Challenge Sprint 4x tape recorder

(/forums/profile/1765/mark_k)

mark_k (/forums/profile/1765/mark_k)

July 2011 (https://www.worldofspectrum.org/forums/discussion/35066/challenge-sprint-4x-tape-recorder) edited August 2011 in Hardware (https://www.worldofspectrum.org/forums/categories/hardware)

Hi,

I noticed a Challenge Sprint (http://www.worldofspectrum.org/infoseekid.cgi?id=1000051) on eBay, and checked out the WoS entry for it. It's a tape recorder which plugs into the user port, and saves/loads at 4 times normal speed. Apparently it included a replacement ROM with modified load/save routines. (The tape format is identical to the normal one, but of course load/save timing needs to be changed to run at the higher speed.)

Does anyone know whether the replacement ROM was in the Challenge Sprint unit, or did it have to be installed in the Spectrum?

It turns out there was a British patent application relating to the Challenge Sprint, if anyone's interested to read about how it works. You can view that by searching for GB2164527A or "High speed cassette tape player" at http://www.espacenet.com (http://www.espacenet.com). For convenience I've also uploaded a PDF file: **File name: GB2164527A.pdf File size: 416.68 KB** (http://www.fileserve.com/file/U4ecNCG).

Edit: From reading the patent application, it sounds like there is a ROM in the unit, which is paged in when addresses corresponding to the load & save routines in the Spectrum's ROM are accessed. I wonder whether it would be possible to dump the ROM inside the recorder by simply saving the normal 16KB ROM region to tape?

Post edited by mark_k on August 2011

(/forums/profile/72/LCD)

LCD (/forums/profile/72/LCD)

July 2011 (/forums/discussion/comment/549603/#Comment_549603) edited July 2011

> mark_k (/forums/profile/1765/mark_k) wrote: » (/forums/discussion/comment/548749#Comment_548749)
> Hi,
>
> I noticed a Challenge Sprint (http://www.worldofspectrum.org/infoseekid.cgi?id=1000051) on eBay, and checked out the WoS entry for it. It's a tape recorder which plugs into the user port, and saves/loads at 4 times normal speed. Apparently it included a replacement ROM with modified load/save routines. (The tape format is identical to the normal one, but of course load/save timing needs to be changed to run at the higher speed.)
>
> Does anyone know whether the replacement ROM was in the Challenge Sprint unit, or did it have to be installed in the Spectrum?
>
> It turns out there was a British patent application relating to the Challenge Sprint, if anyone's interested to read about how it works. You can view that by searching for GB2164527A or "High speed cassette tape player" at http://www.espacenet.com (http://www.espacenet.com). For convenience I've also uploaded a PDF file: **File name: GB2164527A.pdf File size: 416.68 KB** (http://www.fileserve.com/file/U4ecNCG).
>
> Edit: From reading the patent application, it sounds like there is a ROM in the unit, which is paged in when addresses corresponding to the load & save routines in the Spectrum's ROM are accessed. I wonder whether it would be possible to dump the ROM inside the recorder by simply saving the normal 16KB ROM region to tape?

Yes, this would be possible. I had some of these babys. It is not paged in when addresses corresponding to the load & save routines in the Spectrum's ROM are accessed. The switch can switch between original ROM and modified ROM, so this is a hardware solution. If you want to use Sprint, switch the ROM in it on, or if you want to use normal tape deck or disc interface, switch it off. Very simple!

(/forums/profile/1765/mark_k)
mark_k (/forums/profile/1765/mark_k)
July 2011 (/forums/discussion/comment/549742/#Comment_549742) edited July 2011

> LCD (/forums/profile/72/LCD) wrote: » (/forums/discussion/comment/549603#Comment_549603)
> Yes, this would be possible. I had some of these babys. It is not paged in when addresses corresponding to the load & save routines in the Spectrum's ROM are accessed. The switch can switch between original ROM and modified ROM, so this is a hardware solution. If you want to use Sprint, switch the ROM in it on, or if you want to use normal tape deck or disc interface, switch it off. Very simple!

