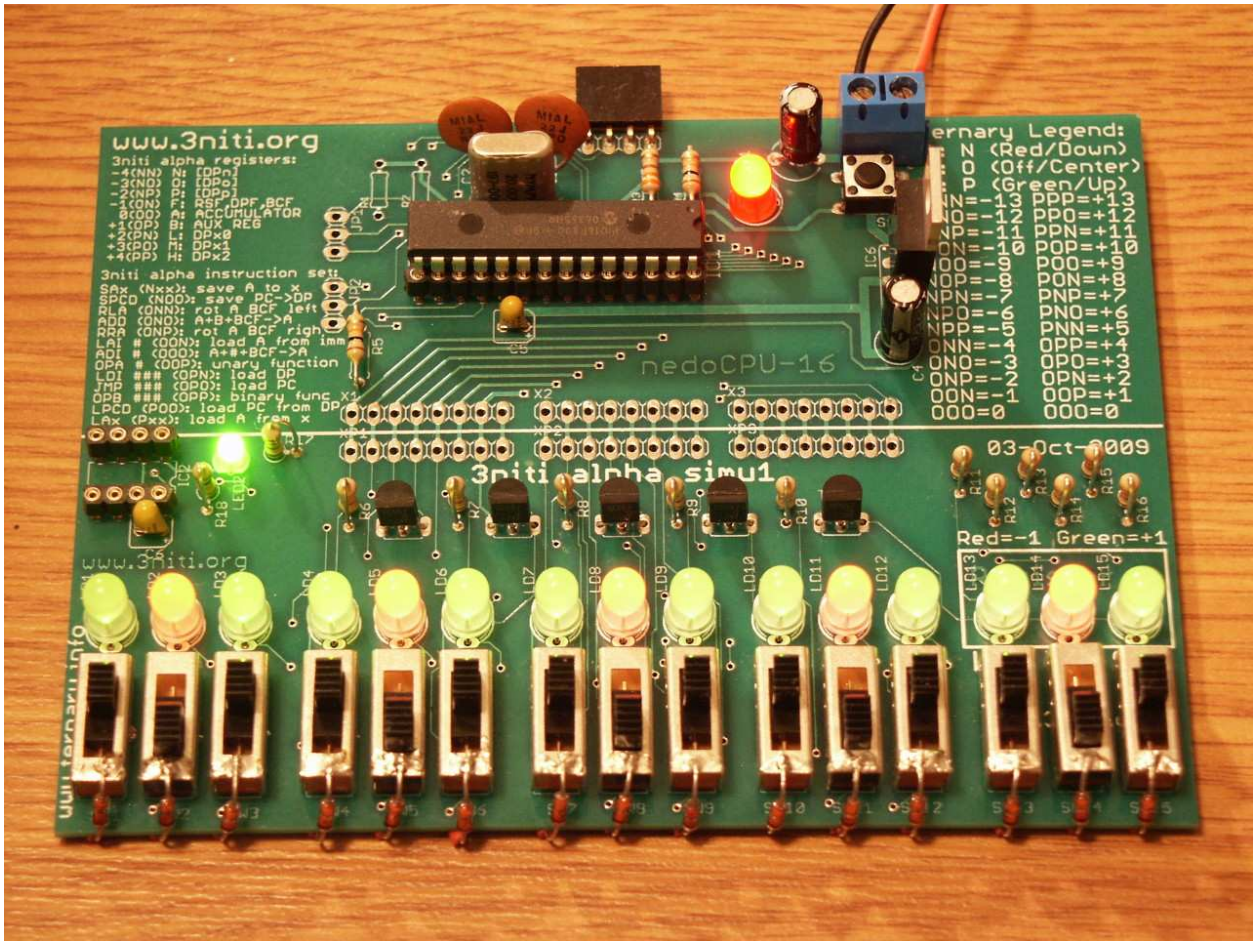
Александр Анатольевич Шабаршин
ashabarshin@gmail.com



Троичная ЭВМ "3niti alpha simu1"

Rev.A (03-Oct-2009)

www.ternary.info
2010

Об авторе: Александр Анатольевич Шабаршин родился в 1973 году в городе Троицке Челябинской области. В 1996 году закончил радиотехнический факультет Уральского Государственного Технического Университета (бывший УПИ) в городе Екатеринбурге по специальности ВМКСС (вычислительные машины, комплексы, системы и сети). В настоящее время проживает в США.

