MLDL2000 Design Specifications

Introduction

The MLDL2000 is a Machine Language Development Lab (MLDL) for the HP41 series of calculators with large amounts of EPROM and SRAM. The basic principle is loosely based on the classic MLDL as published in the PPC Journal, V9N3 in the article by Lynn A. Wilkins. This design will be referenced to as the 'Classic MLDL'. Added is an I/O Interface to exchange data with a PC and modified interfacing with the actual memory devices. This specification serves as a description for the HP41 bus as well and touches upon several issues like bank switching and other topics that are relevant for the design and use of the MLDL2000. The CPLD based MLDL2000 allows changing, expanding or upgrading the functionality.

As the name implies, the project started just before the turn of the century. Available time limited the amount of work that could be done on the project, but the start of the basic thinking and collecting information dates from the end of 1999. The design is based on a Xilinx programmable device. A prototype environment was purchased in December 2002, based on a 256-macrocell XPLA3 device from Xilinx. The final MLDL2000 is realized with a 384 macrocell device.

In May 2003 the first prototype of the MLDL2000 was running. The implementation was based on a simple ROMBox application, with support of only one bank of the traditional L8 and U2 EPROMs. The prototype setup was tested with a 2 speeded HP41C and an unmodified HP41CX. A more advanced prototype resembling the final product was done in January 2004, and the final PCB's were received in November 2004. Beta's were made in February 2005. First shipments out of production were done in July 2005. Preparations are underway for a new version 2 of the MLDL2000.*

All information about the MLDL2000 is published on my website and is public under the GPL license conditions. Working devices will be offered on the free market against reasonable cost, with the objective to make the MLDL2000 available to the HP41 user community on a non-commercial base. This specification is intended to be a source of information and to understand how the MLDL is working. A separate User Manual is available to explain the actual use of the MLDL2000.

Meindert Kuipers *Email:* meindert@kuiprs.nl

September 2008 *Website: www.kuiprs.nl*

References and credits

- VHDL for Programmable Logic, book by Kevin Skahill — ISBN 0201895730
- Museum of HP Calculators, website by David Hicks — www.hpmuseum.org
- HP41 Archive website, by Warren Furlow — www.hp41.org
- Inside the HP41C, article by Kelly McLellan — PPC Calculator Journal V6N7
- HP41 Bus Interfacing, article by Jim De Arras — PPC Calculator Journal V7N3
- HP41 Machine Language Development Lab, article by Lynn A. Wilkins — PPC Calculator Journal V9N3
- HP41C/CV/CX Technical Manual — www.hp41.org
- Xilinx tools, datasheets, application notes en support site — www.xilinx.com
- Diego Diaz for his cooperation and his Clonix and NoVRAM modules
- Many individual contributors for their tips and encouragement
- FTDI USB interface IC FT2232C — www.ftdichip.com,

There is no particular order of importance intended.

Conventions

00FF	Hexadecimal numbers are used to indicate addresses, leading zero's are typically used to indicate the total possible range, e.g. 00FFFF. The 'x' character is used for a "don't care" situation
$040	The '$' character is used to indicate hexadecimal values when context is not clear
17o	Octal values are indicated with the character 'o' after the value
%1010	Binary values are preceded with the '%' character, the 'x' character is used for don't care bits
/OE	Signal names preceded by a slash ("/") indicate that this signal is either active low or that it becomes active at a falling edge. 'Asserted' or 'active' in this case means that the signal is low or logic '0';
in/out	Signal directions generally refer to the MLDL2000 assembly or CPLD, "in" meaning "to the MLDL2000", "out" meaning "coming from it"
HP41	HP41 indicates all versions of the HP41 calculator, including HP41C, CV, CX, etc

Copyrights and Disclaimer

© Copyright 2005-2008 by Meindert Kuipers, The Netherlands
This document, the MLDL2000 design, VHDL code and software are copyrighted under the GNU General Public License. This means that anyone is free to use the design for his or her own purposes and may modify it. The MLDL2000 and accompanying software is supplied "as is" without warranty of any kind nor do I assume any kind of liability including consequential damages. The specifications, functionality or contents may be changed without notification to the user. If you wish to incorporate (parts of) the design into products which are distributed under any other conditions than the GNU Public License (commercial on non-commercial) you will have to have permission from the author. Please make certain that you have read and understood the GNU General Public License if you plan to use (parts of) the design.
Note that some parts of the MLDL2000, specifically the USB interface and driver software, are commercially licensed but free (without source code). Any license fees should be handled directly with the manufacturer. The development software for the Xilinx devices is basically free but requires registration with Xilinx. The application software controlling the USB module is written in Borland Delphi, the compiler must be licensed. All brand or product names are trademarks or registered trademarks of their respective holders.
Information in this document has been carefully checked and is believed to be accurate as of the date of publication; however, no responsibility is assumed for inaccuracies. I will not be liable for any consequential or incidental damages arising from reliance on the accuracy of this document. The information contained herein is subject to change without notice.

Table of Contents

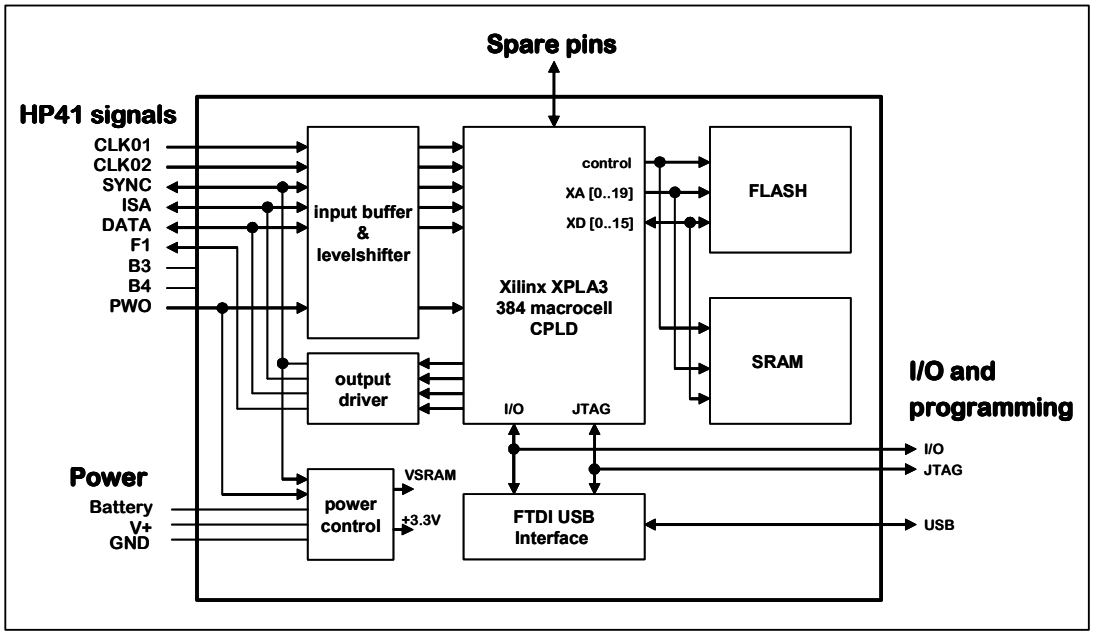
Introduction 1
References and credits 1
Conventions 1
Copyrights and Disclaimer 2
Table of Contents 2
Scope of the MLDL2000 3
What is an HP41 MLDL? 3
MLDL2000 Features 4
The HP41 interface bus 4
Signal Timing Description 5
MLDL2000 Operation 5
MLDL memory layout 8
Detailed Memory Map 10
FLASH or SRAM read operation (Typical Instruction Fetch) 11
SRAM write operation (Typical WROM instruction) 11
Bank Switching 12
External interface 12
MLDL2000 Hardware Design 17
Bling-bling your MLDL2000 17
Power Control 18
Appendix A: HP41 Timing Diagram 19
Appendix B: MLDL2000 CPLD pinout for Xilinx XCR3384 CPLD 20
Appendix C: MLDL2000 Connections and PCB 22
Appendix D: MLDL2000 revision history 25
Appendix E: Errata 26

Scope of the MLDL2000

My father gave me my first HP21 when I was 15 or 16. This one was quickly upgraded to the programmable HP25. At that time I had a holiday job at a local technical bookstore, which also sold cash registers, computers and … HP calculators. When the HP41 came out I was quickly there to buy one, as that was very useful in my physics study. Having the '41 I quickly became a PPC member and through a message in the PPC Journal and the local HP dealer I came into contact with a group of other HP41 users. We had regular meetings, and being quite handy with a soldering iron I did many speed upgrades, module doubling (even did one triple module once) and built modules inside calculators. I ruined only one HP41 (my own), but quickly got another one. In the meantime I had built my own Eramco MLDL (bought the PCB's at Eramco and sorted out the rest myself) and had loads of fun with it. I even made my own rom: MEINROM which evolved into the –ML ROM, a co-production with another HP enthusiast.
After many years I decided to wipe the dust off my HP41. I also got myself (through eBay) a HP41CX in very good condition against a reasonable price. I downloaded the Xilinx WebISE environment for developing FPGA's and CPLD's and I decided to start using it to implement an MLDL. The Xilinx development environment is free (after registration) and offers a full VHDL suite with timing analyzer and simulator. A 3.3V Xilinx Coolrunner XPLA3 CPLD device is chosen for its low power consumption and possibility to interface directly to 5V CMOS and TTL levels.
I wanted both ROM emulation and MLDL (that is creating ROM) functionality. Combine that with a PC interface and the rough specs (or should I say thoughts) are quickly lined out. Flash EPROM it should be, to prevent use of programmers and UV erasers. SRAM obviously and a flexible interface to the outside world, also needed to get data into and out of the MLDL2000.
The MLDL2000 is my first serious attempt to program a device in VHDL. It will not be perfect and super efficient, but I intend to make it work. I donate the design to the user community for reproduction or improvement, on the condition that no commercial use is being made of the design and that I get all feedback. Users: have fun with it, and let me know!

What is an HP41 MLDL?

MLDL stand for Machine Language Development Lab. Sounds cool and high-tech, doesn't it? But what does it? The HP41 architecture understands two programming languages: User Code and Machine Code (or M-Code). The latter is normally not available to end-users. Expansion of the HP41 system is usually done with plug-in modules, or ROM (for Read Only Memory). These contain either User Code or M-code (or a mix thereof). These ROMs could be produced by HP only, although larger groups (like the PPC who made the famous PPC ROM) could develop code and have it produced by HP. When details of the HP41 interface became known it was apparent that the ROM could actually be emulated by (battery backed) RAM. An unused HP41 instruction was defined by the user community to enable writing into the RAM and the Machine Language Development Lab was born. It was now possible for end-users to write and test their own M-Code and later burn it into an EPROM.



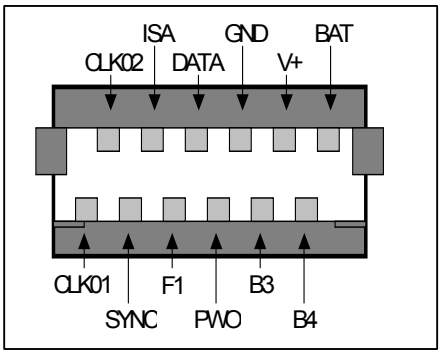
MLDL2000 Features

Features of the MLDL2000 are:

- 255 ROM Banks of FLASH EPROM memory, each 4k * 10 bits, can be relocated to any HP41 ROM location at any bank, including the HP41 System ROMS. Total capacity 1M * 10 bits
- 64 banks of MLDL SRAM memory, each 4k * 10 bits, can be relocated to any HP41 ROM location at any bank, including the HP41 System ROMS. Total capacity 512K * 10 bits (some units have 1M* 10 bits)
- Bi-directional serial I/O interface, can be relocated to any HP41 ROM location
- Each ROM/RAM block can be individually enabled or disabled
- Non-volatile Settings Registers, with multiple sets of Settings Registers enabled by switches
- USB interface to connect to PC for programming FLASH and SRAM and I/O with the HP41
- FLASH and SRAM memory programmable through PC interface, no eraser or programmer required
- In-System Programmable CPLD as control interface to allow upgrades and/or functional changes (special JTAG programmer cable required, reprogramming over USB a a future upgrade)
- Power consumption about 6-7 mA additional to the HP41 system (typically an HP41CV consumes about 6 mA when no extra modules or peripherals are connected) when running.

Other features may be added as required. In theory it is possible for example to mimic HP41 User Memory (and thus Extended Memory as well) if enough SRAM is available. As the CPLD is fully configurable, other interfaces might be added, like HEPAX support, printer, infrared, etc. Preparations will be taken to allow for these future expansions.
Please note that the user of the MLDL is responsible for the correct settings of the MLDL, as it is rather easy to create conflicts with ROMs sitting at the same address, especially if the HP41 has internal ROMS or if ROMS are plugged into any of the ports.
The MLDL2000 is designed to minimize the number of external parts and to minimize power consumption. The physics of the HP41 interface requires a more complicated circuit for driving the ISA (and DATA) line back to the HP41 in addition to level conversions from the HP41 (around 6 Volt) to the CPLD levels (3.3 Volt)
Minimal part count is important, to enable a version of MLDL2000 to be built in a very small space with a relatively easy layout to fit in a card reader housing.