The patent does specifically mention the ROM in the unit being paged in when certain addresses are accessed. (The switch would just disable the ROM paging.)

There would be a couple of advantages to doing it that way, rather than having a complete replacement ROM. First, no copyright infringement. Second, the amount of ROM required would be much less than 16KB, so cheaper to manufacture.

It would be interesting to see the circuit board inside the Sprint to see how it is actually implemented.

(/forums/profile/72/LCD)
LCD (/forums/profile/72/LCD)
July 2011 (/forums/discussion/comment/549749/#Comment_549749) edited July 2011

It would not be ceaper because I think, the decoder logic for translating adresses to EPROM would cost more than a larger EPROM.
I have no SPRINT anymore to check out how it was done

(/forums/profile/329/Fraser)
Fraser (/forums/profile/329/Fraser)
July 2011 (/forums/discussion/comment/549813/#Comment_549813) edited July 2011

I was wondering if it uses the standard hardware port or a different one?

(/forums/profile/4509/mcleod_ideafix)
mcleod_ideafix (/forums/profile/4509/mcleod_ideafix)
August 2011 (/forums/discussion/comment/554708/#Comment_554708) edited August 2011

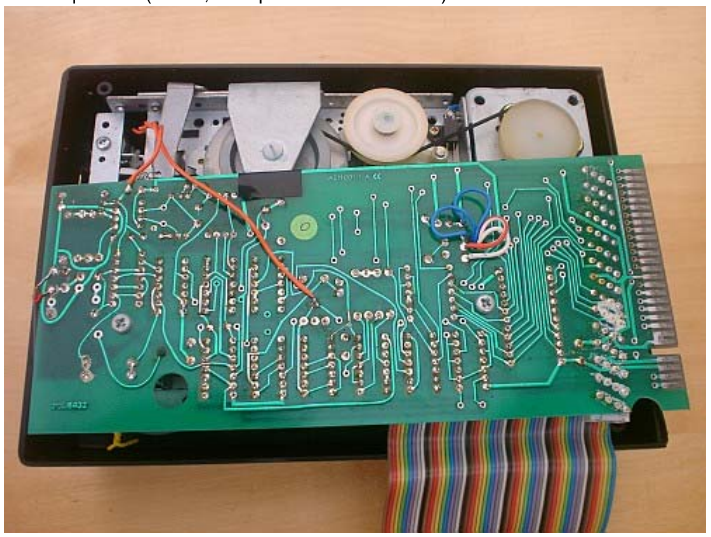
I think I'm the one who bought that Sprint you saw on eBay.

There comes a little review:

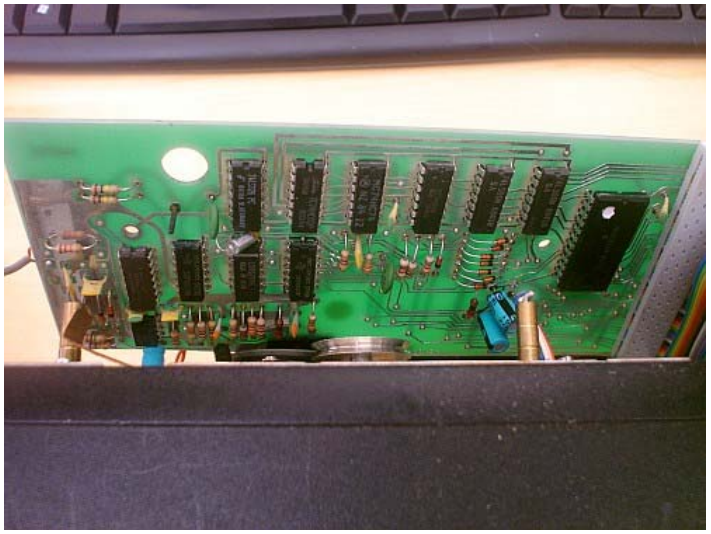
The unit has no power plug. It takes power supply from the rear bus. Haven't checked if it uses 5V or 9V, or both. It surely takes 5V as the logic circuits (we'll see them in a moment) need that, and no regulator has been found. It hasn't got any volume knobs either. The rainbow ribbon cable is just long enough to bend it forming a 90 degree angle to plug it into the rear bus and leave the cassette unit just at the right of the computer.