Содержание

Generated by LaTeX on June 6, 2010

Архитектура "3niti alpha"

Архитектура простой троичной ЭВМ "3niti alpha", работающей в сбалансированном троичном коде, была разработана мной в 2004. Основное отличие троичной ЭВМ от обычных двоичных состоит в том, что вычисления ведутся не над битами (принимающими одно из двух значений - 1 или 0), а над тритами (принимающими одно из трёх значений – 1, 0 или -1). Для того чтобы не путать бинарные значения с "тринарными", мы обозначим их несколько иначе – N для -1 (*negative*), O для 0 (просто похоже по написанию на 0), P для +1 (*positive*). Из трёх тритов можно составить триаду, которая может нести в себе некое целочисленное значение:

NNN	-13
NNO	-12
NNP	-11
NON	-10
NOO	-9
NOP	-8
NPN	-7
NPO	-6
NPP	-5
ONN	-4
ONO	-3
ONP	-2
OON	-1
OOO	0
OOP	+1
OPN	+2
OPO	+3
OPP	+4
PNN	+5
PNO	+6
PNP	+7
PON	+8
POO	+9
POP	+10
PPN	+11
PPO	+12
PPP	+13

Чтобы получить числовое значение из состояний тритов триады воспользуемся степенями тройки:

$$i = 3^2 \times t_2 + 3^1 \times t_1 + 3^0 \times t_0 = 9 \times t_2 + 3 \times t_1 + t_0,$$

где t_2 – числовое значение самого старшего трита триады, t_1 – среднего, t_0 – младшего. Например PPP превращается в $9 \times 1 + 3 \times 1 + 1 = 13$.

В случае "3niti alpha" размер одной ячейки памяти (а также размер каждого регистра) будет равняться одной триаде (три трита). Адреса в "3niti alpha" являются 9-тритными (или состоящими из трёх триад), что позволяет адресовать 19683 ячейки памяти (не очень много, но вполне достаточно для чего-то полезного). Память у нас будет использоватся как для хранения кода, так и для хранения данных.

Машина "3niti alpha" позволяет непосредственно адресовать 9 регистров (часть из которых являются лишь окнами для доступа к большему количеству регистров). Регистр задаётся двумя тритами и имеет однобуквенное наименование (не путайте названия некоторых регистров с буквенными значениями трита):

NN	-4	**N**	ячейки памяти с адресом DPn
NO	-3	**O**	ячейка памяти с адресом DPo
NP	-2	**P**	ячейка памяти с адресом DPp
ON	-1	**F**	регистр флага (см. ниже)
OO	0	**A**	регистр аккумулятора
OP	+1	**B**	дополнительный регистр
PN	+2	**L**	младшая триада текущего DP
PO	+3	**M**	средняя триада текущего DP
PP	+4	**H**	старшая триада текущего DP

Регистр F состоит из 3 троичных флагов: Старший трит: RSF (Result Sign Flag) - флаг знака результата Средний трит: DPF (Data Pointer Flag) - флаг указателя данных Младший трит: BCF (Borrow Carry Flag) - флаг переноса-заёма

Также существют три 9-тритовых регистра указателя данных (data pointer = DP) - DPn,

DPo, DPp, доступные через имена регистров L/M/H когда флаг DPF имеет значение N, O, P соответственно.

Теневой регистр программного счётчика (program counter = PC) напрямую недоступен - только через команды копирования между PC и текущим DP.

