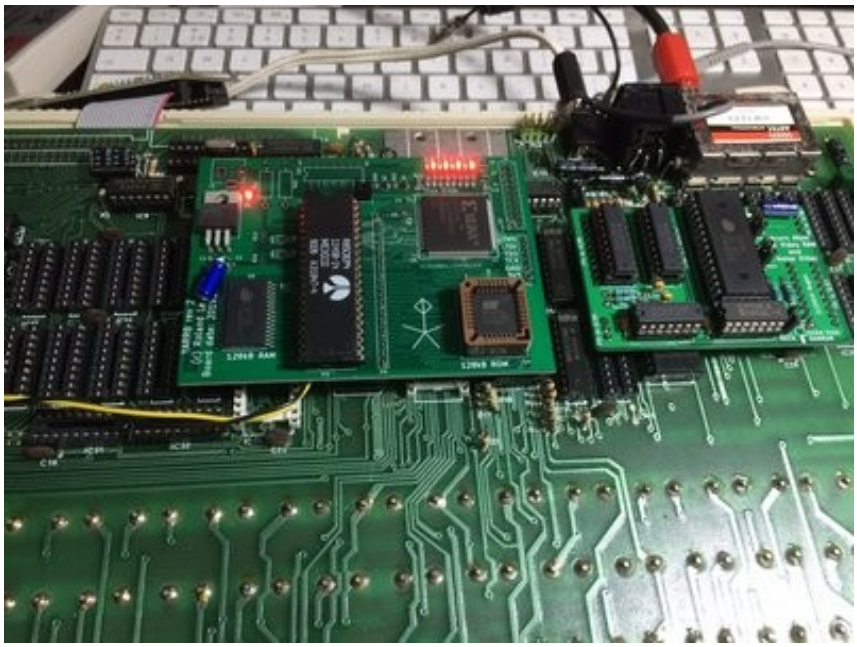
Yarrb User Manual



Author: Roland Leurs (roland@acornatom.nl)

Version: 1.0

Date: June 11, 2020

Table of contents

1. Yarrb design philosophy

YARRB = Yet Another Ram/Rom Board

For my Acorn Atom Blue I had designed another Ram/rom board. Why? Just because I want a memory extension for my classic Atoms that offer as much memory and possibilities as the Atom2015 has. So this board has 128kB of ram and 128kB of rom. It also offers three cpu[1] clock speeds: 1, 2 and 4 MHz.

In this document I will refer to the memory area #C000-#FFFF as the Operating System. This contains Atom Basic, Floating Point ROM extension, Mass Storage (AtomDOS or AtoMMC) and the MOS ROM. The area #A000-#AFFF is referred as Utility ROM(-banks).

Besides the cpu clock speed it also offers a control signal for the Atom bus buffers. In the power-on default I want to select AtoMMC rom at #Exxx but when the second ROM-bank (with AtomDOS at #Exxx) is selected, the board should disable memory at #0Axx and enable the Atom bus for accessing the 8271 FDC controller of the disc controller board[2].

Installing the board will be a matter of removing a bunch of IC's and adding two wires on the Atom main board. There is no breaking of tracks or other scary stuff on the Atom's main board so reverting the changes is quite simple. This board will be installed in the 6502 socket so it leaves the PL6/7 connectors free for more exiting purposes like a 80 column board or a matchbox copro.

Why not using other memory boards? Well, simply because they have no RAM to load an operating system and most boards have only one bank of RAM at #Axxx. Sure, you don't need to *INIT the Atom to load all the roms, but I like the flexibility of loading the rom. That's my humble opinion.

There are two versions of the Yarrb board. Both have the same features, unless specified otherwise.

1 Your cpu must support higher clock rates like a 6502A for 2 MHz operation or R65C02P4 for 4 MHz operation.
2 By modifying the CPLD logic you can also make AtomDOS as default.

2 Installing Yarrb

Since you are probably eager to use your new addition, we start here with the build and installation instructions. The changes are not destructive so you can easily undo them if you want to restore your Atom to its original state.

2.1 Building the Yarrb board

Version 1 of the board could be delivered as a complete kit. As some components (like the XC9572XL-PC44) are not manufactured any more I designed a new version. This new board (Yarrb2) has a few serious surface mounted components like a 32 pin static RAM and a 144 pin CPLD with very limited space between its pins. So these boards are for 80% assembled during fabrication. So the next text about completing the board is about Yarrb2.