The HP41 interface bus



This paragraph offers some background information for understanding the design of the MLDL2000. It is by no means a complete description of the HP41 interface. The HP41 interface bus is a serial bus available on the HP41 I/O ports for plugging in modules and peripherals. Information in this part is mainly based on the article by Jim De Arras in PPC Journal V7N3 and the HP41 Service Manual. The figure shows the signal assignment (view from the front of the module) with the following signals:

CLK01 Phase One Clock: This signal is used to read data on the bus lines and the rising edge of CLK01 qualifies the data on DATA and ISA for reading

CLK02 Phase Two Clock: This signal is used to count the phases in the processor cycle and to output data to bus lines.

SYNC SYNC Signal: The signal is used to indicate the start of the 56-bit processor cycle. It is low when the HP41 is in SLEEP mode, SYNC validates instructions in the ISA line. When fetching data or a peripheral is in control SYNC is remains low. Driver circuitry is anticipated for possible future use.

ISA ISA Signal: This line carries the 16-bit address of the ROM word to be fetched, and the ROM drives this line with the fetched 10-bit instruction word during the end of the cycle.

F1 F1/FIN, FLAG IN signal: This line triggers the HP41 when peripherals want to interrupt the system. Driver circuitry is anticipated for possible future use.

DATA Data Signal: This line is the serial input/output of CPU data, and typically reflects the contents of the CPU C-register. It is used by the MLDL for determining data and address for the word to be written to MLDL memory. The DATA signal is driven by the memory or peripheral device when reading data. Driver circuitry is anticipated for possible future use.

PWO Power On Signal: This line indicates to the system that the CPU is entering RUN mode and can be used for devices to enter power down mode when the signal is low in combination with SYNC. The HP41 is in SLEEP mode (switched off) when both PWO and SYNC are low. It is in STANDBY mode when PWO is low and SYNC is high.

GND Ground: Referral to system Ground.

B3 B3: Used in combination with B4 to determine the port address. B3 is high in ports 2 and 4, not connected in ports 1 and 3. B3 is not used on the MLDL2000 as the device is totally ignorant of the port it is plugged in.

V+ V+ is the system power signal at +6.5V, up to 20 mA in RUN and STANDBY mode. It drops to battery voltage in SLEEP mode.

B4 B4: Used in combination with B3 to determine the port address. B4 is high in ports 3 and 4, not connected in ports 1 and 2. B3 and B4 are not used on the MLDL2000 as the device is totally ignorant of the port it is plugged in.

BAT BAT is a diode isolated line to the HP41's battery, and may be used for devices drawing higher current. The line is always connected to the battery PLUS terminal.

Usage of B3/B4

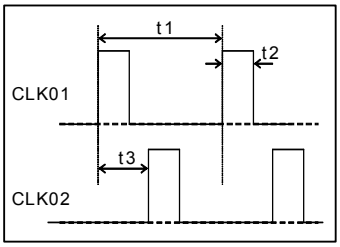
Note that 0 means not connected, and 1 means connected to Vcc. A pull down resistor is required for correct logical levels.

B4	B3	Port	Decoded Address Range
0	0	1	$8000-$9FFF
0	1	2	$A000-$BFFF
1	0	3	$C000-$DFFF
1	1	4	$E000-$FFFF

Signal Timing Description

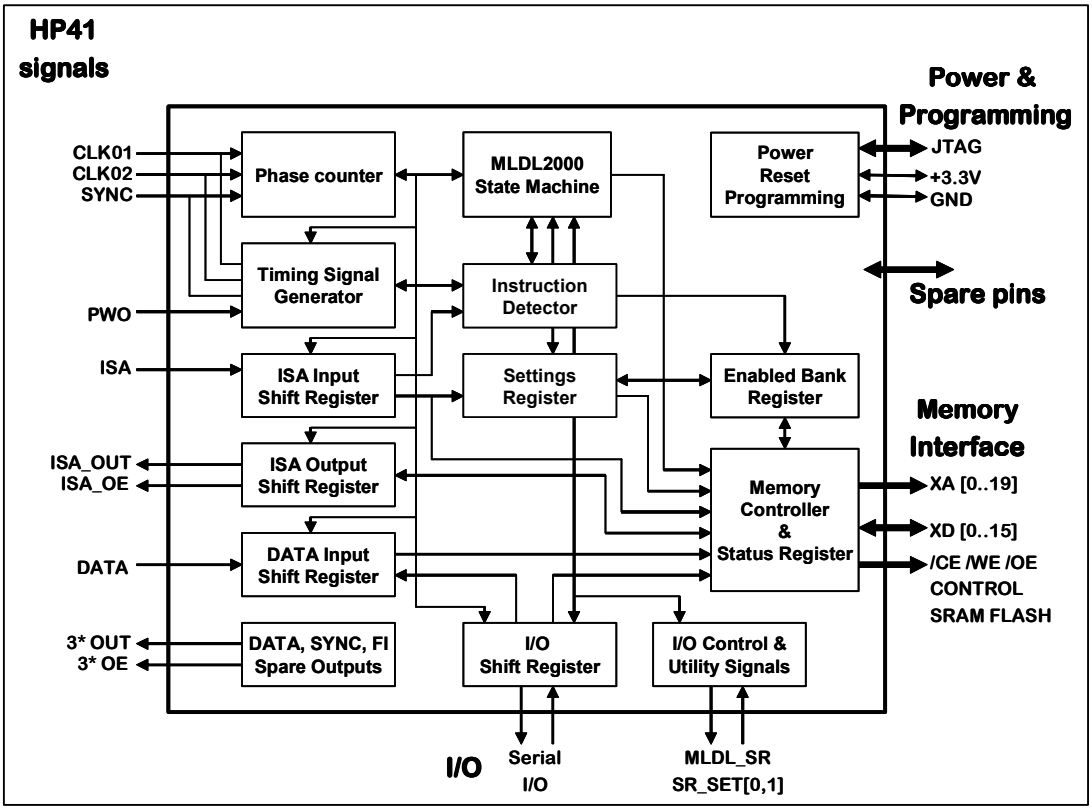
The HP41 Signal Timing is shown in the figure, and has a number of phases. The timing will only be described with respect to the relevance for the MLDL2000 and the user is encouraged to use other sources for a more elaborate description of the HP41 interface. The total Instruction Cycle Time for the original HP41C is about 160 µs (6300 instructions per second). Timing is measured in December 2002 on an HP41C (appr. 2 times speedup) and an HP41CX (no speedup). Exact timing may vary between individual machines. The CLK duty cycle is approximately 20%. An overview of the HP41 instruction cycle can be found in the appendix.

		Original HP41	Appr. 2* Speed-up
instructions/second		6400	9400
instruction cycle	Sync-to-sync	156.8 µs	106.4 µs
clock cycle	T1	2. 8 µs	1.9 µs
clock frequency		357 kHz	526 kHz
clock pulse width	T2	0.6 µs	0.38 µs
CLK01 to CLK02 delay	T3	1.04 µs	0.72 µs



MLDL2000 Operation

See the block diagram on page 3 for a high level overview. The exact pin-out is listed in Appendix B. The MLDL2000 is divided in the following logical blocks:



1. ***Input and Output buffers and level shifter***

   Buffers and level shifters are required to interface the 3.3V based CPLD with the HP41. The CPLD I/O's are 5V tolerant, but the HP41 power and signals are typically around 6.5V and that is above the maximum rating of the CPLD. The DATA

and SYNC lines are implemented as bidirectional signals, and FI as an output. The MLDL2000 is not driving DATA, SYNC or FI and the corresponding Output Enables are always driven high. This may be used in future upgrades.

2. ***Power Regulator***
Generates the 3.3V power for the CPLD and memory devices. In addition it supports circuitry for switching a backup battery for maintaining the contents of the SRAM on power down.

3. ***Phase Counter and Timing Logic (internal in CPLD)***
The serial bus of the HP41 needs a reference for the start of the 56-bit processor cycle, and the end of the SYNC signal serves this function. Due to the design of the Phase Counter in the MLDL2000 the number assigned to the phases is shifted by one clock. The MLDL Phase Counter is a 6-bit counter that resets at phase #55 (67o), as SYNC may not always be present (SYNC indicates an instruction being fetched, if there is no SYNC it is a read of data). The actual reset of the Phase Counter is done by latching subsequent (SYNC and CLK01) followed by (SYNC low and CLK01). Various events are indicated by the Phase Counter. Note that in the description the 'real' phase numbers are used, the phases as used in the MLDL VHDL design files are all one less due to the shifting.
The Phase Counter and Timing Logic is critical to the MLDL2000. Most important part are 6-bit phase counter and the SYNC detector. The SYNC detector resets the phase counter after detecting the presence of last falling edge of SYNC. Since SYNC may not be present at all times the SYNC detector also resets the phase counter if the count of 67o is reached. The ph_reset signal is an asynchronous reset for the phase counter. Some other logic and registers are used to indicate the various states of the MLDL2000, to facilitate control of the main state machine.

4. ***Instruction Detector: I_STAT (internal in CPLD)***
The Instruction Decoder is responsible for recognizing one of the supported instructions in the MLDL. Specific instructions may be defined here to allow I/O operations to user defined I/O or to communicate with the PC interface. Other options may exist to control the Settings Registers or to access FLASH and SRAM. The instruction decoder only operates when SYNC was present during the instruction fetch. If no SYNC was present then the 'instruction' might have been data or the second part of a branch instruction. The Instruction Detector controls the state of I_Stat:

I_IDLE		default state, returns to I_IDLE on start of ISA Address (phase 17 octal)
I_WROM	$040	write data in C[2:0] to address in C[6:3] (only in RAM version) or to I/O
I_FETCH	$330	read data from ROM address in C[6:3] to C[2:0]
I_ENBANK	$100, $180, $140, $1C0	enable ROM bank 1, 2, 3 or 4 for current device

5. ***Enabled Bank Registers: EBR (internal in CPLD or in SRAM)***
The EBR store information about the current active bank. Bank Switching is described in a separate paragraph.

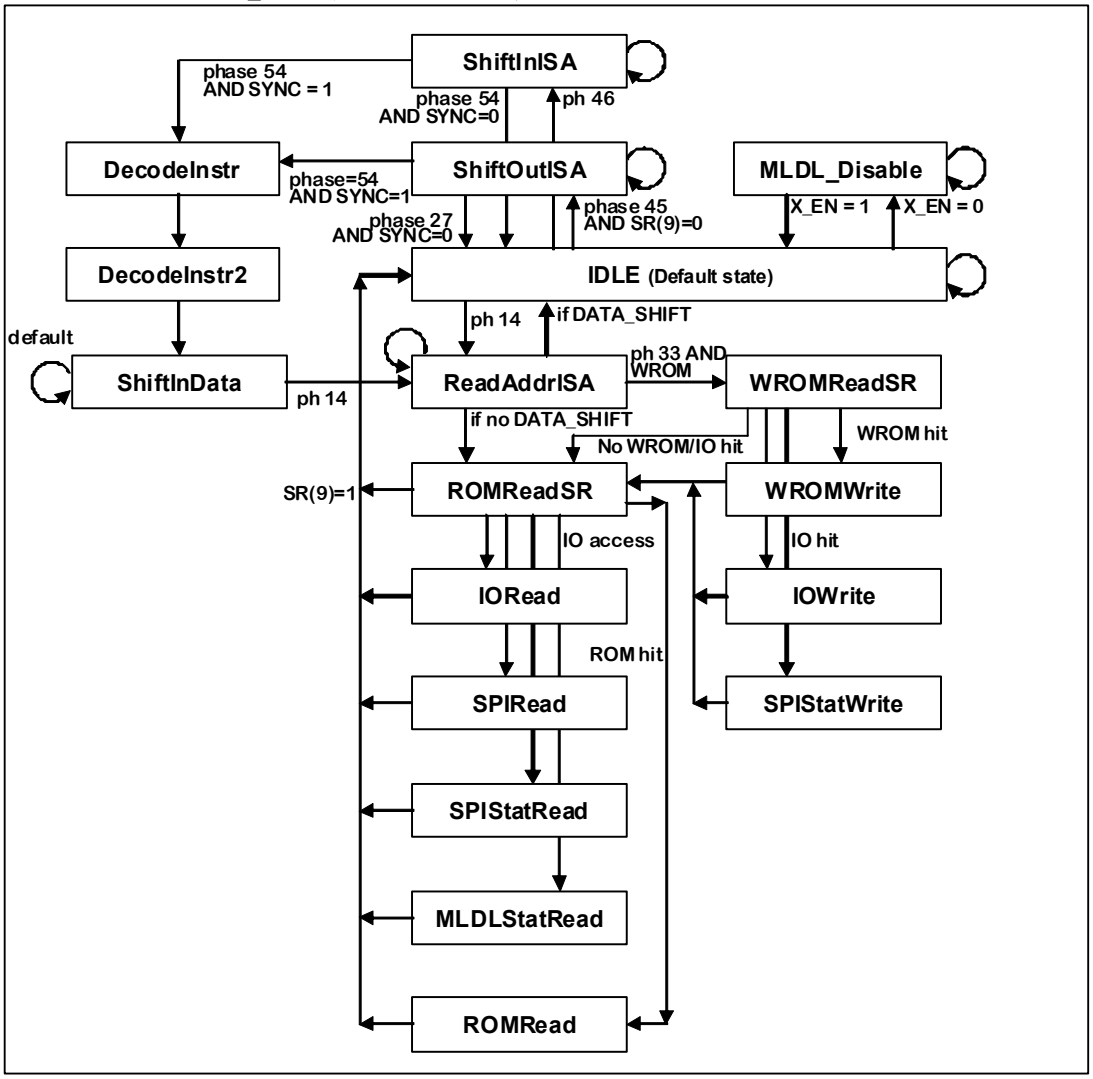
6. ***ISA Input Shift Register: ISR (internal in CPLD)***
The ISR reads the ISA line into a shift register for decoding the address and the instruction to act upon. When the external interface is enabled (assertion of /X_EN) the ISA Input Shift Register is used for the serial I/O data stream. The first task in the MLDL design is to determine the address of the instruction to be fetched. This is done by moving the data on ISA (starting at phase #16) to the ISA Shift Register (ISR) and using the 4 most significant address bits (which is the HP41 Port Address) to access the SR in combination with the currently selected Bank (2 bits).
An 10-bit word is returned by the SR, which indicates where in the MLDL2000 memory map the ROM block is located and if it is SRAM, FLASH or IO and if the Port/Bank combination is enabled at all. In addition, SRAM might be write-protected.
Having read the 10-bit word from SR, the data is used in combination with the 12 least significant address bits to address the appropriate location in EPROM, SRAM or I/O.