The Sprint continues the rear bus, so joystick interfaces and the like are possible. The unit has the "disable switch" (a 5 GBP add-on at the time) that let the user to disable the Sprint ROM without having to turn the computer off, remove the ribbon cable, then turn on.

The control PCB is quite large, but not dense populated. The two yellow wires solded to a point in the center of the PCB and to a point near the top left corner go to the disable switch. When the switch is in the ON position (that is, both points are connected) the interface is disabled.



A componet view (sort of) of the above PCB. The large chip is the ROM, a N82S141N, which is a 512 byte PROM. The only analog chip I've seen in the PCB is a TL082 (dual op-amp). There is a CD4007, which I think sits between the analog and the digital part, as it has three CMOS inverters that can be externally wired to obtain NOT gates, NOR gates, NAND gates, etc, so it may be needed to square shape the signal amplified in the TL082.

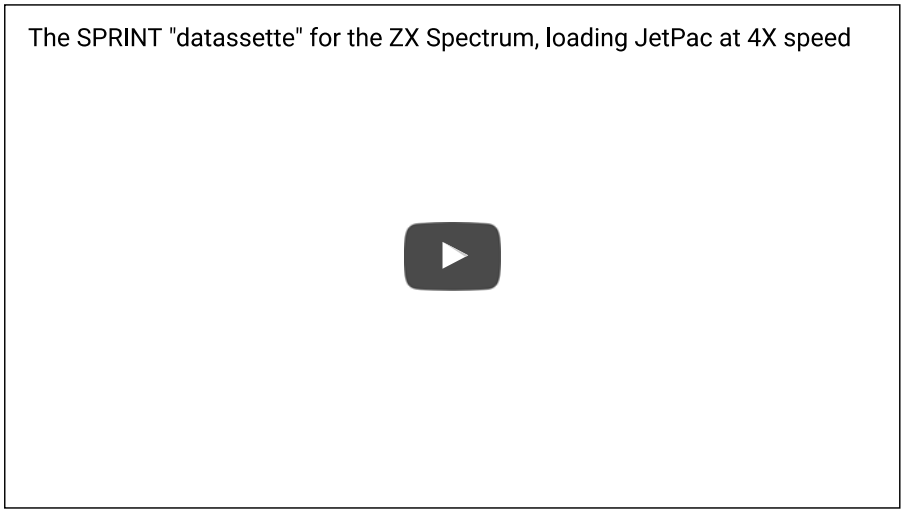


I've performed a load test with some tapes I have around. Some loaded succesfully, some others not. Reviews I've seen at WOS mention that this unit has not azimut adjustment. Mine has, but it's not of very use, because you cannot hear any loading noises through the computer speaker. The Sprint has no built-in speaker (although I think it's very easy to add one). The only moment in which you hear any sound is while the computer is actually loading something, or when the pilot tone is playing.

I wanted to perform a test which showed that a normal speed program can be loaded with this device at a very high speed. As I didn't trust my old tapes, I decided to grab an almost-new TDK cassette and record into it a digitally generated audio copy of Jet Pac using a standard cassette player (a SONY unit).

