The MGT Disciple and Plus D PALS, finally unveiled V2

(Updated March 2024)

Background

One of the biggest mysteries of the Disciple and Plus D interfaces are the PAL2L08 ICs, what they do, how to get replacements, and how to get the fusemap (jedec file / equations) that they contain. I hope this page will answer all these questions, along with a thorough explanation of each logic equation that these nice ICs contain in their silicon make up.

2024 Update

Updated by original author to correct some typos (namely N1 and NREQ, should be M1 and MREQ) and to correct the interpretation of some equations (around Inhibit switch operation and RESET behaviour). Added a section on the Plus D describing its PAL logic.

Copyright

The copyright for all the equations in the PAL ICs is owned by Bruce Gordon, the original Disciple & Plus D designer. From what I understand from reading some posts on usenet, the original equations were on a disc that got lost, and remained unknown for some time. I also have seen some posts from Bob Brenchley indicating that he was in touch with Bruce, and if they were found, he would have no problem with the distribution.

As these ICs have a protection bit which stops you reading their fuse map, which is usually blown, finding out their contents and replacing them is difficult. Bruce, if you are out there, thanks for a GREAT interface, and if you want me to pull this page, I'll be more than happy to do so.

How did you figure it out then?

Fortunately, a nice Dutch man called Rudi Biesma placed his ICs in a high / low device which sends every logic combination to every pin and produces a table of the outputs. It then can produce a fuse map of the IC. With this lovely Jedec file from Rudy, I disassembled it and obtained the equations using PALASM, and then set about figuring out just what the heck was going on.

What is a PAL20L8?

A PAL IC is an older type of programmable logic IC. You can program it with Boolean equations to give whatever outputs you want under certain input conditions. PALs were later replaced by GALS (which can be programmed to act like a PAL) and now GALS have been largely replaced by CPLDs. The PAL20L8 ICs in the Disciple & Plus D have been out of production for many years now, and are increasingly hard to come by. Your best bet is to search for obsolete electronic component suppliers on the web. They are available, but can be pricey as minimum order quantities are required. The ones normally found in a Disciple are manufactured by national semiconductor, and are of the A variety (A is a speed rating A is the slowest). I have successfully replaced them with MMI PAL20L8B ICs which are slightly faster, and are pin for pin compatible. PAL20L8 ICs from other manufactures may be just as compatible, but it is well worth checking as these things are expensive (around £10 each, plus the min order, which is normally > £100). Rudi Biesma tried to replace his PAL20L8s with GALS but reported no luck.

Where can I get fully programmed PALs for the Disciple or Plus D?

Either buy the ICs yourself and get someone with a programmer to program them, or if you are really stuck I will supply both PAL ICs, fully programmed for £30 + postage.

The Disciple PAL ICs

The two PAL ICs (IC8 & IC9) in the Disciple do the following operations:

- Page out the Speccy ROM and enable the Disciple RAM / ROM when the Z80 accesses certain memory addresses (0x0001 (*), 0x0008, 0x0066, 0x28E) or an NMI is generated (snapshot button)
- Page out Disciple ROM & RAM when a hardware reset is performed
- Page out Speccy ROM, enable Disciple ROM & RAM during network requests
- Holding the Speccy ROM out and the Disciple ROM & RAM enabled after they have been brought in until they need to be paged out again
- Provide the IO decoding when a request is written to port 0x7B (reset / set boot ; control of DOS needs to reloaded)
- Enable the WD1772 disc controller (IC4) when an IO request is written or read at its' I/O ports (0x1B, 0x5B, 0x9B, 0xDB)
- Generate a clock pulse for IC5 flip flop when an IO write request is sent to the Disciple control port (0x1F)
- Output a clock pulse to the printer port IC11 when data is written to port (0xFB)
- Enable the joystick controller ICs (IC1 & IC10) when an IO request is read from the joystick ports (Kempston and Sinclair)
- Tell the spectrum to WAIT when data is being pushed out to the network
- Avoid paging out Speccy ROM or responding to IO requests when Inhibit switch is pressed, or when Inhibited via an OUT instruction to port #1F, bit 4 (Note Inhibit switch needs to be pressed for software control of it via this method to work – check the circuit diagram)

(*) Except during a hardware reset cycle

You will see these sweet little IC's actually control most of the Disciple, and nothing works without either one, as their logic equations are deeply intertwined between them.