When you ordered a kit then you receive the basic board with the CPLD, memory, voltage regulator, capacitors, resistors and LED's already mounted. The kit contains also a few headers, a 40p socket for the CPU and a 32p PLCC socket for the ROM. Both the CPLD and ROM are already programmed with a basic configuration and are ready for usage.

At first solder the two needle strips to the bottom side of the board. Do this before soldering the CPU socket on top of the board. Then you can solder the CPU socket and the ROM socket to the board. Also solder the 2 and 3 pin jumper pins near pins 1 and 40 of the CPU.

Before installing the board, make sure that you didn't create any short circuits between pins!

2.2 Installing the Yarrb board

The installation of both boards is exactly the same.

- Remove all the 2114 at the left side of the 6502.
- Remove IC5 (74LS30) and IC6 (74LS138)
- Remove the Atom roms (IC20, IC21 and IC24)
- Remove the 6502 and place it on your Yarrb board
- Solder a wire from pin 36 of the 6502 socket to pin 8 of the IC5 socket (this is the new buffer enable signal)
- Bend out pin 10 of IC44 (74LS393)
- Solder a wire from pin 13 of IC44 to pin 37 of the 6502 socket
- Remove LK3 (the interrupt from PL8) if you want to use the AtoMMC (optional)

After you did these simple mods, place the 6502 into the socket on the YARRB board and install YARRB in the Atom's 6502 socket. It just fits even if the heat sink is still there. As best practice I suggest to use a 40p socket with turned pins between the Atom's socket and YARRB. It makes installation easier, provides a little bit more space between the board and other components and if

you break a pin you just have to replace the socket and not the header strip :-).

2.3 Jumper settings Yarrb 1 board

This board has six jumpers:

P4	Connect Vpp This connects pin 44 to pin 1 (the Vpp input) of the EEPROM. This is only suitable for an EEPROM in the 28xxx series. Most Yarrb boards are provided with a 39SF010 which has pin 1 not connected and so this jumper is useless. If this jumper is closed then P6 should be open.
P5	Connects WR to the EEPROM. You can write protect your EEPROM by opening this jumper. Viruses cannot erase or modify your EEPROM. However, if this jumper is open you should add a 4k7 pull up resistor between pins 31 and 32 of the EEPROM (it's a fault that this resistor is not on the board).
P6	Connects A1 to pin 44 of the CPLD. If this jumper is closed then P4 should be open.
JP3	In position 1-2 this jumper connects A1 to the decoding of #BFFx. In position 2-3 this jumper makes the decoder ignoring A1.
JP4	In position 1-2 this jumper connects Phi2 of the CPU to the Atom bus. This is a normal setting for most NMOS and R65C02 CPU's. In position 2-3 this jumper makes pin 39 unconnected and connects Phi2 to Phi0 (system clock). This setting is used for the W65C02 CPU.
JP5	This jumper connects pin 1 of the CPU to GND. This is a normal setting for most NMOS and R65C02 CPU's. Leave this jumper open for W65C02 since that CPU uses pin 1 for VPB (Vector Pull, output).

So, in short:

- Leave P4 and P6 open.
- Close P5.
- For NMOS and R65C02 CPU set JP4 to position 1-2 and close JP5
- Close position 1-2 of JP3

Pin 1 of each jumper is identified by a square solder pad!

What about that A1 stuff?

In the basic design the CPLD only decodes the registers #BFFE and #BFFF. Normally this is sufficient. However, if you need one or two extra registers you can replace Vpp by A1 so you can double the number of registers in the CPLD.

Some CPU and jumper setting tests (JP4)[3]:

Synertek 6502 (7905) - works only on 1-2; on 2-3 the Atom screen memory writes are corrupted
Synertek SY6502 (8125) - works on either setting
Synertek SY6502 (8219) - works on either setting
UM6502CE - works on either setting
R65C02P4 - works on either setting

2.4 Jumper settings on Yarrb 2

Yarrb 2 has also some jumpers. For normal operation in an Atom you only have to pay attention to JP1 and JP2:

JP1	This jumper connects pin 1 of the CPU to GND. This is a normal setting for most NMOS and R65C02 CPU's. Leave this jumper open for W65C02 since that CPU uses pin 1 for VPB (Vector Pull, output).
JP2	In position 1-2 this jumper connects Phi2 of the CPU to the Atom bus. This is a normal setting for most NMOS and R65C02 CPU's. In position 2-3 this jumper makes pin 39 unconnected and connects Phi2 to Phi0 (system clock). This setting is used for the W65C02 CPU.
JP3	This jumper connects the IRQ output of the CPLD via a transistor to the IRQ input of the CPU. By default the CPLD doesn't generate interrupts so this jumper is not needed. If the board is a stand-alone system then you also want to add R13 as a pull-up for the IRQ input.
JP4	This jumper connects the NMI output of the CPLD via a transistor to the NMI input of the CPU. By default the CPLD doesn't generate interrupts so this jumper is not needed. If the board is a stand-alone system then you also want to add R16 as a pull-up for the NMI input.

So, in short:

- For NMOS and R65C02 CPU set JP2 to position 1-2 and close JP1
- Leave JP3 and JP4 open

Pin 1 of each jumper is identified by a square solder pad!

3 Information provided by David Banks. Thank you David for testing.

3 Using the Yarrb board

Basically your Yarrb board is now ready to use. Power on your Atom. Most Atoms will directly run fine, however if you still get a scrambled screen then press BREAK to see if your Atom will boot.

If your Atom clears the screen, prints “ACORN ATOM” and just hangs then you have most likely the AtoMMC ROM enabled but your AtoMMC is not connected or not responding. Just press CTRL and BREAK to see if the Atom boots. It will show the default banner and a prompt and is ready to respond to key pressing. You can enter ?#BFFE=0 to switch to Atom DOS mode which does not hang your Atom after a reset.

Assuming that your Atom works fine you can now start using it. For most day to day purposes it's fine this way. You can run almost any programme from the Atom Software Archive. If you didn't know: the menu will start automatically with SHIFT+BREAK.

To control the memory banks there are a few registers integrated in the CPLD:

#BFFF	this is the bank switching register for the eight utility roms at #Axxx; it also takes care of write protecting the RAM banks that will contain ROM code
#BFFE	this register controls the memory profiles of the Atom (see chapter 4) and it has the speed selector bits.
#BFFD	LED control register (Yarrb2 only)

3.1 Switching CPU clock frequency

You can switch between 1, 2 and 4 MHz in your software. Bits 6 and 5 of #BFFE let change you the frequency:

bit 5:	set → 4 MHz	clear →1 or 2 MHz (depending on bit 6)
bit 6:	set → 2 MHz	clear → 1 MHz

Please not that when bit 5 is set then bit 6 is ignored.

3.2 Useless LED's on Yarrb2

The Yarrb2 board has eight LED's that can be switched on and off. In most cases you build the Yarrb board into your Atom and close its case. So you won't see the LED's which makes them fairly useless. You can switch the LED's on by setting the corresponding bits in #BFFD and switch them off by clearing the bits in #BFFD.

These LED's are added in case the Yarrb2 board is used as a stand-alone processor board. See also chapter 5.

4 Memory profiles

The Yarrb board has four memory profiles:

1. The RAM/ROM board profile (the default)
2. BBC Basic
3. Atom 2k14/2k15 profile
4. Almost same as profile 3

4.1 Profile 1: Atom RAM/ROM board

In the profile the Atom behaves much like a RAM/ROM board (by Prime). It has eight utility roms in the EEPROM:

bank 0:	SDDOS	bank 4:	SALFAA
bank 1:	GAGSROM	bank 5:	Atomic Windows
bank 2:	P-CHARME	bank 6:	WEROM
bank 3:	JOSBOX (AXR-1)	bank 7:	Programmers Toolbox

In this memory profile you can choose either AtoMMC or AtomDOS enabled. The memory map in this profile is most similar to that of a standard Atom:

#0000 - #7FFF:	32 kB	Workspace and program memory
#8000 - #9FFF:	8 kB	Video RAM (top 2k not used)
#A000 - #AFFF:	8 * 4 kB	Utility ROMS (in ROM)
#B000 - #BFFF:	4 kB	I/O space
#C000 - #FFFF:	16 kB	Operating System (in ROM)

4.1.1 Using the AtomDOS file system

This memory profile enables you to use the Atom DOS ROM at #Exxx. You can select it with

```
?#BFFE=0
```

This will also disable the RAM memory at #A00 - #AFF where normally the FDC controller is located. If you don't use a disk drive (just cassette) you can enable this RAM block with

?#BFFF=2

4.1.2 Using the AtoMMC file system

This memory profile enables you to use the AtoMMC ROM at #Exxx. You can select it with

?#BFFE=6

Since this also sets bit 1 of this register the memory at #A00 - #AFF is also enabled. There's no need to disable this memory if there is no disk drive connected.