I chose Jet Pac because:
- It's a 16K game, so won't take "ages" to load, even at normal speed.
- It was designed for a 16K RAM Spectrum, so it loads entirely into contended memory, which might cause a little harm to the loading routine, as stack-related instructions and the instruction that stores the last listened value into memory will have to deal with the ULA timming restrictions. As you will see, there's very few affected instructions in the loading loop.
- It shows its loading screen as it loads, so you'll see how fast the loading is.
- Was the first game I loaded into my 48K Spectrum, back in 1984

Here you have it:



On the other hand, I wanted to have a copy of the shadow ROM, which is very easy with this device. I only had to issue a SAVE "rom" CODE 0,16384. The Sprint can save, in addition to load, at high speed. The saved block was loaded again at address 32768.

When I listened to the computer noise while saving, I noticed a pattern repeating all the time, as if the same block of bytes were repeating again and again. It was clear that the ROM decoding was partial, so the same 512 byte block was present 32 times, to fill the entire 16K space.

So I checked this by PEEKing at some locations starting 32768. Then, with the Sprint disabled, I saved the 512 byte portion of interest at normal speed, while Spectaculator was loading it using the "Load from Audio Source" option. Once in the emulator, the "Export" option was used to save a copy of the ROM into a .BIN file, which you have here:
Sprint ROM (http://www.zxprojects.com/images/stories/sprint/rom_sprint.bin)

Some of you will wonder if I simply could have saved the ROM file at "Sprint speed" and load it directly into Spectaculator by having the CPU boosted at 14MHz. I tried that but didn't work.

This 512-byte ROM, as I said, is present beginning at different addresses: 0000h, 0200h, 0400h, ... That is, it begins at any address with the format XN00h, where X can be 0,1,2 or 3, and N is even.

Here you have the disassembly of the Sprint ROM (http://www.zxprojects.com/images/stories/sprint/rom_sprint.html). To ease the disassembly process, I have assumed that the 512 byte block is present at address 0400h to 05FFh. This is because the starting points of the original SAVE and LOAD routines are at 04C2h and 0556h respectively, so they fall entirely into this 512-byte block (as expected).

There's a JP 4 instruction right at the beginning of the ROM (after an EI instruction). As this block is also at 0000h, the JP 4 instruction merely jumps to the next instruction, which OUT's a 0 into port BFh. I think this unpages the Sprint ROM, and the next instruction executed, already from the main ROM, begins at 0008h, the ERROR restart.

By the way: this ROM (and thus, the device) uses these ports:
* BFh. Write-only. The ROM only writes 00h here. I think it's for disabling the Sprint ROM
* 7Fh. Write-only. The ROM writes 00h or 80h here. It's the new "MIC" port, bit 7.
* FFh. Read-only. "EAR" port, bit 7. Decoded bit value, bit 0 (see below)

(/forums/profile/4509/mcleod_ideafix)mcleod_ideafix (/forums/profile/4509/mcleod_ideafix)
August 2011 (/forums/discussion/comment/554858/#Comment_554858) edited August 2011

> mcleod_ideafix (/forums/profile/4509/mcleod_ideafix) wrote: » (/forums/discussion/comment/554708#Comment_554708)
> Some of you will wonder if I simply could have saved the ROM file at "Sprint speed" and load it directly into Spectaculator by having the CPU boosted at 14MHz. I tried that but didn't work.

I've realized why this didn't work. It happens that the loading noise I hear during a SAVE or LOAD operations is not actually the sound the computer listen to.

A more carefull analysis of the source code I posted shows that the SAVE process sends bits thru port 7Fh and port FEh. The former is the actual bit recorded onto tape. The later is merely a loading-like noise, which follows the actual data, but not all bits, every other bit.

I mean: the noise I hear is twice the frequency of a standard load scheme, not 4 times the frequency, as I was expected. Besides, the bits issued to MIC and EAR are half the bits recorded onto tape, so trying to use these sounds directly from the SPRINT to a "boosted Spectrum" (i.e. an emulator with a emulated CPU running at 7MHz and an option for loading from the sound card) will fail.

I'm in the process of adapting the load and save routines from the SPRINT rom to use the standard EAR/MIC port. For SAVE I think I've got it. Using the "audio record to WAV" option present in Spectaculator, I've succesfully recorded an audio signal with the actual content that the SPRINT issues to the tape.