7. ***Data Input Shift Register: DSR (internal in CPLD)***
The Data Input Shift Register takes the HP41 DATA line and extracts information from it for the WROM instruction. D0-D9 contains the 10-bit instruction, D12-27 carries the 16-bit address. The DATA line is used only during the WROM and FETCH instructions and contains the address and the instruction to be written to the indicated location in MLDL memory. During the WROM instruction the DATA line is used as follows:

D00..09	Instruction to be written (lowest 10 bits of DATA in S&X), where D0 is LSB of data
D12..27	Address to write Instruction to (lowest bits of Mantissa), where D12 is LSB of address
D28..31	Spare bits for direct addressing of MLDL2000 memory and used by the I/O Interface

The WROM instruction sets up the MLDL for programming an SRAM location based on information that is present on the DATA line. For future additions a DATA Output with driver is available but currently not in use.
The DSR is used as shift register for I/O operations when /X_EN is asserted to allow access to SRAM and FLASH through the I/O interface.

8. Main State Machine: MLDL_STAT (internal in CPLD)



The Main State Machine is the heart and soul of the MLDL2000. Using various signals from the Phase Counter and Timing Logic is determines when and what actions should be taken for interfacing with memory and when data has to be shifted in or out. Most other blocks use the state output of this block for control.
The phase timing of the different states may vary, depending on the previous state. This allows for a much more flexible and straightforward state machine implementation (in a VHDL Finite State Machine). Note that IDLE does not mean that the MLDL has nothing to do, as various operations are in parallel to the Main State Machine, for example the timing of the DATA Shift Register and ISA Input Shift Register, but these are not under direct control of the Main State Machine.
The states SPIRead, SPIStatRead and SPIStatWrite are there for compatibility with V2 of the MLDL2000.

9. Settings Register: SR (internal in CPLD)
The Settings Register reads and stores the results of the access to the SR's in FLASH or SRAM to determine if and how an HP41 ROM (or RAM) page should be accessed.

10. ISA Output Shift Register: OSR (internal in CPLD)
Transfers parallel data from (FLASH or SRAM) memory as output to the HP41 on the ISA line. This block has a direct interface with the external data bus.

11. Memory Access Logic (internal in CPLD)
The Memory Access Logic controls the external memory devices FLASH and SRAM. It is an 20-bit wide address and 16-bit wide data bus, with separate chip enable, write enable and output enable for external devices. The Memory Access Logic is the largest piece of logic in the MLDL2000. Every state of MLDL_STAT that demands a memory access is handled in this block. Manipulation of the various address and data lines are described later. Due to the operation of the Main State Machine the memory access cycles may not always be in the same clock phase.

12. ***FLASH (Non Volatile Flash Memory)***
    Non Volatile Flash Memory replaces the traditional EPROM's on the MLDL. A Flash programming interface will be provided through the IO Interface, as Flash programming through the HP41 interface is probably not an option due to the strict timing requirements for programming. The FLASH contains both the ROM images and Settings Registers, and the Memory Map layout will be described later. The FLASH memory is a 3.3V single voltage device that does not require any special programming voltages.

13. ***SRAM (Non Volatile Battery Backed Static Memory)***
    SRAM is used for storing user developed ROM images in combination with Settings Registers. Data may be written to the SRAM by the HP41 by means of the WROM ($040) instruction.

14. ***SR (Settings Registers) (in FLASH and SRAM memory)***
    The Settings Registers contain the memory layout of the MLDL, i.e. which ROM bank can be found on which memory location in either FLASH or SRAM, and if the ROM is enabled at all. The SR can be in SRAM or FLASH, depending on the setting of an external input. This is done mainly to prevent unexpected behavior of the MLDL when the SRAM is empty or corrupted (for example after battery failure or exchange). Four sets of SR are supported in both FLASH and SRAM.

15. ***IO Shift Register: IOR (internal in CPLD)***
    The Input/Output Interface Logic can be used to communicate with the outside world and offers a serial interface port using a simple protocol. The I/O Interface is described in more detail in a separate chapter. The IO Interface is memory mapped in one of the (Bank switched) HP41 ROM spaces, as this is easiest to implement. Memory mapped I/O is chosen to make the MLDL totally independent and not to having to rely on certain ROM contents or instructions.
    A separate part of the IO Interface Logic is used to control the MLDL2000 itself and allows reading, writing, erasing and programming of SRAM, FLASH and SR through the external interface that can be controlled from the parallel port of a PC or any other device that supports free control of a number of digital I/O lines.

16. ***JTAG Programming Interface***
    A JTAG interface is provided to enable programming of the CPLD logic. A special programming cable (supplied by Xilinx or third parties) with software should be used to (re-)program the CPLD. It is also possible to program the CPLD through the USB Interface (not supported in the Beta version). The JTAG pins are shared with some pins of the External I/O Interface.

17. ***Spare I/O pins***
    A number of pins that are unused on the CPLD are routed to pads on the MLDL2000 PCB. These may be used for debugging or future or custom applications.

MLDL memory layout

Purpose of the MLDL is to be as flexible as possible, and enable as much possibilities as is practical. This means that the settings of the MLDL will be somewhat complicated and that these settings will have to be programmable from the HP41 itself as well as from the external interface.
The MLDL2000 has two external devices which are controlled by their own Chip Select and Enable lines: FLASH and SRAM. The Settings Registers are embedded in either the FLASH or SRAM. An external switch indicates if SRAM or FLASH is to be used for the SR by selecting the MLDL_SR input. In addition, signals SR_SET[0..1] enable to choose 1 out 4 sets of Settings Registers to allow fast switching between different configurations. It is strongly recommended to use one of the configurations with a minimum setting to recover from lock-up situations (for example after use of the Service ROM), and possibly another set to completely disable the MLDL2000. A 4th signal, MLDL_DIS, is used to disable all output from the MLDL2000 to the HP41, for example when initializing the MLDL2000.
To access the devices a 20-bit address bus (XA, for eXternal Address) is used with the 16-bit XD (eXternal Data). The top block of FLASH and SRAM is used to store the Settings Registers. Of the 16 data bits only the 10 least significant bits are actually used.
The SRAM memory above $40000 can not be used for storing ROM images, as the SR's do not allow usage of address lines A18 and A19. The SR's are stored in SRAM starting at address $7FF00. The I/O interface allows direct access to this area for specific applications. Possible future expansions of the MLDL2000 could include access of the SRAM area above $40000 for emulation of user registers.
The FLASH device used is the AM29LV160DB, with a sector size of 32K words (good for 8 ROM images each). The lowest 4 blocks are boot blocks and have smaller sector sizes (see table). Since there is plenty of room in the FLASH memory it is advised to spread the ROM images as much as possible to allow erasing of only a single ROM image. Best practice is to collect all ROMS that will never change (like copies of the original ROMS) together in FLASH sectors, and spread out those that are more likely to change.
The Settings Registers are all in one sector of the FLASH memory. Changing a value may require erasing and reprogramming of the entire sector! It is therefore best not to keep any ROM images in the sector of the Settings Registers.

Each of the individual 10-bit Settings Registers has the following layout:

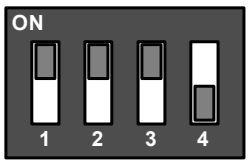
Setting Register Layout									
Bit 9	***Bit 8***	***Bit 7***	***Bit 6***	***Bit 5***	***Bit 4***	***Bit 3***	***Bit 2***	***Bit 1***	***Bit 0***
EN	FL/SR=1	IO	WP	A17	A16	A15	A14	A13	A12
EN	FL/SR=0	A19	A18	A17	A16	A15	A14	A13	A12

Bit 9: EN, Enable
0 = FLASH/SRAM/IO Bank is enabled 1 = FLASH/SRAM/IO Bank is disabled
Bit 8: FL/SR: Flash or SRAM
0 = ROM Image is in FLASH 1 = ROM Image is in SRAM
If Bit 9 is 0 (ROM Image is in FLASH) then Bit 5 to 0 have the following meaning:
Bit 7-0: A19-12 for addressing the FLASH memory (or the ROM image number)
If Bit 8 is 1 (ROM Image is in SRAM) then Bit 5 to 0 have the following meaning:
Bit 7: IO, I/O Interface
0 = Bank is SRAM 1 = Bank is IOI
Bit 6: WP: Write Protect (for SRAM only, not for I/O)
0 = Write Enabled 1 = Write Protected
Bit 5-0: A17-12 for addressing the SRAM memory (or the ROM image number)
If Bit 8 is 1 and Bit 7 is 1 (I/O Interface) then bit 6 indicates if the image is mapped in SRAM or FLASH. The block can therefore not be write protected. When the image is in FLASH, it can only be located in the first 64 blocks of FLASH (FLASH 0-63, addresses $00000-$3FFFF.
Bit [9-8-7] = '1-1-1' (I/O Bank)
Bit 6: 0 = ROM mapped in FLASH 1 = ROM mapped in SRAM

NOTE 1: For writing, an SRAM bank does not have to be enabled (bit 0) to allow initializing memory but it must be Write Enabled.
NOTE 2: SRAM banks that do not exist in an HP41 memory map must be write protected, and all Setting Registers must be initialized at meaningful values to prevent overwriting existing memory contents.
NOTE 3: It is possible to have different SR's point to the same ROM bank, although this might give unexpected results. This would be the same as plugging identical modules in multiple ports.
NOTE 4: Although multiple IO banks are possible, these all alias to the same physical I/O interface and registers for accessing SRAM and FLASH.
NOTE 5: A disabled bank is best indicated with $3FF in the SR. This allows for non-initialized FLASH to be used as SR that can be reprogrammed without first erasing the whole block.
NOTE 6: FLASH memory is erased by sector, and a sector may contain multiple ROM images and/or Setting Register sets.
NOTE 7: FLASH memory bits can only be programmed from '1' to '0'. An erase operation (of a complete sector) is required to set bits from '0' to '1'.
NOTE 8: An I/O block cannot be write protected when it is in SRAM.
NOTE 9: An I/O block can only be mapped in the first 64 blocks of FLASH.

The switch has the following layout. Note that when ***ON*** the respective input is connected to GND. When ***OFF*** the input is pulled up by the internal pullup resistors.

