4C 19
0170 48
0171 4A
0172 A4
0172 4A

Program D. The final, polished, complete version for reading and writing data in the Hypertape format.
```
fired off to the KIM User
Notes for more extensive
field testing. Acknowledgement
was given to Julian
Dube for his help.

As you can see, Hyper-
tape's speed came from
putting the bits more com-
 pactly onto tape. There are
still other areas where the
signal can be made more
efficient. For example, each
byte of storage is translated
into two hexadecimal char-
acters. That's a waste of two-
to-one, since 16 bits are used
to store eight. Then there's
the question of the 2/1
coding scheme; that uses
three bit-times to store each
bit. And of course, we
haven't touched on the
question of data compression.
There are still worlds to
conquer. But I think I'll take
it easy for a while. After all,
there's something to be said
for full compatibility with
the KIM monitor. Then again,
if Julian isn't doing anything
next month....

---

The Realistic Controls
FORTRAN IV Minifloppy TM
Kit gives you the power of
FORTRAN and the speed
and convenience of the
minifloppy drive —
all for $1095.

* Distributed in the United
 States under license from
 Unified Technologies, Inc.,
of Islington, Ontario.

---

FLOPPY DISK SYSTEM FOR YOUR
S-II00 BUS COMPUTER

Our kit includes:
• One Shugart SA400 minifloppyTM
  DRIVE (assembled and
tested) — Second drive is optional.

• Fully socketed interface module kit featuring processor-
  independent timing, 7 level vectored interrupts, bootstrap
  and diagnostic PROM, parallel 8-bit I/O ports.

• Cables, cabinet and regulator parts.

• System minidiskette™ with FORTRAN IV (featuring IBM
  compatible floating point, subroutine library, 8080 extensions);
  Livermore BASIC; Disk Operating System; Text Editor;
  Utilities.

The system requires one standard S-100 bus slot, power from
your system supply, and 24K of RAM.

Z/25 FORTRAN IV-Minifloppy Kit .... $1095
(Assembly & Tested) .... $1220
Second Minifloppy & Expansion
Parts .... $ 449
(Assembly & Tested) .... $ 495
Formatted Minidiskettes
(65K capacity) .... $ 25

Write for detailed brochure; or
order sending check, money
order, BA, or MC card No. with
expiration date and signature.
PO if D&B rated.
Signature of software non-
disclosure agreement is required.
Freight added to all non-prepaid
orders; Iowa and Minnesota
residents add sales tax.
UPS COD shipments accepted
upon 25% down payment.

---

CALL DIANNA AT:
realistic controls corporation
404 WEST 35TH STREET
DAVENPORT, IOWA 52806
(319) 396-4400

---

---