The Disciple PAL Equations explained

Before you go wading in, please understand a few things about how Boolean logic symbols are represented in this document:

^ means active low (ie. ^RD means that if a read is happening, this line will be LOW)

/ means NOT

+ means OR

* means AND

^/ is seen when an active low control line is in the LOW state (i.e. active). Example: ^/MEMREQ means the MEMREQ line is LOW (active)

You will also have to have a rough understanding of how the good ole Z80 works (what lines are set when performing IO, memory fetches, read / writes etc) and a reasonable understanding of the Disciple itself. The latter can be obtained for the excellent Ramsoft "Disciple / +D Technical Guide", although please note it incorrectly states the Disciple and Plus D are paged in at address 0x0000 which is incorrect for both interfaces.

A short note about tristate. Some IC input / outputs can have 3 levels. These are logic zero, logic one and disconnected (i.e. just floating). The third state is very handy when you only want an IC to do stuff some of the time and pretend it's not there others. Just like the disciple ROM / RAM. While physically connected to the address / data bus, unless their Chip Enable (CE) pin is activated, their pins will be internally disconnected, so won't interfere unless requested to.

It is also recommended you look at the Disciple circuit diagram shown below to see how these ICs fit together and link to other Disciple components.

Be aware that outputs O17 and O18 of IC9 feed into inputs I1 and I2 of IC8 and output O22 from IC8 feed into input I14 of IC9, providing a logic link between all the equations – this makes it slightly harder to understand what is going on.

In the equations I've substituted wherever possible the names of the signals on the Spectrum bus the Inputs and Outputs are connected to. The references p8_O22, p9_O17 and p9_O18 refer to PAL IC8 Output 22, PAL IC9 Output 17 and PAL IC9 Output 18 respectively.

Onwards with these equations. Each PAL equation is listed, with my explanation below on the next pages.

CHIP DIS_ASM PAL20L8 (PAL IC8)

Pin Descriptions

```
PIN 1 p9_O18

PIN 2 p9_017

PIN 3 A10

PIN 4 A0

PIN 5 A4

PIN 6 ^RESET

PIN 7 ^WR

PIN 8 ^RD

PIN 9 ^MREQ

PIN 10 A12

PIN 11 A13

PIN 12 GND

PIN 13 A15

PIN 14 A14

PIN 15 O15 (/CE Disciple RAM)

PIN 16 O16 (/CE Disciple ROM)

PIN 17 O17 (/M1 & INHIBIT Switch)

PIN 18 O18 (NC – used as flip flop)

PIN 19 O19 (NC – used as flip flop)

PIN 20 ^ROMCS

PIN 21 O21 (NC)

PIN 22 O22 (PAL IC9, I12)

PIN 23 A11

PIN 24 VCC
```

Disciple IC8 EQUATIONS

```
O22.TRST = VCC

/O22 = /A4 * /^RD * ^MREQ

+ /A4 * /^WR * ^MREQ

+ /A10 * /A11 * /A8 * /A4 * /^RD * /^MREQ * /A12 * /A14 *

/A13 * /A15
```

Used as a partial equation to help form other equations in PAL IC9

```
O21.TRST = VCC

/O21 = ^ROMCS

+ /p9_O18* /^RD * O17
```

Used as a partial equation to O15, O16 and O20 (ROMCS). Condition met when Spectrum ROM paged out.