Switch 1	MLDL_DIS	***OFF***:		MLDL Disabled	***ON***:	MLDL Enabled
Switch 2	MLDL_SR	***OFF***:		SR from FLASH	***ON***:	SR from SRAM
Switch 3-4	SR_SET[1,0]	***OFF***	- ***OFF***	SR Set 0, base $FFF00 (FLASH), $7FF00 (SRAM)		
		OFF	- ***ON***	SR Set 1, base $FFF40 (FLASH), $7FF00 (SRAM)		
		ON	- ***OFF***	SR Set 2, base $FFF80 (FLASH), $7FF00 (SRAM)		
		ON	- ***ON***	SR Set 3, base $FFFC0 (FLASH), $7FF00 (SRAM)		



Detailed Memory Map

FLASH Memory Layout		
Start	**End**	**Contents**
FFF00	FFFFF	**SECTOR 34 Settings Registers**
F8000	FEFFF	**SECTOR 34, ROM 248-254**
F0000	F7FFF	**SECTOR 33, ROM 240-247**
E8000	EFFFF	**SECTOR 32, ROM 232-239**
E0000	E7FFF	**SECTOR 31, ROM 224-231**
D8000	DFFFF	**SECTOR 30, ROM 216-223**
D0000	D7FFF	**SECTOR 29, ROM 208-215**
C8000	CFFFF	**SECTOR 28, ROM 200-207**
C0000	C7FFF	**SECTOR 27, ROM 192-199**
B8000	BFFFF	**SECTOR 26, ROM 184-191**
B0000	B7FFF	**SECTOR 25, ROM 176-183**
A8000	AFFFF	**SECTOR 24, ROM 168-175**
A0000	A7FFF	**SECTOR 23, ROM 160-167**
98000	9FFFF	**SECTOR 22, ROM 152-159**
90000	97FFF	**SECTOR 21, ROM 144-151**
88000	8FFFF	**SECTOR 20, ROM 136-143**
80000	87FFF	**SECTOR 19, ROM 128-135**
78000	7FFFF	**SECTOR 18, ROM 120-127**
70000	77FFF	**SECTOR 17, ROM 112-119**
68000	6FFFF	**SECTOR 16, ROM 104-111**
60000	67FFF	**SECTOR 15, ROM 96-103**
58000	5FFFF	**SECTOR 14, ROM 88-95**
50000	57FFF	**SECTOR 13, ROM 80-87**
48000	4FFFF	**SECTOR 12, ROM 72-79**
40000	47FFF	**SECTOR 11, ROM 64-71**
38000	3FFFF	**SECTOR 10, ROM 56-63**
30000	37FFF	**SECTOR 09, ROM 48-55**
28000	2FFFF	**SECTOR 08, ROM 40-47**
20000	27FFF	**SECTOR 07, ROM 32-39**
18000	1FFFF	**SECTOR 06, ROM 24-31**
10000	17FFF	**SECTOR 05, ROM 16-23**
08000	0FFFF	**SECTOR 04, ROM 08-15**
04000	07FFF	**SECTOR 03, ROM 04-07**
03000	03FFF	**SECTOR 02, ROM 03**
02000	02FFF	**SECTOR 01, ROM 02**
00000	01FFF	**SECTOR 00, ROM 00-01**

SRAM Memory Layout		
Start	**End**	**Contents**
7FF00	7FFFF	**Settings Registers**
70000	7FEFF	**Reserved**
60000	6FFFF	**Reserved**
50000	5FFFF	**Reserved**
40000	4FFFF	**Reserved**
30000	3FFFF	**ROM 48-63**
20000	2FFFF	**ROM 32-47**
10000	1FFFF	**ROM 16-31**
00000	0FFFF	**ROM 00-15**

SETTINGS REGISTERS MEMORY LAYOUT		
ADDR	**HP41 Port**	**Bank**
FFFFF	F	3
FFFFE	F	2
FFFFD	F	1
FFFFC	F	0
FFFFB	E	3
FFFFA	E	2
FFFF9	E	1
FFFF8	E	0
FFFF7	D	3
FFFF6	D	2
FFFF5	D	1
FFFF4	D	0
FFFF3	C	3
FFFF2	C	2
FFFF1	C	1
FFFF0	C	0
FFFEF	B	3
FFFEE	B	2
FFFED	B	1
FFFEC	B	0
FFFEB	A	3
FFFEA	A	2
FFFE9	A	1
FFFE8	A	0
FFFE7	9	3
FFFE6	9	2
FFFE5	9	1
FFFE4	9	0
FFFE3	8	3
FFFE2	8	2
FFFE1	8	1
FFFE0	8	0
FFFDF	7	3
FFFDE	7	2
FFFDD	7	1
FFFDC	7	0
FFFDB	6	3
FFFDA	6	2
FFFD9	6	1
FFFD8	6	0
FFFD7	5	3
FFFD6	5	2
FFFD5	5	1
FFFD4	5	0
FFFD3	4	3
FFFD2	4	2
FFFD1	4	1
FFFD0	4	0
FFFCF	3	3
FFFCE	3	2
FFFCD	3	1
FFFCC	3	0
FFFCB	2	3
FFFCA	2	2
FFFC9	2	1
FFFC8	2	0
FFFC7	1	3
FFFC6	1	2
FFFC5	1	1
FFFC4	1	0
FFFC3	0	3
FFFC2	0	2
FFFC1	0	1
FFFC0	0	0

Above is configuration:
SR_SET(0) = 0, SR_SET(1) = 0
SRAM SR from $7FF00 to $7FFFF

SETTINGS REGISTERS BASE SR_SET inputs	
SR_SET[0..1]	**SR Base**
00	FFFC0
01	FFF80
10	FFF40
11	FFF00

SR_Set inputs work for both FLASH and SRAM

NOTE: SRAM only has 64 ROM images. SRAM memory above $40000 is currently not used. Settings Registers use the top of the memory space, this block can not be used by a ROM image. Maximum 20 address bits are implemented, yielding an address space of $00000-$FFFFF. SRAM does not have A19 connected, and the areas above $7FFFF are aliased with regular addressable area.
NOTE: All addresses are hex!
NOTE: Some shipped MLDL2000 units have even more SRAM. The extra amount of memory is not used, but may be supported in future upgrades of the MLDL2000
NOTE: Some shipped MLDL2000 units have different FLASH memory IC's than indicated, where the sector layout is different (TOP BOOT BLOCK instead of BOTTOM BOOT BLOCK). This is recognized by the MLDL2000 software, and the correct list of sectors is displayed in the software.

FLASH or SRAM read operation (Typical Instruction Fetch)

HAxx	HP41 Address from ISA (Instruction Fetch), in ISA Shift Register
DAxx	HP41 Address from DATA (WROM operation), in DATA Shift Register
SRSx	SR_SET inputs
Ixx	Instruction from DATA (WROM operation)
Px	Current Active Bank Pointer (Banks 1, 2, 3 or 4)
XAxx	External SRAM/FLASH Address Bus
XDx	External SRAM/FLASH Data Bus
RAxx	Rom start Address inFLASH/SRAM, read from SR
Dx	Data Read from FLASH/SRAM

HP41 Address from ISA	HA15	HA14	HA13	HA12	HA11	HA10	HA09	HA08	HA07	HA06	HA05	HA04	HA03	HA02	HA01	HA00

Entry in Active Bank Table, addressed from HA[14..12]
(Bank Switching in system pages not supported)

Page 8 - 15	P1	P0

ROMReadSR, at $FFFC0	XA19	XA18	XA17	XA16	XA15	XA14	XA13	XA12	XA11	XA10	XA09	XA08	XA07	XA06	XA05	XA04	XA03	XA02	XA01	XA00
same for SRAM and FLASH	1	1	1	1	1	1	1	1	1	1	1	1	/SRS1	/SRS0	HA15	HA14	HA13	HA12	P1	P0

Settings Register Read, uses inputs SR_SET[0..1].
Base at $FFF00 (FLASH) or $7FF00 (SRAM)

External Data Bus	XD09	XD08	XD07	XD06	XD05	XD04	XD03	XD02	XD01	XD00
if FLASH (FL/SR = 1)	EN	FL/SR	RA19	RA18	RA17	RA16	RA15	RA14	RA13	RA12
if SRAM (FL/SR = 0)	EN	FL/SR	IO	WP	RA17	RA16	RA15	RA14	RA13	RA12

ROMRead	XA19	XA18	XA17	XA16	XA15	XA14	XA13	XA12	XA11	XA10	XA09	XA08	XA07	XA06	XA05	XA04	XA03	XA02	XA01	XA00
if FLASH	RA19	RA18	RA17	RA16	RA15	RA14	RA13	RA12	HA11	HA10	HA09	HA08	HA07	HA06	HA05	HA04	HA03	HA02	HA01	HA00
if SRAM	0	0	RA17	RA16	RA15	RA14	RA13	RA12	HA11	HA10	HA09	HA08	HA07	HA06	HA05	HA04	HA03	HA02	HA01	HA00

Read data from FLASH or SRAM

Output Shift Register	OSR9	OSR8	OSR7	OSR6	OSR5	OSR4	OSR3	OSR2	OSR1	OSR0
	D09	D08	D07	D06	D05	D04	D03	D02	D01	D00

SRAM write operation (Typical WROM instruction)

HP41 Address from DATA[27..12]	DA15	DA14	DA13	DA12	DA11	DA10	DA09	DA08	DA07	DA06	DA05	DA04	DA03	DA02	DA01	DA00

Instruction te be written from DATA[09..00]	I09	I08	I07	I06	I05	I04	I03	I02	I01	I00

Entry in Active Bank Table, addressed from HA[14..12]
(Bank Switching in system pages not supported)

Page 8 - 15	P1	P0

WROMReadSR, at $FFFC0	XA19	XA18	XA17	XA16	XA15	XA14	XA13	XA12	XA11	XA10	XA09	XA08	XA07	XA06	XA05	XA04	XA03	XA02	XA01	XA00
same for SRAM and FLASH	1	1	1	1	1	1	1	1	1	1	1	1	/SRS1	/SRS0	DA15	DA14	DA13	DA12	P1	P0

Settings Register Read, uses inputs SR_SET[0..1].
Base at $ FFF00 (FLASH) or $7FF00 (SRAM)

	XD09	XD08	XD07	XD06	XD05	XD04	XD03	XD02	XD01	XD00
if FLASH (FL/SR = 0)	EN	FL/SR	RA19	RA18	RA17	RA16	RA15	RA14	RA13	RA12
if SRAM (FL/SR = 1)	EN	FL/SR	IO	WP	RA17	RA16	RA15	RA14	RA13	RA12

WROMWrite	XA19	XA18	XA17	XA16	XA15	XA14	XA13	XA12	XA11	XA10	XA09	XA08	XA07	XA06	XA05	XA04	XA03	XA02	XA01	XA00
only SRAM possible	0	0	RA17	RA16	RA15	RA14	RA13	RA12	DA11	DA10	DA09	DA08	DA07	DA06	DA05	DA04	DA03	DA02	DA01	DA00

Write L8 data to SRAM

External Data Bus	XD09	XD08	XD07	XD06	XD05	XD04	XD03	XD02	XD01	XD00
	I09	I08	I07	I06	I05	I04	I03	I02	I01	I00

Bank Switching

Bank Switching is implemented in the MLDL2000 in a somewhat limited way because Bank Switching is not a function of the HP41 processor, but implemented in the expansion module. As there are various different implementations a generic mechanism is chosen that should support the most common ROMs.
Banks switching in the MLDL2000 uses registers that are located inside the CPLD: Enabled Bank Registers (EBR). This means that the state is forgotten when power is removed from the CPLD, which also happens in *STANDBY* mode. A jumper is provided on the MLDL2000 to allow the user to select the required mode,
Bank Switching is supported only in the ROM pages that are accessible to the user, ROM page $6 and up. Banks that are in the same port (i.e. the even and odd page) share the same set of EBR's, meaning that switching in an even page will switch banks in the odd page as well. For every set of pages 4 banks are possible. Please be aware of possible conflicts with modules or ROM images that implement bank switching in another way. Bank switching for the internal HP41CX ROMs is not supported.

External interface

The description of the external interface is for the purpose of documentation only. The CPLD print and USB print (with FT2232 USB controller) should always be used together and it is not recommended that users create their own hardware interfaces.
The MLDL has an external interface for two purposes: to enable initialization, updating and programming the FLASH and SRAM of the MLDL and to provide a means to exchange information with the HP41. The interface is done with a minimum number of signals to keep the interface on the MLDL side as simple as possible, while at the same time providing maximum control of the MLDL settings. The FLASH memory device requires special access cycles for erasing and programming, and therefore it must be possible to have full access to all address-, control- and data lines of the memory devices. A relatively simple serial I/O Interface is defined to communicate with the HP41 with a minimum amount of hardware as this facilitates the 'casual' user. Some of the I/O signals can be connected to the JTAG signals in order to allow programming of the CPLD through the external interface without the need for a special JTAG programming cable.
NOTE: Access cycles to the FLASH that are intended to program, erase or perform other commands should follow the exact device specifications.
The I/O Interface supports a serial mode which offers sufficient possibilities to control the MLDL2000 with a wide range of equipment. It is recommended that 3.3V I/O is being used, although the CPLD is 5V tolerant. Care should be taken not to exceed the maximum specified limits of the I/O inputs. For maximum flexibility I recommend using a USB I/O interface as this is very easy to connect to a computer. A USB interface based on the FTDI FT2232C is used and software was written in Delphi to support the basic functionality. This software is available in source code for free under the GNU license.
There is a shift register that can be accessed with the serial I/O interface, part of which is implemented using the Data Shift Register (DSR). This means that the MLDL cannot be used during I/O operations for accessing memory. The information is used to drive address and data lines for access to FLASH or SRAM. An additional Command determines what exactly needs to be done: read or write to or from the HP41.
While data is being shifted the MLDL will be disabled under control of the /X_EN input, because the HP41 cannot not access FLASH or SRAM at the same time. Disabling the MLDL by using the /X_EN line should only be done when the HP41 is in (light) sleep mode or when turned off to prevent crashes, lockups or other undesirable behavior. It basically switches off all internal clocks and HP41 signal lines into the MLDL2000 to allow full control of the memory interface.