It's because of this that I have realized what was going on with the loading noise. In fact, I first titled my video as "2X speed" because the sound and the width of the border stripes suggested that.

I know that trying to load such a high pitch tone will fail in a real Spectrum, as the analog circuitry next to the EAR socket will block the higher frequency portions of the audio signal (the accelerated loading uses frequencies up to 8kHz) so I'm trying first with emulators. Not with many success by now...

(/forums/profile/5240/fog)
fog (/forums/profile/5240/fog)
August 2011 (/forums/discussion/comment/554860/#Comment_554860) edited August 2011

i found it interesting about the bleeps / pause on jetpac.. how it wasn't tripped up by that when it should be perhaps.

that thing has a thing of stretching tapes..so better to use backups , I was reading up about it at one point..

(/forums/profile/4509/mcleod_ideafix)
mcleod_ideafix (/forums/profile/4509/mcleod_ideafix)
August 2011 (/forums/discussion/comment/554865/#Comment_554865) edited August 2011

> mcleod_ideafix (/forums/profile/4509/mcleod_ideafix) wrote: » (/forums/discussion/comment/554858#Comment_554858)
> ...so I'm trying first with emulators. Not with many success by now...

And I know now that I won't success at all. The patent document that Mark has provided is very clear. The clue is at apge 3. The thing actually decodes FSK!! so the value at bit 7 of port FFh is not the current state of the "EAR" input, as in normal loaders, but the actual bit of the current loading byte.

So, as the patent states, there no need for tight loops, as the time measuring is performed by a monostable, which is reset to 0 on each positive edge of the incoming signal. The computer has to keep reading the monostable value while it waits until the next positive edge. A polarity correction circuit ensures that the edges are right regardless of the polarity the tape were recorded with.

The monostable is configured so it turns to '1' after a specific period of time. Time that is roughly 3/4 times the period of a '1' bit in FSK (remember that the '1' bit lasts double the time than a '0' bit). So if a '1' is currently playing, the monostable will switch to '1' and that will be the value read by the computer, but if a '0' is playing, the next positive edge will happen long before the monostable switchs to '1', hence a '0' will be read.

Reviewing the source code, I think that port FFh offers two things: the current state of the incoming signal, at bit 7, needed to track the pilot tone at the first part of the loading routine, and to detect edges in

the second part. The current value of the monostable, that is, the bit after the FSK decoding process, seems to be at bit 0. The routine reads port FFh, stores bit 0 into the carry with a RRA instruction, some instructions after that, the routine retrieves the bit again into H using the instruction RL H.

```
ROM:0492 loc_492:
ROM:0492                dec     l
ROM:0493                jr      z, loc_4A8
ROM:0495                in      a, (c)
ROM:0497                jp      m, loc_492
;loops while the pulse is high, so it exits
;just after a positive to negative edge has ocurred

ROM:049A
ROM:049A loc_49A:
ROM:049A                dec     l
ROM:049B                jr      z, loc_4A8
ROM:049D                rra
ROM:049E                in      a, (c)
ROM:04A0                jp      p, loc_49A
;loops while the pulse is low, so it exists just after
;a negative to positive edge has ocurred. The carry
;bit holds the value of bit 0 read in the previous IN
;operation, as at the precise moment a falling edge
;happens, the monostable is reset to 0.

ROM:04A3                rl      h  ;load the bit into the H register.
ROM:04A5                jr      nc, loc_482
```

This explains why I have seen no tight loops, but some NOP's inside the saving and loading loop. The computer uses timming loops to detect the pilot tone, but the monostable for actual byte loading. That explains why I've able to see the border stripes while playing the pilot tone, but failed to actually load anything when I've tried to use a version of the loading routine changing the FFh port by the FEh port.

I've found it very simple and clever.