When system is NOT inhibited – it will respond to an IN IOREQ to port 0xBB (page IN Disciple) and memory reads from 0x0001 (except during a hardware reset), 0x008 (RST8), 0x0066 (NMI code), 0x028E.

```
^ROMCS.TRST = VCC

/^ROMCS = /O17

+ O21

+ /^RESET

+ /p9_O18* /^WR
```

Pull speccy rom IN when Disciple Inhibited (/O17); hardware RESET or an OUT IOREQ to port 0xBB (page OUT Disciple) and ignore memory writes to 0x0001, 0x008 (RST8), 0x0066 (NMI code), 0x028E

```
O19.TRST = VCC

/O19 = O18

+ /^RESET

+ /p9_O17* /^RD
```

Partial equation to O15 and O18 below. Condition met when a reset happens or an IN to port 0x7B (set/unset boot flipflop), system reset or O18 below.

```
O18.TRST = VCC

/O18 = O19

+ /p9_O17* /^WR
```

Condition met when an OUT 0x7B (set boot flipflop) or O19 (IN 0x7B, reset boot flip flop). Used as a partial to O16 and O15 below.

```
O16.TRST = VCC

/O16 = ^ROMCS * O18 * /^RD * /^MREQ * /A14 * /A13 * /A15

+ ^ROMCS * O19 * /^RD * /^MREQ * /A14 * A13 * /A15
```

O16 is the Disciple ROM CE line, so this equation controls when the Disciple ROM should be paged in. Met when the Speccy ROM is OUT and we are reading from address < 8192 and the RAM flip flop is unset (Disciple ROM at 0x0000 – normal operation) OR if the address is < 16384 (Disciple ROM at 0x2000) and the RAM flip flop is not set.

```
O15.TRST = VCC

/O15 = ^ROMCS * O19 * /^RD * /^MREQ * /A14 * /A13 * /A15

+ ^ROMCS * O19 * /^WR * /^MREQ * /A14 * /A13 * /A15

+ ^ROMCS * O18 * /^RD * /^MREQ * /A14 * A13 * /A15

+ ^ROMCS * /^WR * O18 * /^MREQ * /A14 * A13 * /A15
```

O15 is the Disciple RAM CE line, so this equation controls when the Disciple RAM should be paged in. Met when the speccy ROM is OUT and we are reading / writing from address < 8192 (RAM flip flop set) or reading / writing from address < 16384 (RAM flip flop not set).

Phew IC8 done. It was the hardest. Glad that’s over. Ready for IC9?

CHIP DIS_ASM PAL20L8 (Disciple IC9)

Pin Descriptions

```
PIN 1 A9

PIN 2 A5

PIN 3 A6

PIN 4 A7

PIN 5 ^WR

PIN 6 ^RD

PIN 7 ^IORQ

PIN 8 A3

PIN 9 A2

PIN 10 A1

PIN 11 A0

PIN 12 GND

PIN 13 NET

PIN 14 p8_O22

PIN 15 O15 (CLK Disciple IC5 flip flop, control port #1F)

PIN 16 O16 (/CS Disciple IC4 WD1772)

PIN 17 O17 (I2 PAL IC8)

PIN 18 O18 (I1 PAL IC8)

PIN 19 O19 (/CE Disciple IC1a joystick1)

PIN 20 O20 (/CE Disciple IC10a joystick2)

PIN 21 O21 (/WAIT)

PIN 22 O22 (CLK Disciple IC11)

PIN 23 ^M1

PIN 24 VCC
```

EQUATIONS

```
O22.TRST = VCC

/O22 = A5* A6 * ^M1 * A7 * /^WR * /^IORQ * A3 * /A2 *

A1 * p8_O22 * A0
```

Generate a clock pulse for the printer IC11 when an IOREQ write is sent to printer port (OUT 0xFB).

```
O21.TRST = VCC

/O21 = A5* /A6 * ^M1 * /A7 * /^WR * /^IORQ * A3 * /A2 *

A1 * p8_O22 * A0 * NET
```

Tell the speccy to wait while a network request is written to the network port (OUT 0x3B).

```
O20.TRST = VCC

/O20 = /A5* /A6 * ^M1 * /A7 * /^RD * /^IORQ * A3 * A2 *

A1 * p8_O22 * A0
```

IOREQ read of joy1 port (0x1F). Set CE on IC10

```
O19.TRST = VCC

/O19 = A5* A6 * ^M1 * A7 * /^RD * /^IORQ * A3 * A2 *

A1 * p8_O22 * /A0
```