MLDL pin	Function		
/X_EN	External Interface Enable, MLDL2000 will be disabled from the perspective of the HP41. This is best done while the HP41 is not active. /X_EN should not be asserted to communicate with the HP41. By default /X_EN is de-asserted. For I/O with the HP41 /X_EN should not be asserted.	in	active low, 3.3 V
X_DIN	External Data Input, combined with the JTAG signal TDI.	in	active high, 3.3 V
X_DOUT	External Data Output, combined with the JTAG signal TDO.	out	active high, 3.3 V
/X_CLK	External Clock, rising edge is used to clock data in or out. The external interface is designed in such a way that data is latched on the rising edge of X-CLK. During a Read command /X_CLK is used to latch data from FLASH or SRAM into output register when /X_STROBE is asserted. /X_CLK is used to assert the /OE or /WE signals and proper timing for the memory devices should be observed. Signal is combined with the JTAG signal TCK.	in	active low, 3.3 V
/X_STROBE	External Data Strobe, Execute Command. /X_STROBE is used to drive the data lines and /CE signals of the FLASH and SRAM devices on the MLDL2000 and proper timing should be observed. X_STROBE finalizes data transfer in the IOR when writing to the MLDL2000 for HP41 I/O. Signal is combined with the JTAG signal TMS..	in	active low, 3.3 V
/X_DAV	I/O Data Available from HP41 in MLDL, ready for reading when high. The data will already be on the outputs of X_D when /X_OE is asserted. Data should be read with a pulse on /X_CLK or /X_RD, this will reset /X_DAV. /X_DAV is directly connected to H_DAV in the internal MLDL I/O Register for the HP41.	out	active high, 3.3 V

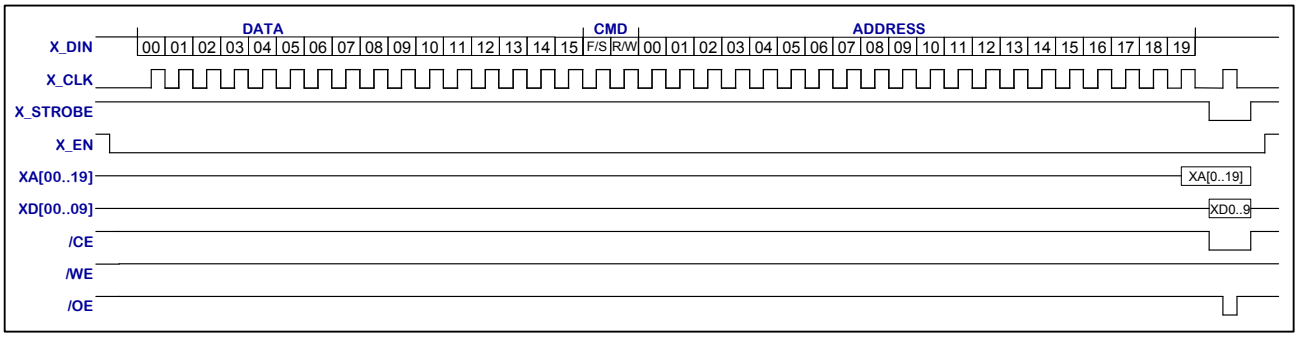
/X_BSY	I/O Data Pending in MLDL, HP41 did not yet read previous written data when high. This signal is directly connected to the H_BSY bit in the internal MLDL I/O Register for the HP41. A read from this register will reset /X_BSY. When /X_EN is asserted (MLDL2000 disabled) /X_BSY reflects the state of the FLASH device's RY-BY signal for easy detection of the end of the FLASH programming or erase cycle.	out	active high, 3.3 V
PE	Program Enable, input for CPLD to enable programming of the CPLD through the external interface instead of having to use the JTAG cable. PE must be low during normal operation and initialization and is high when programming through the JTAG interface is required.	in	active high, 3.3 V

When using the external JTAG interface for programming the CPLD, all I/O signals must be in the high-impedance state when the USB interface is used for programming. This can best be guaranteed to program all signals as input when the USB module is used, or to disconnect all lines. PE must driven high to enable the JTAG programming interface. PE is normally low and pulled to ground with a resistor.
For interfacing with the FLASH and SRAM 20 address bits are required, 16 bits of data are required for write operations and a 2-bit command is used. Only the 10 least significant bits are used by the HP41. Data is shifted in first, followed by the 2-bit command and finally the address as shown in the timing diagram below. In a read command data will not be used and does not have to be shifted in. Data is shifted in the IOSR (I/O Shift Register) upon the falling edge of /X_STROBE.
Assertion of /X_STROBE will cause the command to be executed and the address in the IOSR to be put on the XA lines.
The command is as follows:

Command	Function
00	Write to Flash
01	Write to SRAM
10	Read from FLASH
11	Read from SRAM

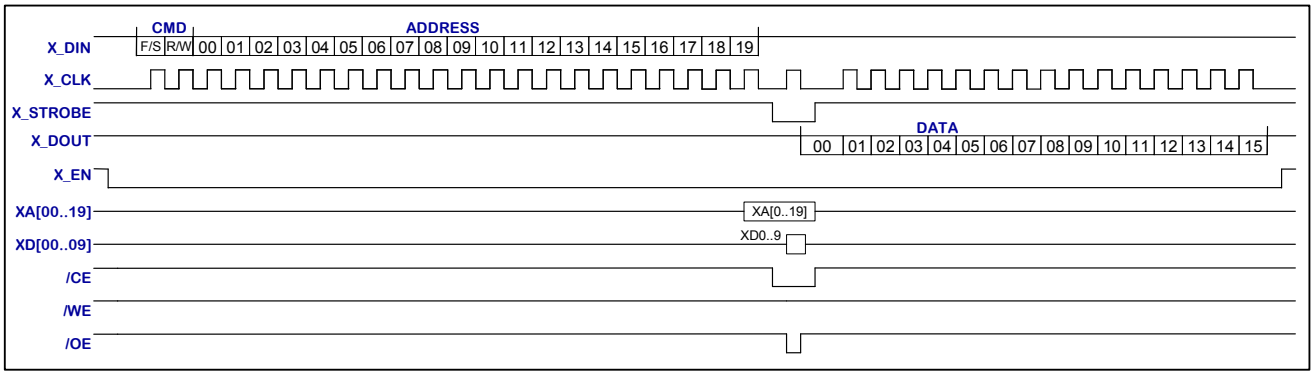
/X_EN is used to select between two possible modes: the first one is I/O with the HP41 for use in programs running on the HP41. /X_EN should be high (de-asserted) when this mode is required. The second mode is when /X_EN is asserted, and allows direct access to the SRAM and FLASH memory from the external interface. The HP41 can not access any of the MLDL memory when /X_EN is asserted.

Memory Access mode: /X_EN asserted
This mode is used to manipulate the memory contents of FLASH and SRAM through the external interface. Note that the HP41 can no longer communicate with the MLDL2000! The I/O is implemented with a USB interface to 'bitbang' the serial interface with two possible commands: read and write. The serial bit stream is shown in the following picture for a Memory Write operation (/X_EN is low!):



Data is shifted in first, then 20 address bits. F/S indicates FLASH or SRAM to be addressed, F/S = 0 means FLASH, F/S = 1 means SRAM. W/R = 0 for a Write operation.
The assertion of /X_STROBE will activate address, data and /CE lines for the memory device, /X_CLK will assert the /OE or /WE lines of the selected device. Timing for /X_STROBE and /X_CLK should respect the requirements for the specific memory device.
A write operation to FLASH memory does not mean that data is actually written, but rather that a write cycle to the FLASH is done. Writing to FLASH is done by performing a special sequence of write (and sometimes read) operations like the one described above. Please refer to the datasheets of the FLASH memory device (AM29LV160DB) for details.
For easy aligning with 8-bit transfers 2 dummy bits and clocks may be inserted before the stream above.

The sequence for reading data from memory is below:



In this case R/W = 1 to indicate a read operation. Data will be latched in the I/O Shift Register upon the falling edge of /X_CLK when /X_STROBE is low and will be put on X_DOUT on every rising edge of /X_CLK for reading by the external interface. Note that in all cases when /X_CLK is present data will be shifted in and out, and that this data may not always be meaningful. For easy aligning with 8-bit transfers 2 dummy bits and clocks may be inserted before the stream above.

I/O interface

The I/O interface with the HP41 is used for direct data transfer between the HP41 and the outside world. The register can be used without an external command interface and offers a simple handshake mechanism. For practical purposes the I/O register is mapped in the memory space of a ROM as a 10 bits register, with the two highest bits being used for status. Using the I/O interface means that one ROM page is fully reserved for I/O use. The I/O interface is enabled by default when /X_EN is high.

IO Interface as seen from the HP41

The IO Interface is memory mapped in a ROM bank (see description of the Status Register) to allow for an easy and flexible IO Interface. Although multiple IO Banks could be mapped, these all alias to the same physical IO Interface. The IO Bank can be used as any normal ROM, and it can be put in both SRAM and FLASH under special conditions. The real I/O is enabled by registers which are embedded in the ROM memory map, and only these locations cannot be used for any HP41 code.
Within a ROM page that is mapped to an IO Bank the address x800 to x80F are reserved for I/O registers. Currently the following registers are defined:

Address	Read Operation (FETCH S&X)	Write Operation (WROM, $040)
$x800		I/O Register 0 write
$x801		I/O Register 1 write
$x802		I/O Register 2 write
$x803	I/O Register read	I/O Register 3 write
$x804		I/O Register 4 write
$x805		I/O Register 5 write
$x806		I/O Register 6 write
$x807		I/O Register 7 write
$x808	MLDL Status register	Writes are ignored
$x809	Reserved (SPI Data read)	Reserved (SPI Data write)
$x90A	Reserved (SPI Status read)	Reserved (SPI Status write)
$x80B-x80F	Reserved for future use, read values may vary	Reserved for future use, write are ignored

The remaining parts of the block can be used for regular ROM code. The registers are mapped to the I/O even when the ROM block is mapped in FLASH (using a special setting in the Settings Registers). The SPI registers are intended to be used for the MLDL2000 V2 and are reserved in V1 of the MLDL2000. Using the reserved registers may lead to unexpected results.
Note that the I/O registers can be read under other circumstances, for example when doing a ROM checksum. Since the register contents may change at any time, ROMS with embedded I/O will typically not have a valid ROM checksum.

The I/O Register
There is only one I/O register for read, this is aliased throughout the addresses $x800..$x807. The internal MLDL2000 I/O register is 16 bits wide, and when writing the 3 least significant address bits are transferred to the MLDL I/O register according to the following table:

C-register	C15	C14	C13	C12	C11	C10	C09	C08	C07	C06	C05	C04	C03	C02	C01	C00
	0	io14	io13	io12	io11	io10	io09	io08	io07	io06	io05	io04	io03	io02	io01	io00
	LSB part of C Mantissa (IO register address)				C-register S&X											

The I/O Registers can only be used by ROM's which have the Settings Register bit 7 set, and it is indicated as a block in SRAM (bit 8 is set). Bit 6 in this case is not used anymore fro Write Protected, but allows the ROM block to be located in either SRAM or FLASH. This also means that it is NOT possible to write protect a block containing I/O. When mapped in FLASH, the block must be mapped in the first 64 FLASH blocks.
It is recommended to program NOP's in the I/O locations x800-x80F when creating an image of the ROM on disk.