Enabling AtoMMC has another extra feature: the utility roms can be mixed. There are several games that use statements from JOSBOX, P-Charme and GAGS. Switching between those banks is done by Branquar, the Switching Operating System. Unfortunately, Branquar needs a little setup before it works properly.

When Branquar was developed most bank switching hardware had a write-only register at #BFFF. So to remember what bank was active the switching systems needed a shadow byte. Branquar uses #FD for this. However, this byte is not automatically cleared at power on so it contains another value than the register at #BFFF. The fix to synchronize both bytes is simple:

?#FD=0;?#BFFF=0

The register #BFFF on the Yarrb board is read-write but since Branquar is not adapted it won't benefit from this.

4.2 Profile 2: BBC Basic Board

This profile switches from Atom Basic to BBC Basic. This version of BBC Basic has an improved MOS ROM and supports the AtoMMC as well. Please note that you can use the syntax of BBC Basic but programmes that rely on the hardware of a BBC microcomputer will probably not run as expected. There are not many programmes written for BBC Basic on the Atom.

To start BBC Basic type: ?#BFFE=8 and press BREAK

To return to Atom Basic: ?&BFFE=6 and press BREAK

The BBC Basic memory map is:

&0000 - &07FF:	2 kB	workspace in RAM
&0800 - &5FFF:	22 kB	program memory
&6000 - &6FFF:	8 x 4 kB	ExtROM 1: extension ROMs (currently not in use)
&7000 - &7FFF:	4 kB	ExtROM 2: extension ROM, currently AtoMMC
&8000 - &9FFF:	8 kB	Video RAM (top 2k not used)
&A000 - &AFFF:	4 kB	BBC Basic (low)
&B000 - &BFFF:	4 kB	I/O space
&C000 - &EFFF:	12 kB	BBC Basic (high)
&FFFF - &FFFF:	4 kB	BBC MOS 3.0

4.3 Profile 3: Atom2k15

In the memory profiles 3 it is possible to use four banks of 16K at #4000 - #7FFF.

#0000 - #3FFF:	16 kB	Workspace and program memory
#4000 - #7FFF:	4 x 16 kB	program memory
#8000 - #9FFF:	8 kB	Video RAM (top 2k not used)
#A000 - #AFFF:	8 * 4 kB	Utility ROMS (in RAM)
#B000 - #BFFF:	4 kB	I/O space
#C000 - #FFFF:	16 kB	Operating System (in RAM)

The four banks of 16kB at #4000 - #7FFF are called eXpanded Memory Area (XMA). You can select one of those banks with bits 0 and 1 of #BFFE.

The eight utility banks at #A000 - #AFFF are selectable with bits 0, 1 and 2 of #BFFF.

Beware that you have to load the Operating System in RAM before switching to memory profile 3 or 4 otherwise your Atom will hang until power off; even BREAK won't work because there might not be an operating system (if you want to give it a try: ?#BFFE=#10). An easy way to load the utility ROMS and the Operating System in RAM is starting the Atom Software Archive menu and

press the 'R' key[4].

To protect the RAM space that hold the utilities and the operating system you can write protect those memory areas by setting bit 7 of #BFFE. A common setting of this register is #94 which stands for:

bit 7 is set: write protect on

bit 4 is set: operating system in RAM

bit 2 is set: Disk RAM is enabled, this is the memory block #A00 - #AFF.

4.4 Profile 4: Atom2k15 (non persistent)

The difference between profiles 3 and 4 is that profile 3 is preserved after BREAK and 4 is not (after BREAK profile 1 is selected so you get an Atom prompt). So if you want to experiment with your own operating system changes you can do this in profile 4 and if the Atom hangs you can reset it with BREAK without losing your work. A hangup in profile 3 can only be recovered with a power cycle and so you might lose data.

4 Only software archive V11 ((AtomSoftwareArchive_20200512_1005_V11.zip). However there is still a small bug in the file /SYS/YARRB. It does not copy the operating system into RAM. You can easily fix this by changing line 440 to LINK #3C00 and save the file back to the MMC card. This bug is fixed in version 11.1 of the archive.