IOREQ read of joy2 port (0xFE). Set CE on IC1

```
O18.TRST = VCC


/O18 = A5* /A6 * ^M1 * A7 * /^IORQ * A3 * /A2 * A1 *

p8_O22 * A0

+ /A5* /A9* /A6 * /^M1 * /A7 * /^RD * ^IORQ * A3 *

/A2 * /A1 * /p8_O22 * /A0

+ A5* /A9* A6 * /^M1 * /A7 * /^RD * ^IORQ * /A3 *

A2 * A1 * /p8_O22 * /A0

+ /A5* A9* /A6 * /^M1 * A7 * /^RD * ^IORQ * A3 *

A2 * A1 * /p8_O22 * /A0

+ /A5* /A9* /A6 * /^M1 * /A7 * /^RD * ^IORQ * /A3 *

/A2 * /A1 * /p8_O22 * A0
```

Partial equation to IC8. Take this one a line at a time:

IOREQ to port 0xBB OR

Memory Read location 0x0008 (RST8) OR

Memory Read location 0x0066 (NMI code) OR

Memory Read location 0x028E (Keyscan) OR

Memory Read location 0x0001 - Note at hardware reset the logic for this does not trigger, due to a presumed race condition on the RESET line and the Disciple logic to page itself out on RESET. This was tested on real hardware (Disciple & Spectrum +2).

```
O17.TRST = VCC

/O17 = A5* A6 * ^M1 * /A7 * /^IORQ * A3 * /A2 * A1 *

p8_O22 * A0
```

Partial equation to IC8. IOREQ to port 0x7B (reset / set boot port)

```
O16.TRST = VCC

/O16 = /A5* ^M1 * /^IORQ * A3 * /A2 * A1 * p8_O22 * A0
```

This is CE for IC4 WD1772 disk controller. Condition met when any IOREQ to ports (0x1B, 0x5B, 0xDB, 0x9B)

```
O15.TRST = VCC

/O15 = /A5* /A6 * ^M1 * /A7 * /^WR * /^IORQ * A3 * A2 *

A1 * p8_O22 * A0
```

This will generate a clock pulse for IC5 latch when data is written to the control IO port (0x1F, 31 Decimal). IC5 is the physical implementation of OUT 0x1F :

Port #1F OUT:

B0	Drive Select
B1	Side select
B2	Single / Double Density
B3	ROM bank select
B4	Inhibit switch control
B5	Through Edge connector A30
B6	Printer strobe
B7	Network

The Plus D PAL Equations explained

The Plus D has one PAL IC, a 20L8, the same as used in the Disciple. The equations are naturally different, the Plus D uses different IO ports and differs in some addresses being paged in (notably address 0x003A – spectrum maskable interrupt). I've also noticed that it doesn't have full decoding logic so some of the IO ports can be accessed on multiple addresses. Also, note some of the address line decoding logic is done by discrete logic (IC10A, IC10B, IC10C and IC9D), the output of which is fed into pin 23 of the PAL IC. My reading of this discrete logic says when all of the lines fed to the inputs are zero, the output will be 1. This is primarily used to decode the high address lines when doing IOREQ operations (my guess it avoids the need for another PAL IC).