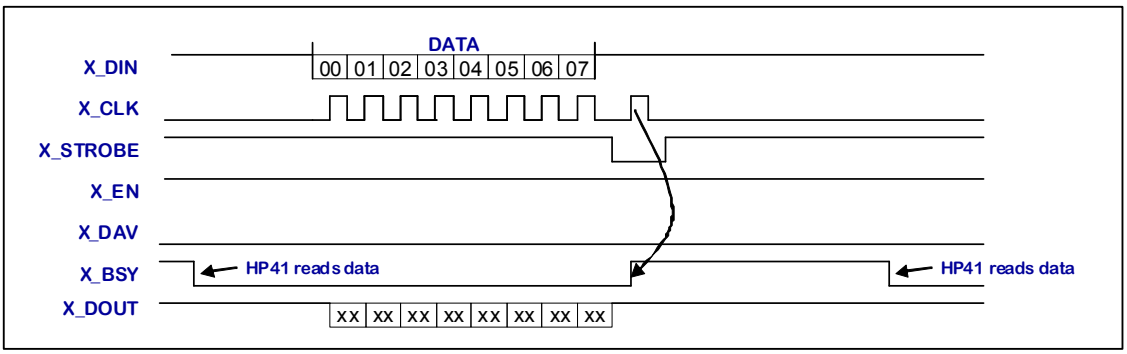
When reading the I/O Register, the meaning of the bits are the following, for example after reading with FETCH S&X:

	C-register	C09	C08	C07	C06	C05	C04	C03	C02	C01	C00
		H_BSY	H_DAV	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0

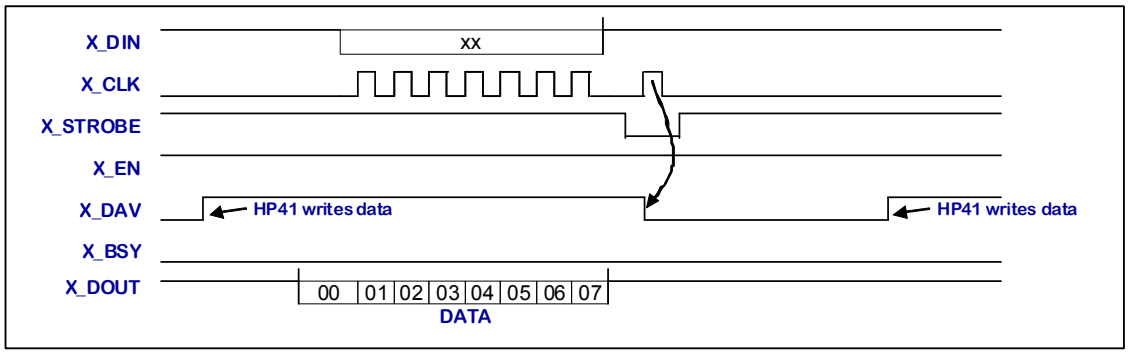
Only 8-bit data read transfers are possible with the 8 least significant bits to and from nibble 0 and 1 of the C-register. The 2 most significant bits are used to create a basic handshaking mechanism, which are connected to X_BSY (eXternal Interface Busy) and X_DAV (eXternal Interface Data Available) on the external MLDL2000 interface. The bits are called H_BSY (HP41 Busy) and H_DAV (HP41 Data Available).
Writes to the I/O register can be done with all 12 bits of the C register, S&X part. As long as the external interface did not read the I/O, reading the I/O register from the HP41 will generally return the 8 least significant bits of the written data. Bit 8 and 9 are always the H_BSY and H_DAV status bits, bits 10 and 11 read back as 0 (like in any normal FETCH S&X).
X_DAV is set immediately after a write to the I/O Register. This indicates to the external interface that data is available from the HP41. The signal will be reset upon a read by the external interface. The HP41 can use the H_DAV status bit to monitor when the external interface has read the data and new data can be written.
H_BSY is set by the external interface when writing to it. When the HP41 reads from the I/O Register the H_BSY bit is read with the data, and then immediately reset. This indicates to the external interface that the HP41 has read data. When the HP41 expects data from the external interface, it can simply read the I/O register until H_BSY is set. The data that is read with the H_BSY high signal is the new valid data. Be aware that when H_BSY is set, reading the I/O register will reset this bit.
Write and read operations should not be mixed, otherwise the handshaking mechanism may fail.
Data should only be written by the HP41if H_DAV is low, meaning that no data was be previously written to the IO Register by the HP41 or that the previous data was effectively read by the external interface. When data is written with H_DAV high the previous data will be overwritten. A read of the I/O register after a write will read the data back, with H_DAV set. Reading the I/O register when the data has been read by the external interface will return zero's in most cases, with H_DAV low.
It is important to note that the I/O register is affected by bank switching.

The I/O register external interface
The signals of the external interface are described earlier. The I/O register is a shift register with a parallel load (when writing to it from the HP41) and parallel read (when reading from the HP41). Reading and writing form the external interface is serial, in much the same way as the memory access functionality. The main difference is that only 16 data bits are transferred and that the X_BSY/X_DAV handshaking mechanism is involved.
The MLDL2000 can not distinguish between reading and writing from the external interface. It is therefore important not to mix reading and writing and to respect the status of the handshake signals. The I/O register is constantly active when the MLDL2000 is enabled (X_EN signal high).

Below is a timing diagram of the HP41 read / external interface write cycle with handshaking:



The next timing diagram shows the HP41 write / external interface read cycle:



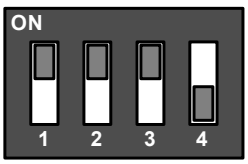
Writes from the HP41 can be done using up to 16 bits at a time, the contents coming from C S&X and part of C M. The most significant bit will always be 0. The external interface may read as many bits as required, the X_DAV signal is reset upon assertion of the X_CLK while X_STROBE is low.
Read and write I/O operations cannot be done at the same time, and the user has to ensure no writing to the HP41 takes place while X_DAV is asserted. This will result in loss of data.

The MLDL Status Register
The MLDL Status register is provided to be able to get information from the MLDL2000. Writes to the register can be used to control the MLDL2000 to a certain extent
When reading the MLDL Status Register, the meaning of the bits are the following, for example after reading with FETCH S&X:

	C-register	C09	C08	C07	C06	C05	C04	C03	C02	C01	C00
		H_BSY	H_DAV	'0'	LED3	LED2	LED1	LED0	SW2	SW3	SW4

Reading this register will not change the state of the bits, so the MLDL Status register may be used to check the status of H_BSY and H_DAV without actually changing H_BSY.
Bits 0..2 read the settings of the external switch. Switch 1 cannot be read, since it disables the MLDL2000, and the register would not be accessible anyway. The ON position reads a '1'.

Switch 2 MLDL_SR ***OFF***: SR from FLASH ***ON***: SR from SRAM
Switch 3-4 SR_SET[1,0]
OFF - ***OFF*** SR Set 0, base $FFF00 (FLASH), $7FF00 (SRAM)
OFF - ***ON*** SR Set 1, base $FFF40 (FLASH), $7FF00 (SRAM)
ON - ***OFF*** SR Set 2, base $FFF80 (FLASH), $7FF00 (SRAM)
ON - ***ON*** SR Set 3, base $FFFC0 (FLASH), $7FF00 (SRAM)



Writes to the MLDL Status register controls the status of the MLDL 2000 in the following way:

	C-register	C11	C10	C09	C08	C07	C06	C05	C04	C03	C02	C01	C00
		x	x	x	x	x	SPARE3	SPARE2	SPARE1	SPARE0	x	x	x

SPARE0..3: changes the respective SPARE output. A '1' means high, a '0' is low. Note that in MLDL V1 the SPARE signals are optionally connected to LED's. Writing a '0' will turn the LED on!

MLDL2000 Hardware Design

The MLDL2000 hardware is implemented with 3 different PCB's:

1. CPLD and Memory PCB, containing the CPLD, FLASH and SRAM IC's. The memories have direct connections with the CPLD. Connections are provided for power (with a separate power input for the SRAM), I/O and programming.
2. I/O and Supply print, containing the level shifters and buffers and the power supply. The HP41 signals are buffered with a 74HC4050 buffer to convert the HP41 signals to 3.3 Volt levels. Outputs from the CPLD to the HP41 are driven with 74HCT125 3-state drivers. Currently only driving of the ISA line is supported, but signals are provided to drive DATA, SYNC and FI as well for future use. The PCB also contains the voltage regulator for the CPLD and circuitry for switching power to the CPLD and battery backup. A jumper is provided to allow different power saving modes (see the section about Power Control).
3. USB controller, which carries the FTDI USB interface chip and EEPROM for the USB interface with.

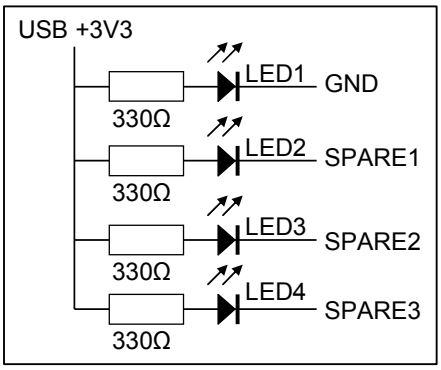
The three PCB's of the MLDL2000 are designed to fit in a Card Reader housing. The appendix contains information on the exact layout of the PCB's. The MLDL2000 User Manual has more details on the mechanical positioning of the PCB's inside the cardreader housing and how these interconnect.

Bling-bling your MLDL2000

Just for fun and based on a request from a user I have made a special version of the MLDL2000 that contains a number of LED's. Control of the LED's is supported in firmware version 1.50 in the following way:

LED1: lights up whenever the MLDL2000 is connected to the USB interface
LED2: lights up whenever the MLDL2000 is being accessed by the HP41
LED3: lights up when there is USB traffic
LED4: toggles when a WROM is done

The schematic for connecting the LED's is shown below. To prevent excessive current drain, the LED's are powered form the USB interface, so these will not work when the MLDL2000 is running from batteries. USB power is taken from the USB print, J2-1 (with the patch in place, see specifications pages 23). The SPARE signals are on J2 of the CPLD and Memory print.



In Firmware version 1.51 the LED's are controlled by writes to the MLDL Status Register.

Note that the support for LED's may be removed from future firmware versions.

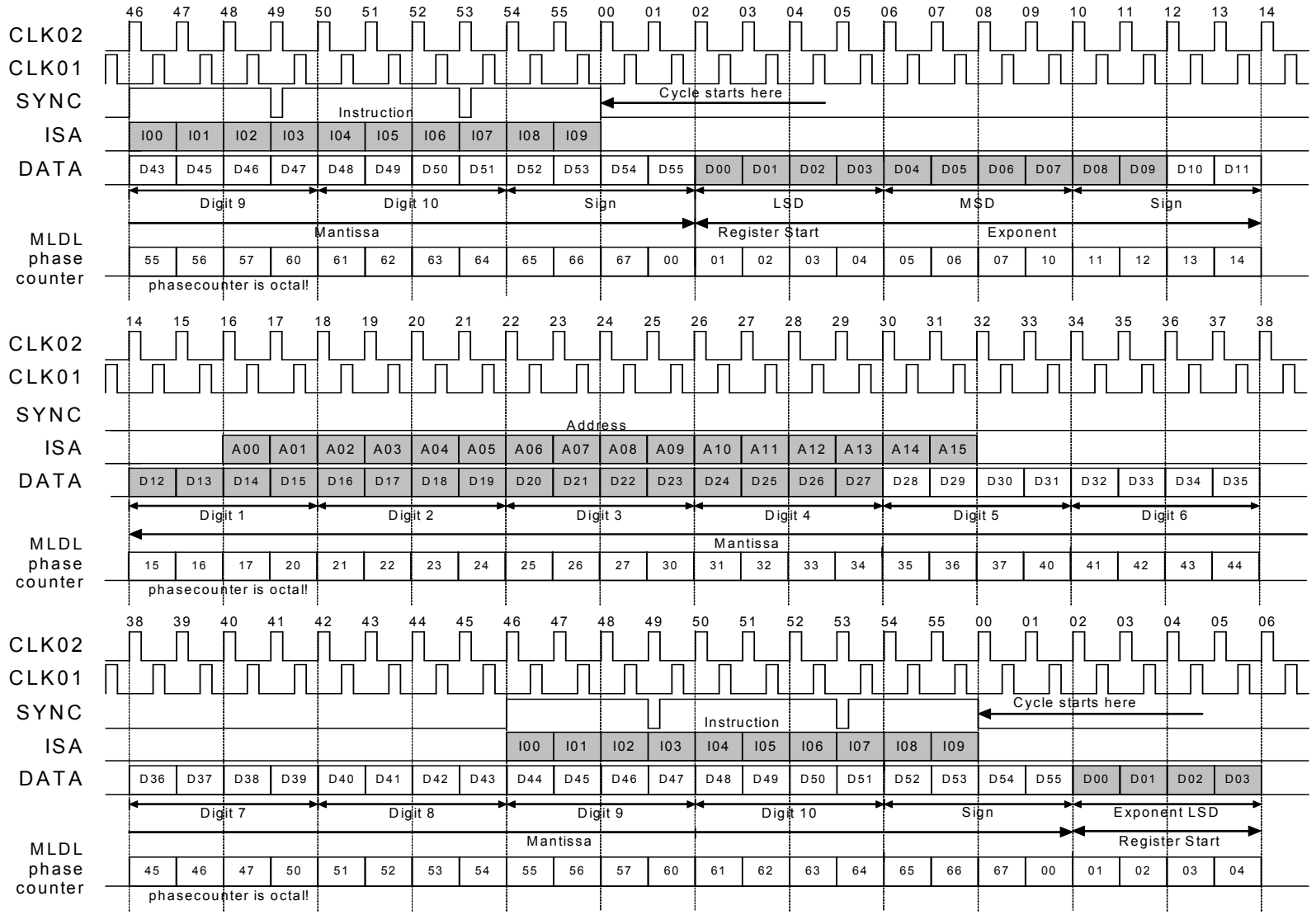
Power Control

The CPLD and memories all require 3.3V as their supply voltage. This is achieved for the CPLD and FLASH memory by using a 3.3V LDO regulator that is powered by V+ from the HP41. The SRAM is powered by the V+ voltage directly through a resistor network. External power from USB is connected to the regulator by a protection diode in order to power the MLDL when it is disconnected from the HP41. It is good practice however to save the SRAM contents on a computer when the MLDL2000 is not used for a prolonged period. It is recommended to use a lithium battery or goldcap to power the SRAM for prolonged periods of time when the MLDL2000 is not connected to the HP41. An additional external power input is connected (without protection diodes!) to the HP41 battery line to power the HP41 from an external source when no batteries are in the HP41. This HP41 battery line is not connected to anything in the MLDL2000, all power is taken from the regulated V+.
In order to save battery power, a number of measures have been taken to power off the CPLD when not in use. As this might conflict with the Enabled Bank Registers (which are stored in the CPLD) the user may control the behavior of how and when power is removed from the CPLD.
The HP41 has 3 different operating modes: *SLEEP*, *STANDBY* and *RUN*. In *RUN* mode the display is on and the CPU is active. This mode is indicated by the PWO line being driven high. The MLDL2000 CPLD is powered and the combination of HP41 and MLDL2000 will consume about 12-13 mA (measured on an HP41CV, depending on type, speed and other connected peripherals). This is about 7 mA more than without the MLDL2000.
In *STANDBY* mode the display is on, but the CPU is not running, and no clocks are generated. PWO is low and SYNC is high. By default, the CPLD will be switched off in *STANDBY* mode. If this is undesirable, for example for some bank switching applications, a jumper may be closed/opened.
In *SLEEP* mode the calculator CPU and display are both switched off and the CPLD will be powered down.
To achieve the behavior above, the LDO regulator ON/OFF input is controller by a wired OR (using a network of diodes) from PWO, SYNC and External Power. A jumper J7 is provided to select the power mode from one of the following possibilities:

1. CPLD is always powered, even when the HP41 is in SLEEP mode (jumper in position 1-2)
2. CPLD is powered in RUN and STANDBY mode (jumper in position 2-3)
3. CPLD is not powered in STANDBY mode, only in RUN mode (no jumper installed)

Testing has shown that it is not required to use J7 at all, even with HEPAX or other bankswitching ROMs.
When USB is connected the CPLD can be powered (under software control) by the USB interface even when the HP41 is switched off but the HP41 must be connected. The USB power may not always be present and usually depends on successful initialization of the connected USB peripheral.
When programming the CPLD via JTAG or USB the CPLD **must** be powered. When no external power supply (USB for example) is used, make certain that the HP41 is in such a mode that the CPLD is powered when programming the CPLD.
Disconnecting the USB cable while the HP41 is running may result in very high power consumption (up to 30mA) while the HP41 is running. This condition returns to normal when the HP41 is in *SLEEP* or *STANDBY* mode. Disconnect the USB cable only when the HP41 is turned off.

Appendix A: HP41 Timing Diagram



The landscape layout of this page is intentional

Appendix B: MLDL2000 CPLD pinout for Xilinx XCR3384 CPLD

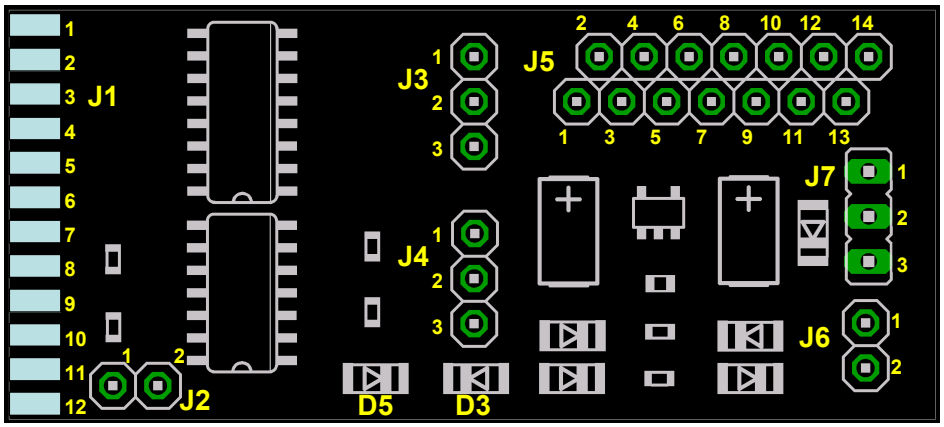
Pin name	Function	in/out	Pin number	Type
HP41 interface signals				
CLK01	HP41 Clock signal lines CLK01	in	128	active high, HP41
CLK02	HP41 Clock signal lines CLK02	in	127	active high, HP41
SYNC	HP41 SYNC Signal	in	7	active high, HP41
SYNC_OUT	HP41 SYNC Signal, used for output through driver [1]	out	143	active high, 3.3 V
SYNC_OE	Output Enable for controlling SYNC Output Driver [1]	out	142	active low, 3.3 V
ISA	HP41 ISA Signal, used for input	in	2	active high, HP41
ISA_OUT	HP41 ISA Signal, used for output through driver	out	5	active high, 3.3 V
/ISA_OE	Output Enable for controlling ISA Output Driver	out	6	active low, 3.3 V
DATA	HP41 DATA Signal, used for input	in	1	active high, HP41
DATA_OUT	HP41 DATA Signal, used for output through driver [1]	out	141	active high, 3.3 V
/DATA_OE	Output Enable for controlling DATA Output Driver [1]	out	140	active low, 3.3 V
FI_OUT	HP41 Peripheral Flag Input [1]	out	139	active high, 3.3V
/FI_OE	Output Enable for controlling FI Output Driver [1]	out	138	active low, 3.3 V
PWO	HP41 PWO Signal	in	4	active high, HP41
Memory interface signals				
XA[0..19]	External Address bus to FLASH and SRAM , 20 address lines. XA19 is not connected to the SRAM [10]	out	54 53 49 48 47 46 45 44 37 29 30 31 32 34 35 36 74 43 42 38	tristate, 3.3 V
XD[0..15]	External Data bus to FLASH and SRAM, 16 lines. [9]	in/out	60 62 65 67 69 71 80 78 61 63 66 68 70 72 79 77	tristate, 3.3 V
/FLASH_CE	FLASH Memory Chip Enable	out	55	active low, 3.3 V
/FLASH_OE	FLASH Memory Output Enable	out	56	active low, 3.3 V
/FLASH_WE	FLASH Memory Write Enable	out	39	active low, 3.3 V
/SRAM_CE1	SRAM Memory Chip Enable 1	out	82	active low, 3.3 V
SRAM_CE2	SRAM Memory Chip Enable 2	out	28	active low, 3.3 V
/SRAM_OE	SRAM Memory Output Enable	out	84	active low, 3.3 V
/SRAM_WE	SRAM Memory Write Enable	out	27	active low, 3.3 V
MEM_RESET	FLASH Reset Signal, connected but not used [4] [8]	out	40	active low, 3.3 V
RY_BY	FLASH Memory Ready/Busy Signal	in	41	active low, 3.3 V
External interface signals				
/X_EN	Enable External Interface, disable MLDL	in	90	active low, 3.3 V
/X_CLK & TCK	External Clock, JTAG TCK [6]	in	86	active high, 3.3 V
X_DIN & TDI	External Data Input, JTAG TDI [6]	in	131	active high, 3.3 V
X_DOUT & TDO	External Data Output, JTAG TDO [6]	out	121	active high, 3.3 V
/X_STROBE & TMS	External Data Strobe, JTAG TMS [6]	in	22	active low, 3.3 V
/X_DAV	Data Available in MLDL	out	92	active low, 3.3 V
/X_BSY	Data Pending in MLDL, FLASH busy programming (X_EN low)	out	94	active low, 3.3 V
SPARE[1..4]	Spare signals [13]	In/out	97 99 103 101	tristate, 3.3 V
Utility signals				
/MLDL_SR	Settings Register from FLASH (when 1) or SRAM (when 0) [12]	in	112+113	active low, 3.3 V
/SR_SET[0..1]	Select one of 4 sets of Settings Registers	in	106 107	active low, 3.3 V
/MLDL_DIS	Disable MLDL outputs (1= disabled, 0=enabled)	in	110	active low, 3.3 V
DSR_clk	Clock input for Data Shift Register [3]	in	126	active high, 3.3 V
DSR_clk_out	Clock output for DSR, connect to DSR_clk! [3]	out	120	active high, 3.3 V
OSR_clk	Clock input for Output Shift Register [3]	in	125	active high, 3.3 V
OSR_clk_out	Clock output for OSR, connect to OSR_clk! [3]	out	122	active high, 3.3 V
Power				
GND	Ground		3 17 52 57 59 64 85 105 124 129 135	Ground
Vcc	3.3 Volt Power Supply		24 50 51 58 73 76 95 115 123 130 144	3.3 V supply
JTAG signals				
PE TDI TDO TCK TMS	JTAG signals, also used for External Interface Signals! [6]	in/out	33 131 121 86 22	JTAG
Unused	8 9 10 11 12 14 15 16 23 25 26 75 81 83 87 88 89 90 91 92 93 94 96 97 98 99 106 107 108* 109* 110 113 114 116 117 118 119 132 133 134 136 137 [4] [11]			

Notes:

1. The outputs and output enables for DATA, SYNC and FI are provided for possible future use and are not functional.
2. Signals preceded with the "/" symbol are active low.
3. Signals DSR_clk and OSR_clk **must** be connected to DSR_clk_out and OSR_clk_out respectively. This is done to be able to use the global clock nets and save routing resources in the CPLD.
4. All defined but unused signals should be connected to a defined level or pulled up or down. All input signals (except the global clocks) have internal weak pull-up resistors except where noted. The resistor value is not guaranteed and typically ranges from 100k-200k.
5. CPLD inputs are 5 Volt tolerant, input voltages should not exceed 5.5 Volt or any of the other maximum ratings. HP41 signal levels are usually between 6 and 6.5 Volt. This is also valid for the External I/O interface signals. Be extremely careful when using direct unbuffered cable connections for example to a PC. It is strongly recommended to only use the supplied USB print in combination with the CPLD print.
6. JTAG signals are use for device programming and the External I/O Interface. The PE pin must be pulled low during normal operation and driven high for device programming. When using an external JTAG programmer while the USB interface is connected all signals on the USB interface must be programmed as input, except PE, which must be driven high during JTAG programming.
7. The signals BYTE_MODE of the FLASH device and SRAM_LB and SRAM_UB of the SRAM device are not connected to the CPLD but connected to GND.
8. MEM_RESET is connected but not used in the CPLD. The signals are set as inputs and pulled up internally.
9. XD[10.15] are connected to the CPLD and used for testing and the I/O interface, the HP41 does not use the extra data bits.
10. XA19 is not connected to the SRAM. Some MLDL2000 units have a different memory IC where A19 is connected to A19 of the FLASH memory by a hand soldered jumper wire. These units have serial numbers below 2050. This extra memory is currently not supported on the MLDL2000.
11. Pin numbers 108 and 109 are not connected inside the CPLD.
12. Due to an error in the layout MLDL_SR is now on pin 112. This is realized by soldering a jumper wire between pin 109 and 112. For ease of soldering, pin 112 and 113 may be connected by a bit of solder.
13. The SPARE[1..4] signals may be used to connect LED's, see this specification for more details.

Appendix C: MLDL2000 Connections and PCB

I/O and Supply PCB



J1: HP41 module connector
1. CLK01
2. CLK02
3. SYNC
4. ISA
5. F1
6. DATA
7. PWO
8. GND
9. B3
10. V+
11. B4
12. BAT

J4: Power for USB interface
1. +3.3V
2. +5V USB power
3. GND

J5: CPLD Interface
1. CLK02
2. CLK01
3. F1 Output Enable
4. DATA Output Enable
5. F1 Out
6. SYNC Output Enable
7. DATA Out
8. SYNC Out
9. PWO input
10. ISA input
11. DATA input
12. ISA Out
13. ISA Output Enable
14. SYNC input

J2: Auxiliary Power
For connecting external power
1. HP41 Battery
2. GND

J3: Power for CPLD/Memory
1. +3.3V
2. SRAM power
3. GND

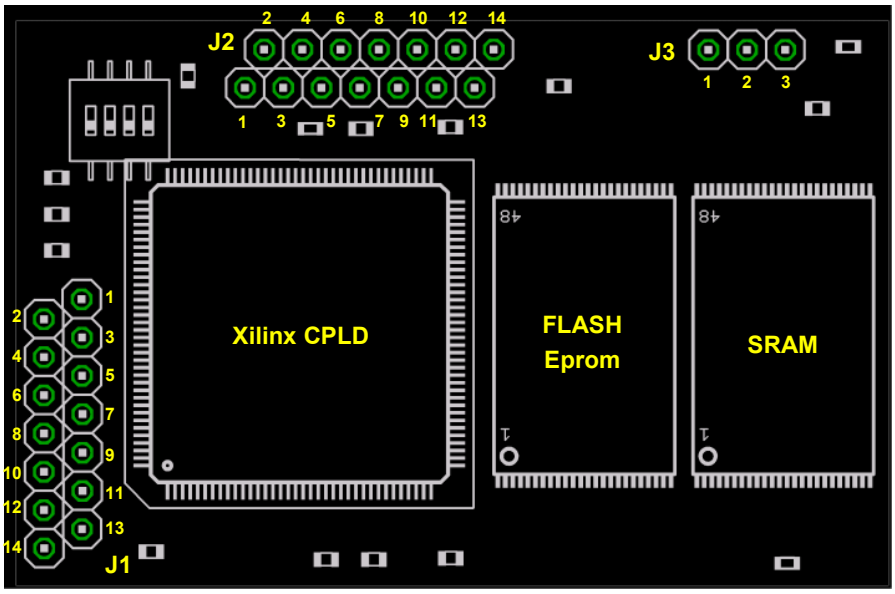
J6: Battery Backup Power
For external backup battery
1. GND
2. SRAM Backup

J7: CPLD Power mode jumper
1-2 connected:
CPLD always powered
2-3 connected:
CPLD powered in RUN and STDBY modes
no connections:
CPLD powered only in RUN mode

J1 Straight connection to the HP41 and is designed to solder a module connector to its pads
J2 Connection for an external DC power source. Note that pin 1 has a direct connection to the HP41 BAT signal and is not protected or buffered!
J3 Provides power to ***J3*** of the CPLD and Memory PCB (note that this is NOT a straight connection!)
J4 Receives power from ***J2*** on the USB Interface, pin 1, +3.3V is not used for USB and can be used for charging a goldcap
J5 Data connection to ***J1*** of the CPLD and Memory PCB
J6 Connection for SRAM battery backup, for example a Lithium cell or goldcap. The SRAM power is handled by a diode network and chooses the highest voltage from 3 sources: USB power or HP41 V+ (through a voltage divider) or pin 2 of J6. Battery backup voltage should be minimum 1.5 Volt (to guarantee data retention) and maximum 3.3 Volt.