5. Special usages

5.1 Yarrb2 as stand-alone CPU board

You can use Yarrb2 also as a stand-alone CPU board. For this purpose it has eight signal LED's (see also paragraph 3.2) and 19 I/O pins. Depending on your demands you may need to modify the logic in the CPLD and the software in the EEPROM. However, the full 256K of memory will be available. You can also use the CPU connector for exposing the complete 6502 bus to an extension board.

To use Yarrb2 as a stand-alone board you need to add the following components (not included):

- X1, an oscillator; frequency depending on your design and needs
- JP3 and JP4 if you want to use IRQ and NMI
- J3 external power supply jack
- J4 (2 x 10 pin header) for 16 I/O pins with power lines

5.2 Yarrb2 as System1 clone

This will be a project for autumn 2020. It's the ultimate example for using Yarrb2 as a stand-alone system.

5.3 Yarrb as sideways RAM/ROM board for the Electron

Although this is not tested yet it must be possible to use both Yarrb boards in an Electron to provide a lot of sideways RAM or ROM without using a Plus-1. This is also a project for the upcoming autumn.

Appendix 1: Yarrb and AtoMMC

For some people the relationship between the Yarrb board and the AtoMMC is not quite clear. So I will try to explain it here for you.

AtoMMC

The AtoMMC is a wonderful mass storage solution for the Acorn Atom. You can use a MMC card to store your files in a directory structure like you are used to on a DOS/Windows PC, an Apple computer or a Unix/Linux system. No hassling with disk images so it is really simple to use. The AtoMMC needs a ROM at either #Axxx (ic socket 24 on the Atom's main board) or at #Exxx on either ROM/RAM expansion board.

If you use the ROM in ic socket 24 then you need to close LK3 and enable the interrupt for the AtoMMC board. On a reset (break) of the Atom, an interrupt will be generated and the Atom kernel will initialize the AtoMMC system automatically.

If you use a ROM at #Exxx then you need a modified Atom kernel which will call the AtoMMC initialization with a JSR #E000 from the Atom's reset routine. This will work without an interrupt. In fact, you are encouraged to disable the AtoMMC interrupt to avoid a hanging Atom if no ROM is present at #Axxx.

The AtoMMC can work without a Yarrb board.

Yarrb board

The Yarrb (Yet Another RAM/ROM Board) is a modern memory extension which gives you 128k RAM and 128k ROM in your Atom. It has three main memory profiles: default Atom, BBC Basic and Atom2k15. More information about the memory map and the configuration bytes #BFFE and #BFFF is in the attached files. The Yarrb2 board has also a register at #BFFD which controls the eight leds on the board.

The Yarrb board does not require an AtoMMC to work.

AtoMMC and Yarrb

As you learned in the above text both Yarrb and AtoMMC can work independent from each other. But the combination of Yarrb and atoMMC gives you a powerful Atom as you have all the memory that you need for using the MMC, the software archive and most (modern) games. The classic games will mostly need 6k of Ram and the modern games need almost all about 32k of Ram.

The default configuration of the Yarrb board is a default Atom with the DOS ROM at #Exxx. There is however also an alternative to the DOS ROM at #Exxx and that is the AtoMMC rom at #Exxx. The control bit to toggle between those ROMS is bit 2 of #BFFE. If this bit is 0 then the DOS ROM is active, with this bit set to 1 the AtoMMC ROM is active. Then there is another option for the address space #A00 - #AFF; this area is used by the non-modified Atom DOS FDC board. But if

you don't use the DOS then you don't need this gap in your memory. So it is also configurable: bit 1 of #BFFE disables RAM at #A00-#AFF when set to 0 and enables RAM here when set to 1. So, when you use the AtoMMC you can set the bits 2 and 1 of #BFFE to 1 and have the full 32k lower memory space without a gap and the AtoMMC ROM enabled at #Exxx. And this is where the magical 6 comes from. Enabling AtoMMC + no gap is done by entering ?#BFFE=6. To enable the Atom DOS with a gap at #Axx you enter ?#BFFE=0; Those bits are "sticky" so they are not reset to 0 when the Atom is reset. They only are set to zero at a power on of the Atom and when you clear these bits in software.