(/forums/profile/4509/mcleod_ideafix)
mcleod_ideafix (/forums/profile/4509/mcleod_ideafix)
August 2011 (/forums/discussion/comment/554868/#Comment_554868) edited August 2011

Finally, I'd like to give some details of what I think it's the very heart of the loading routine, and the code that shows all the magic that the SPRINT cassette offers:

What this tape player implements is no more and no less than a converter from an asynchronous FSK coded signal to a synchronous 1-bit serial line. The DATA bit is the monostable bit (bit 0 of port FFh) and the CLOCK bit is what we have previously called the "signal" bit (bit 7 of port FFh, which gives us the actual pulses present into the tape). As we stated, the conversion is performed in hardware, and DATA is valid just before a negative to positive transition at CLOCK happens. The byte loading routine that follows, just have to wait for this situation, taking into account that the signal flow might be interrupted at any time, so timeouts have to be provided to not to hang the computer into an endless loop because of an interrupted operation.

This is it:

```
;Registers used:
;C = 0FFh (for the IN instruction)
;BC' = 1601h. C is xored with B at each loop. The result is
;outputted to FEh, so these two values provides visual
;and audio feedback of the loading process to the user.
;H = holds the byte that is being read from tape. First bit
;read is MSb.
;L = timeout value for waiting an edge.

;On "normal" exit: H = byte loaded from tape. Carry set.

ROM:0480 LoadOneByte:
ROM:0480                ld      h, 1  ;Mark bit 0 with 1. When H is filled
                                      ;this '1' goes to the carry bit,
                                      ;signaling that the byte is completed.
ROM:0482 NextBit:
;---------------------------------------------------------------
; BREAK CHECKING
ROM:0482                ld      a, 7Fh
ROM:0484                in      a, (0FEh)  ;read SPACE.
ROM:0486                rra
ROM:0487                jr      nc, TotalExit  ;if pressed, early exit.
;---------------------------------------------------------------
; BORDER AND SPEAKER HANDLING
ROM:0489                exx
ROM:048A                ld      a, c
ROM:048B                xor     b
ROM:048C                ld      c, a
ROM:048D                out     (0FEh), a
ROM:048F                exx

ROM:0490                ld      l, 1Eh  ;timeout for waiting for an edge.
;---------------------------------------------------------------
; LOOP FOR WAITING A POSITIVE TO NEGATIVE EDGE.
ROM:0492 WaitFor0:
ROM:0492                dec     l  ;update timeout value
ROM:0493                jr      z, TotalExit  ;if timeout, early exit.
ROM:0495                in      a, (c)  ;reads clock and monostable
ROM:0497                jp      m, WaitFor0  ;loops while clock is '1'
;---------------------------------------------------------------
; LOOP FOR WAITING A NEGATIVE TO POSITIVE EDGE.
ROM:049A WaitFor1:
ROM:049A                dec     l
ROM:049B                jr      z, TotalExit
ROM:049D                rra   ;stores last monostable value read into carry.
ROM:049E                in      a, (c)  ;reads clock and monostable
ROM:04A0                jp      p, WaitFor1  ;loops while clock is '0'
;---------------------------------------------------------------

ROM:04A3                rl      h  ;load monostable value into H
ROM:04A5                jr      nc, NextBit  ;if H is not full, go
                                             ;for the next bit.
ROM:04A7                ret

ROM:0535 TotalExit:
ROM:0535                pop     hl  ;discard return value for this routine
ROM:0536                xor     h  ;clears carry?
```

```
ROM:0537                ret  ;return to the caller of the main load routine.
```

kgmcneil (/forums/profile/4534/kgmcneil)
August 2011 (/forums/discussion/comment/554890/#Comment_554890) edited August 2011

This is facinating.. Id never heard of this system at the time, but am curious about your work there mcleod_ideafix.. I wonder whether your dissassembled code there could be implemented easily as a kind of speedload header before loading the main bit of code?... Could a game tape, for instance, have the first bit of code at the normal frequancy, load in this header speedloader, then have the rest of its data playing at this jacked up frequancy?... Is recording this increased frequancy sound on a tape playing at normal speed (ie: using a standard tape recorder) viable?...