Unlike the Disciple, the Plus D is NOT paged in at address 0x0001 or 0x0000 (as stated by RAMSOFT guide). And again, unlike the Disciple it doesn't have logic to do anything special on a RESET.

```
; JED2EQN -- JEDEC file to Boolean Equations disassembler (Version V063)
; Copyright (c) National Semiconductor Corporation 1990-1993
; Disassembled from alice.dpl. Date: 10-17-101
chip alice PAL20L8

i1=1 i2=2 i3=3 i4=4 i5=5 i6=6 i7=7 A6 =8 i9=9 i10=10 i11=11 GND=12
i13=13 i14=14 o15=15 o16=16 f17=17 o18=18 o19=19 o20=20 f21=21
o22=22 i23=23 VCC=24

/o22 = /MEMREQ* //IOREQ * //WR * A6 * A5 * /A2 * A1
IOREQ write access to 0xE2,0xE3,0xEA,0xF3,0xFB - WD1772 /CS
   + /MEMREQ* //IOREQ * //RD * A6 * A5 * /A2 * A1
IOREQ read access to 0xE2,0xE3,0xEA,0xF3,0xFB - WD1772 /CS
o22.oe = vcc

/f21 = /MEMREQ* //IOREQ * //WR * A6 * A5 * /A4 * A2 * /A3 * A1
   + f17
IOREQ write access to 0xE6,0xE7 (MISSING A7 line) - page out +D
Spectrum /ROMCE - Pages out Spectrum ROM by setting F17 high (need to recheck)
f21.oe = vcc

/o20 = /MEMREQ* //IOREQ * //RD * A6 * A5 * A4 * A2 * /A3 * A1
IOREQ to port 0xF7 (missing A7 line decode) - Printer busy / strobe LS175
o20.oe = vcc

/o19 = /MEMREQ* //IOREQ * //RD * A6  * A5 * /A4 * A2 * A3 * A1
IOREQ to port 0xEF (missing A7 line decode) - Disk control (side, A/B) LS175 clk
o19.oe = vcc

/o18 = /MEMREQ* //IOREQ * //WR * A6 * A5 * A4 * A2 * /A3 * A1
IOREQ WR to ports 0xF6,0xF7 (missing A7 line decode) - Printer output LS374 clk
o18.oe = vcc
```

```
/f17 = f21
IOREQ write to 0xE6,0xE7 (missing A7 line decode) - page out +D

   + /MEMREQ* //IOREQ * //RD * A6 * A5 * /A4 * A2 * /A3 * A1
IOREQ read to 0xE6,0xE7 (missing A7 line decode) - page in +D

   + /A15 * /A14* /A13 * i23 * //MEMREQ* /IOREQ * //RD * /A6  * /A5 * /A4 * /A2
     * A3 * /A1
Memory read at 0x0008 (RST 8 instruction).
Note: i23 is given by discrete logic (IC9, IC10) outside of the PAL – simply put,
A0,A7,A8,lA9,A10,A11,A12 must all be low for i23 to be high.

+ /A15 * /A14* /A13 * i23 * //MEMREQ* /IOREQ * //RD * /A6 * A5 * A4 * /A2
     * A3 * A1
Memory read at 0x003A (Spectrum executing maskable interrupt)

   + /A15 * /A14* /A13 * i23 * //MEMREQ* /IOREQ * //RD * A6  * A5 * /A4 * A2
     * /A3 * A1
Memory read at 0x0066
f17.oe = vcc

/o16 = /A15 * /A14* A13 * //MEMREQ* f21 * //RD
MEMREQ <16384 read (if paged in)  enable Plus D RAM
   + /A15 * /A14* A13 * //MEMREQ* f21 * //WR
MEMREQ <16384 write (if paged in) enable Plus D RAM
o16.oe = vcc

/o15 = /A15 * /A14* /A13 * //MEMREQ* f21 * //RD
MEMREQ <8192 read (if paged in) enable Plus D ROM
o15.oe = vcc
`
```

Why did you do this you madman?

The first time round – 2003 - Amazing what'll you'll do to get your beloved disciple working again after 10 years...... and my way of giving back to both the open source and Sinclair community, as well as helping to fix other broken beautiful Disciple interfaces.

The second time round – 2024 – I was disturbed to find the Disciple was not working in FUSE for the 128k models and was determined to find out why. This also led to me doing the Plus D PALs.

You're wrong about equation XYZ!

Tell me ! This is my interpretation and may well have errors. Let me know, I'll check and fix. Email me at alandpearson@gmail.com

I just want to program my PAL chips! Can I have the jedec source files please?

By all means, download IC8 and IC9 for the Disciple and IC4 for the Plus D.

Okay what about the Plus D?

~~Hmm haven't got that far yet. The jedec source is HERE for the single PAL in it, but I haven't dissassmbled or figured out it's equations yet. Someday.~~ Done !

How can I thank you?

Thank Bruce Gordon for making such wonderful interfaces, and Rudi Biesma for sending me the jedecs.

And the Disciple circuit diagram / schematic?

Oh yeah, here you go.

Thanks to Rudy Biesma for supplying the disciple fuse maps, and Ian Worsley for the Plus D. Also thanks to Ramsoft for producing the disciple technical guide, and the excellent realspec emulator.

Alan Pearson, (alandpearson@gmail.com) March 2024