In an Atom with AtoMMC installed we can tweak the CPLD on the Yarrb board to have the value 6 in register #BFFE by default which will always automatically enable the AtoMMC at power on. This can be achieved by changing the VHDL source and re-program the CPLD.

I hope this explanation makes it a bit more clear what the relationship between the AtoMMC and Yarrb is. You can have either one of them, but the combinations gives you a powerful Atom without serious modifications.

Appendix 4: known issues

There are a several known issues with the Yarrb boards. These are described here. Most issues with Yarrb boards and not working Atoms or unstable Atoms are related to the 4 MHz and PHI2 clock signals.

Unstable Atom, constantly switching into graphics mode.

Some Atoms have this problem with the Yarrb 1 boards. When no program is running the Atom goes randomly into graphics mode (CLEAR 3). This is caused by an unclean 4MHz signal coming from IC44 (74LS393) pin 13. This can be solved with a 100Ω series resistor in this wire near to pin 37 of the CPU socket. Also keep this wire as short as possible.

This resistor is integrated on the Yarrb2 board and there are no Atoms known with Yarrb2 and this problem.

AtoMMC does not work with W65C02 CPU

This is caused by different timing of this specific CPU. The problem is in the AtoMMC board that uses a latch (74LS75) but will work better with a register (74LS175). However, these are not pin compatible so not easy to replace. For more information:
https://stardot.org.uk/forums/viewtopic.php?f=44&t=18255#p253477

Atom shows banner and hangs when AtoMMC installed

This is probably due to the AtoMMC not responding. With CTRL+BREAK you can skip the initialization of the MMC and the prompt should occur. If this won't work, remove LK3 which is the interrupt line from PL8. Interrupts are not needed on a Yarrb'd Atom with MMC.

Appendix 3: Configuration bits and bytes

Mode	B7	B6	B5	B4	B3	B2	B1	B0
Atom Ram/rom	-	clk1	clk0	0	0	DskRomEn	DskRamEn	ExtRamEn
Atom BBC	-	clk1	clk0	0	1	-	ExtRamEn2	ExtRamEn1
Atom 2k15	wp	clk1	clk0	1	0	DskRamEn	xma1	xma0
Atom exp	wp	clk1	clk0	1	1	DskRamEn	xma1	xma0

EEPROM CONTENTS

```
0x00000 - Atom #A000 Bank 0
0x01000 - Atom #A000 Bank 1
0x02000 - Atom #A000 Bank 2
0x03000 - Atom #A000 Bank 3
0x04000 - Atom #A000 Bank 4
0x05000 - Atom #A000 Bank 5
0x06000 - Atom #A000 Bank 6
0x07000 - Atom #A000 Bank 7
0x08000 - BBC #6000 Bank 0 (ExtROM1)
0x09000 - BBC #6000 Bank 1 (ExtROM1)
0x0A000 - BBC #6000 Bank 2 (ExtROM1)
0x0B000 - BBC #6000 Bank 3 (ExtROM1)
0x0C000 - BBC #6000 Bank 4 (ExtROM1)
0x0D000 - BBC #6000 Bank 5 (ExtROM1)
0x0E000 - BBC #6000 Bank 6 (ExtROM1)
0x0F000 - BBC #6000 Bank 7 (ExtROM1)
```