IMPORTANT ERRATA*:* Due to an error in layout two ground planes are not connected which causes some parts of the print to have a floating ground. This must be fixed by connecting J3-pin 3 and J4-pin 3 with a small piece of wire.
Due to design error a 100k resistor should be soldered between GND (for example J3 pin 3) and the soldering pads of either D3 or D5 (cathode end) of the Beta units. In the production version this is done by placing a small SMD resistor across pins 2 and 3 of the 3.3V regulator.

CPLD and Memory PCB
The view is from the component side of the PCB, the DIP switch is mounted on the solder side!



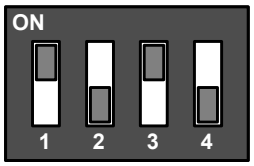
J1 Data connection to ***J5*** of the I/O and Supply PCB
J2 Data connection to ***J1*** of the USB Interface, pin 13 and 14 are available for easy connection of external JTAG programmers or I/O (power reference).
J3 Power supply to ***J3*** of the I/O and Supply PCB
SW1 Controls the use of the Settings Registers, SW1 is mounted on the opposite side from the components and is located for easy access through a cutout in the top of the cardreader housing.

J1: CPLD Interface
1. CLK02
2. CLK01
3. F1 Output Enable
4. DATA Output Enable
5. F1 Out
6. SYNC Output Enable
7. DATA Out
8. SYNC Out
9. PWO input
10. ISA input
11. DATA input
12. ISA Out
13. ISA Output Enable
14. SYNC Input

J2: I/O and JTAG Interface
1. SPARE3
2. SPARE2
3. SPARE1
4. SPARE0
5. X_BSY
6. X_DAV
7. PE
8. X_EN
9. X_STROBE / TMS (JTAG)
10. X_DIN / TDI (JTAG)
11. X_DOUT / TDO (JTAG)
12. X_CLK / TCK (JTAG)
13. GND
14. +3.3V

J3: Power for CPLD and Memory
1. GND
2. +3.3V
3. SRAM Power

SW1: Mode Switch
1. MLDL_DIS
2. MLDL Enable/Disable
3. SR_SET(0)
4. SR_SET(1)



(view from solder side!)

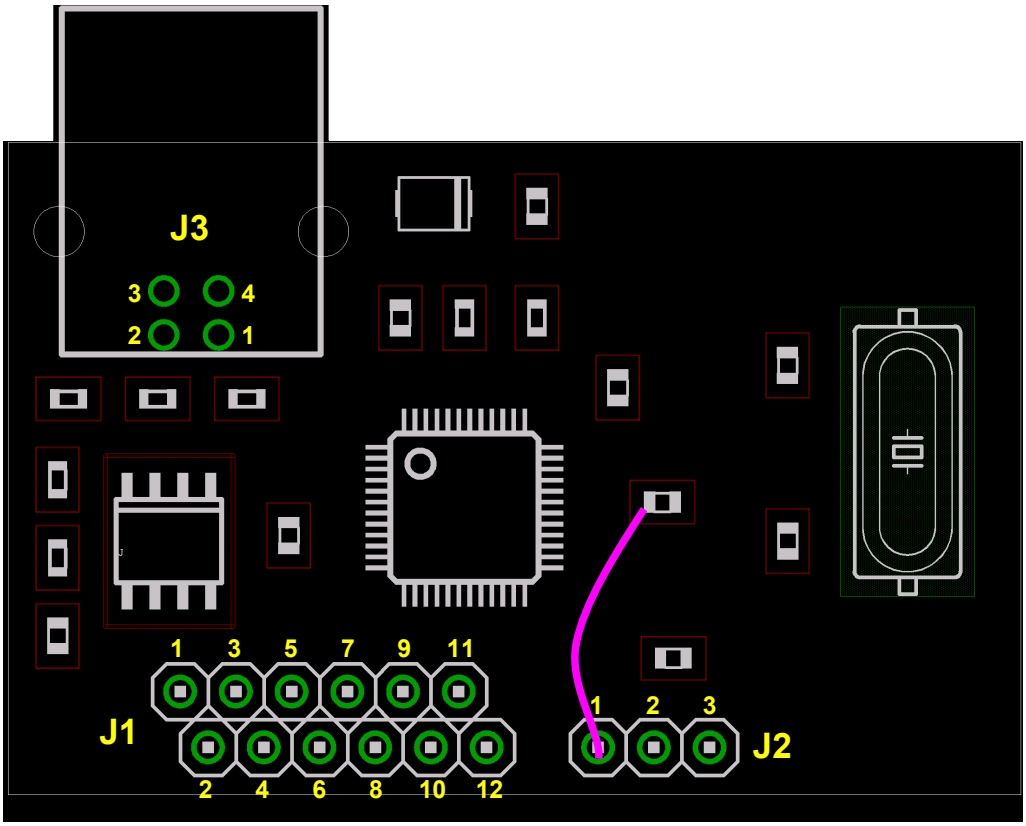
IMPORTANT ERRATA: Due to an error in layout a wire has to be soldered between pins 109 and 112/113 (112 and 113 may be connected to each other with a bit of solder to make soldering easier) of the CPLD. This is a known problem on the PCB.

NOTE: Due to component shortage some of the MLDL2000 units have a different SRAM type, with twice the capacity. To enable actual use of the capacity, a wire has been soldered between A19 of the FLASH and A19 of the SRAM. The MLDL2000 currently does not allow use of this extra capacity from the HP41. The units can be recognized by the extra wire and have a serial number under 2050.

NOTE: The SPARE pins may be used to connect LED's. Firmware is provided to use the LED's.

USB Interface PCB

The USB connector J3 is mounted on the solder side of the PCB, the view is from the component side!



J1: I/O and JTAG interface

1. CPLD_POWER
2. SPARE2
3. SPARE1
4. SPARE0
5. X_BSY
6. X_DAV
7. PE
8. X_EN
9. X_STROBE / TMS (JTAG)
10. X_DIN / TDI (JTAG)
11. X_DOUT / TDO (JTAG)
12. X_CLK / TCK (JTAG)

J2: USB Power interface

1. not used (+3.3V with patch)
2. +5V USB (do not connect!)
3. GND

J3: USB B-type connector

1. +5V USB power
2. USB Data +
3. USB Data -
4. GND

J1 Data connection to ***J2*** of the CPLD and Memory PCB. Note that pin 1 is connected to the I/O and Supply PCB J4 pin 2.
J2 Power connection to ***J4*** of the I/O and Supply PCB. Only GND is connected!
J3 USB B-type connector, positioned on the opposite side of the PCB to allow easy access through a cutout in the lower part of the cardreader housing. The connector is not soldered as standard but can be ordered separately.

NOTE: A patch is possible to have a regulated +3.3V available at J2-1. See the purple wire in the drawing above. This voltage is present when USB is connected to a PC and may be used to charge a supercap. The User Manual has a picture and more explanations. This +3.3V can supply only a very limited amount of current (max. 5 mA).

Appendix D: MLDL2000 revision history

Documentation: Specifications

Version	*Date*	*Description*	*Status*
0.1-0.8	pre 2003	various drafts	MK, done
1.0	July 2003	First public final release	MK, released
1.1	Oct 2003	Various updates and corrections, changed external interface Changed Memory Map, no U2 (de-)multiplexing anymore	MK, released
1.2	Apr 2004	Changed External Interface definitions (again) Phase Counter optimized	MK, released
1.3	May 2004	Changed state machine and memory configuration to non-multiplexed 16-bit devices	MK, released
1.4	July 2004	Added PCB layout and CPLD pin assignment for Beta release	MK, released
1.41	Jan 2005	Corrected various errors for Beta Release	MK, done
1.42	Jan 2005	Corrected various some errors, changed switch layout	MK, done
1.43	Mar 2005	Changed I/O to 16 bits for easier testing	MK, done
1.44	Oct 2005	Changed to reflect first production	MK, done
1.50	Apr 2008	Final version for MLDL2000 V1	MK, done
1.51	Sep 2008	Added I/O functions, changed state machine description, LED description	MK, in process

Documentation: User Manual

Version	*Date*	*Description*	*Status*
0.1	Apr 2004	first draft	MK, done
0.9	Jan 2005	Beta Release version	MK, done
1.1	Jul 2005	First Release version	MK, done
1.5	Apr 2008	Final version for MLDL2000 V1	MK, done

Hardware: PCB

Version	*Date*	*Description*	*Status*
0.1	Apr 2004	first draft	MK, done
0.2	Nov 2004	Beta PCB	MK, done
1.0	Jul 2005	First Release version, final PCB	MK, done

Hardware: CPLD contents

Version	*Date*	*Description*	*Status*
0.1-0.8	pre 2004	test versions	MK, done
0.99	Jan 2005	Beta release version	MK, done
1.0	Jul 2005	First Release version, same as Beta release	MK, done
1.03	Feb 2008	Corrected order of bank 2 and 3, added LED support, synthesis with Xilinx WebISE 9.2. Fixed possible instability by doing synthesis for speed instead of space.	MK, done
1.50	Apr 2008	Final release version, updated version number	MK, done
1.51	Sep 2008	Added I/O functionality, fixed stability issue, changed state machine	MK, in process

Software: M2K Manager

Version	*Date*	*Description*	*Status*
0.1	Apr 2004	test versions	MK, done
0.9	Jan 2005	Beta release version	MK, done
1.0	Jul 2005	First Release version	MK, done
1.50	Apr 2008	Final version for MLDL2000 V1. Added support for MOD files, improved error handling, updated disassembler and various other improvements	MK, done
1.51	Sep 2008	Added I/O functions, fixed some minor issues	MK, in process

Difference between current specification (V1.51) and firmware functionality (1.51) :

1. Bank switching implemented and corrected (order of bank 2 and 3). HEPAX tested and working in specific configuration.
2. I/O is implemented and tested.

Appendix E: Errata

Below is a list of know problems with a description of their behavior and how to fix the problem..

1. Power and Interface PCB ground plane
   DESCRIPTION: Two of the ground planes of the Power and Interface PCB are not connected. This will prevent the +3.3V regulator for working correctly.
   FIX: Solder a wire between J3-pin 3 and J4-pin 3 this is fixed of the Power and Interface PCB

2. +3.3V connection on USB Interface is not connected
   DESCRIPTION: J2-pin 1 of the USB Interface is not connected to anything. This was intended for VCCIO reference, which is actually connected to the internal +3.3V regulator of the FT2232C
   FIX: A fix is not required as everything seems to work just fine. See also Errata 3.

3. Increased power consumption when removing the USB cable
   DESCRIPTION: When removing the USB cable the MLDL2000 consumes too much power
   FIX: Do not connect the +5V power and GND from the USB Interface to the Power and Interface PCB. Instead use the output of USB print J1 pin1 to connect to the Power and Interface PCB J4-pin2 to control the power for the CPLD when USB is connected. In addition a 100k pull-down resistor is required on the ON/OFF input of the +3.3V regulator. In some cases the power consumption may still increase when disconnecting the USB cable while the HP41 is running. It is strongly recommended to disconnect the USB cable only while the HP41 is turned OFF.

4. SR_SET[0] switch is routed to an unused pin of the CPLD
   DESCRIPTION: SR_SET[0] switch is connected to an unused pin of the CPLD, and causes this switch not to work
   FIX: Solder a wire from pin 109 (the pin to which SR_SET[0] is routed) to pin 112 and 113, which is now designated to be used for this function.

5. In some cases the MLDL2000 may become unstable, especially after key presses.
   DESCRIPTION: The firmware contained code that could result in unpredictable behavior because of missing one clock time in the Output Shift Register. There appeared to be a dependency on the VHDL synthesis settings. This issue is visible only on a few MLDL2000 units.
   FIX: This problem was fixed with firmware update 1.51