mcleod_ideafix (/forums/profile/4509/mcleod_ideafix)
August 2011 (/forums/discussion/comment/554925/#Comment_554925) edited August 2011

> kgmcneil (/forums/profile/4534/kgmcneil) wrote: » (/forums/discussion/comment/554890#Comment_554890)
> could be implemented easily as a kind of speedload header before loading the main bit of code?... Could a game tape, for instance, have the first bit of code at the normal frequancy, load in this header speedloader, then have the rest of its data playing at this jacked up frequancy?... Is recording this increased frequancy sound on a tape playing at normal speed (ie: using a standard tape recorder) viable?...

Not al all. First, the high frequencies used are not suitable for a tape playing at normal speed (4.76 cm/s). Besides, loading at such speeds is an unreliable process because the CPU is too slow to measure timming loops accurately. It's for these two reasons that the tape is playing at 4X speed and there is hardware in the device that performs most of the work that is needed to decode the datastream.

For high speed loading routines suitable for an unexpanded ZX Spectrum, I'd use the k7zx (Otla) project. It uses routines that even surpases the speed achieved with the SPRINT, but I'm not very sure about its reliability when recorded onto a standard tape.

lfaria (/forums/profile/6353/lfaria)
August 2011 (/forums/discussion/comment/555839/#Comment_555839) edited August 2011

Fascinating device and analysis, congrats!

As I was reading through the patent application, the polarity detection / inversion is well described, but I can't find supporting code in the disassembly. This detection is suppose to be made at the second sync pulse and I was expecting a bit set / or reset permanently from there on until ld_bytes exits, but I can't see it.

The drawing shows the xor gate driven by something external to the schema.

Maybe it's not implemented (in the version you have analysed)?

mcleod_ideafix (/forums/profile/4509/mcleod_ideafix)
August 2011 (/forums/discussion/comment/556080/#Comment_556080) edited August 2011

> lfaria (/forums/profile/6353/lfaria) wrote: » (/forums/discussion/comment/555839#Comment_555839)
> The drawing shows the xor gate driven by something external to the schema.
>
> Maybe it's not implemented (in the version you have analysed)?

I'm pretty sure that the polarity correction circuit is 100% hardware, and transparent to the software. That is, the software will expect always a certain polarity, regardless of the actual signal recorded on tape. Note that both the "signal" bit and the "monostable" bit are calculated after the XOR gate, so the software will always see a fixed polarity signal.

Until a proper schematic be drawn, we won't know how polarity correction is handled. My guess is that the correction is performed when the hardware detects the sync pulses. May be by using some sort of monostables to measure the pulses as they are played, detecting which of them is the first sync pulse, and memorizing its level: if the level of the first sync pulse is '1', no correction has to be done, and the output of the polarity correction circuit is '0', so the XOR gate has no efect on the incoming signal. If the firsts pulse is memorizedd as '0', then the output is '1', causing the XOR gate to invert the incoming signal.

If the hardware works more or less this way, it means that the loader in software is polarity independent until (included) the sync pulses are processed.

Latest Posts (/forums/discussions)

My new Punk music radio show

mel the bell

on 4:48PM

What am I doing wrong?

hikoki

on 4:47PM

New Game: Rediscovered Realms

Paul-J

on 4:35PM

New spectrum game from RetroWorks: SWORD OF IANNA

F_Clowder

on 4:30PM

ROM emulation in software

Dan Dare

on 4:19PM

Visiting London and looking for retro shops, flea markets and retro events. Need help!

thricenightly

on 4:18PM

I am currently watching.....

mel the bell

on 4:12PM

z88dk and sdcc

thricenightly

on 4:11PM

Exploding Wall (missing artist)

F_Clowder

on 4:00PM

Related Discussions

WANTED - Tape Recorder