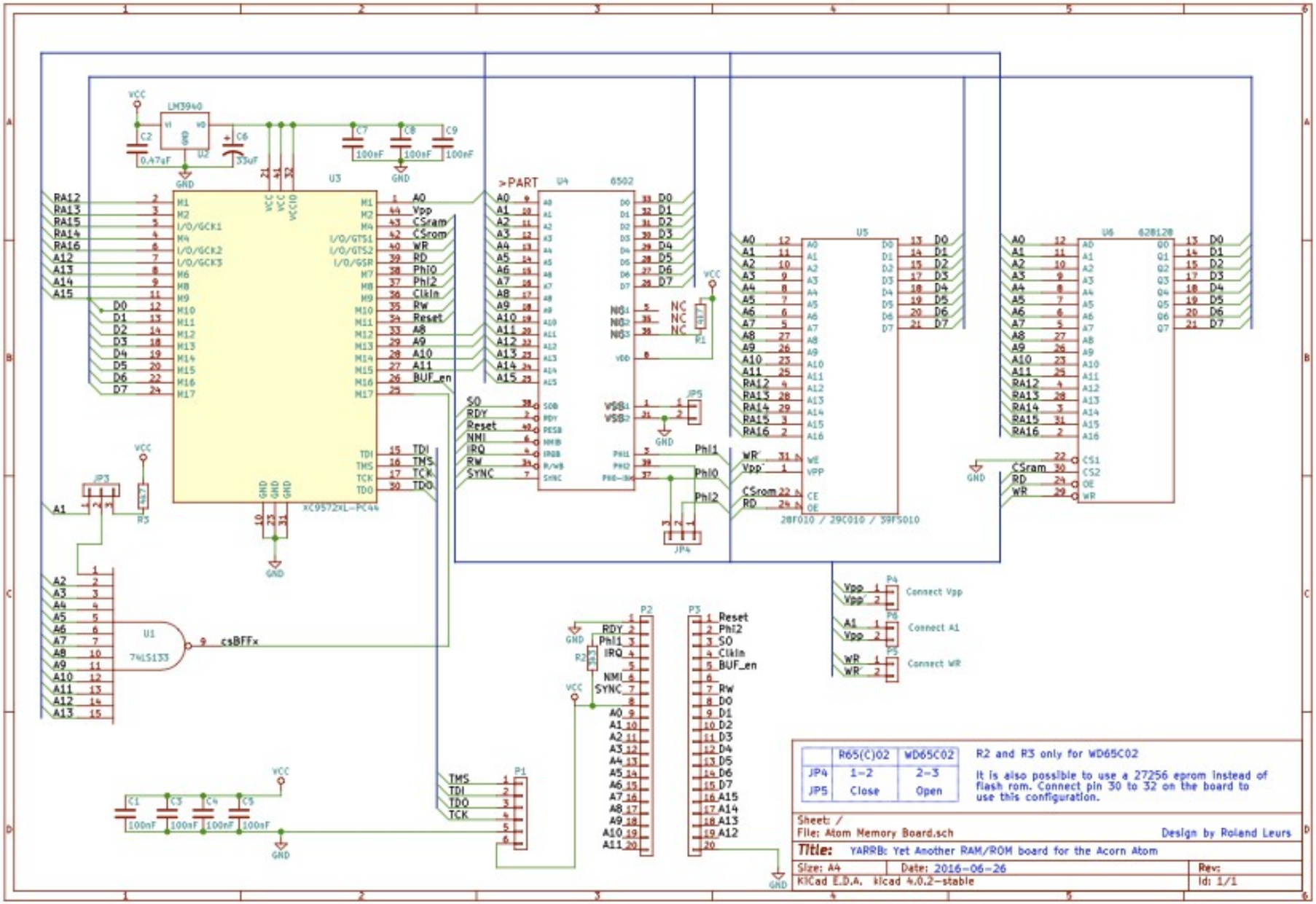
```
0x10000 - Atom Basic (DskRomEn=1)
0x11000 - Atom FP (DskRomEn=1)
0x12000 - Atom MMC (DskRomEn=1)
0x13000 - Atom Kernel (DskRomEn=1)
0x14000 - Atom Basic (DskRomEn=0)
0x15000 - Atom FP (DskRomEn=0)
0x16000 - unused
0x17000 - Atom Kernel (DskRomEn=0)
0x18000 - unused
0x19000 - BBC #7000 (ExtROM2)
0x1A000 - BBC Basic 1/4
0x1B000 - unused
0x1C000 - BBC Basic 2/4
0x1D000 - BBC Basic 3/4
0x1E000 - BBC Basic 4/4
0x1F000 - BBC MOS 3.0
```

LATCH #BFFE

```
In Atom Mode:

#BFFE Bit 0 ExtRamEn controls whether #A000 bank0
            is ROM (0) or RAM (1)
#BFFE Bit 1 DskRamEn controls whether #0A00 is
            RAM
#BFFE Bit 2 DskRomEn controls which OS set is
            selected
#BFFE Bit 3 Mode = 0 (AtomMode)
```

```
In Beeb Mode:

#BFFE Bit 0 ExtRamEn1 controls whether #6000
            bank 0 is ROM (0) or RAM (1)
#BFFE Bit 1 ExtRamEn2 controls whether #7000 is
            ROM (0) or RAM (1)
#BFFE Bit 2 unused

#BFFE Bit 3 Mode = 1 (BeebMode)
```

Appendix 4: Yarrb 1 circuit diagram



Appendix 5: Yarrb 2 circuit diagram