Полный список команд архитектуры:
NNN (-13) **SAN** - сохранить регистр A в N;
NNO (-12) **SAO** - сохранить регистр A в O;
NNP (-11) **SAP** - сохранить регистр A в P;
NON (-10) **SAF** - сохранить регистр A в F;
NOO (-9) **SPCD** - сохранить PC в текущем DP;
NOP (-8) **SAB** - сохранить регистр A в B;
NPN (-7) **SAL** - сохранить регистр A в L (младшая триада текущего DP);
NPO (-6) **SAM** - сохранить регистр A в M (средняя триада текущего DP);
NPP (-5) **SAH** - сохранить регистр A в H (старшая триада текущего DP);
ONN (-4) **RLA** - сдвинуть регистр A влево через флаг BCF;
ONO (-3) **ADD** - сложить регистр A с регистром B и флагом BCF, сохранить результат в A и флаге BCF, установить флаг знака RSF;
ONP (-2) **RRA** - сдвинуть регистр A вправо через флаг BCF;
OON (-1) **LAI #** - загрузить регистр A данными из триады идущей следом;
OOO (0) **ADI #** - сложить регистр A с данными из триады идущей следом и флагом BCF, сохранить результат в регистре A и флаге BCF, установить флаг знака RSF;
OOP (1) **OPA #** - выполнить унарную потритовую операцию над регистром A (функция устанавливается данными из триады идущей следом), установить флаг знака RSF;
OPN (2) **LDI # # #** - загрузить текущий регистр DP данными из 3 триад идущих следом (старшая, средняя, младшая);
OPO (3) **JMP # # #** - передать управление на адрес, взятый из 3 триад идущих следом (старшая, средняя, младшая);
OPP (4) **OPB # # #** - выполнить бинарную потритовую операцию над регистрами A и B, сохранить результат в регистре A (функция устанавливается данными из 3 триад идущих следом), установить флаг знака RSF;
PNN (5) **LAN** - загрузить регистр A из N;
PNO (6) **LAO** - загрузить регистр A из O;
PNP (7) **LAP** - загрузить регистр A из P;
PON (8) **LAF** - загрузить регистр A из F;
POO (9) **LPCD** - загрузить PC из текущего DP;
POP (10) **LAB** - загрузить регистр A из B;
PPN (11) **LAL** - загрузить регистр A из L (младшая триада текущего DP);
PPO (12) **LAM** - загрузить регистр A из M (средняя триада текущего DP);
PPP (13) **LAH** - загрузить регистр A из H (старшая триада текущего DP).

Специально для архитектуры "3niti alpha" был разработан человеко-машинный интерфейс, состоящий из линейки троичных (красно-зелёных) светодиодов и линейки троичных (трёх-позиционных) переключаталей. Линейка светодиодов отображает содержимое теневого регистра PC (9 светодиодов), одну триаду данных (3 светодиода) и также содержимое регистра A (3 светодиода) - всего 15 штук. Под светодиодами располагаются троичные переключатели - 9 для задания адреса (первые реализации могут иметь память, адресуемую лишь 8 тритами), 3 для задания вводмых данных и 3 переключателя управления (крайние справа):

W/R (write/read) - ввод-вывод: P (верхнее положение) - по установленному адресу записывается установленная триада, которая также отображается на светодиодах триады данных (write); O (среднее положение) - установленный адрес и установленная триада игнорируются (т.е. переключатели адреса и данных в этом режиме можно переключать), а на светодиодах триады данных отображается содержимое ячейки с адресом PC; N (нижнее положение) - установленная триада игнорируется, а на светодиодах триады данных отображается содержимое ячейки по установленному адресу (read).

I/M (interrupt/main) - выбор программы: P (верхнее положение) - прервать работу основной программы сохранив текущий адрес и запустить подпрограмму по установленному адресу (interrupt); O (среднее положение) - не препятствовать работе основной программы или ранее запущенной подпрограммы; N (нижнее положение) - прервать работу ранее запущенной подпрограммы и вернутся к основной программе по сохранённому адресу (main program).

S/G (step/go) - пошаговая отладка : P (верхнее положение) - осуществить один шаг вперед (step); O (среднее положение) - режим паузы; N (нижнее положение) - обычный прогон программы (go).

Плата "3niti alpha simu1"

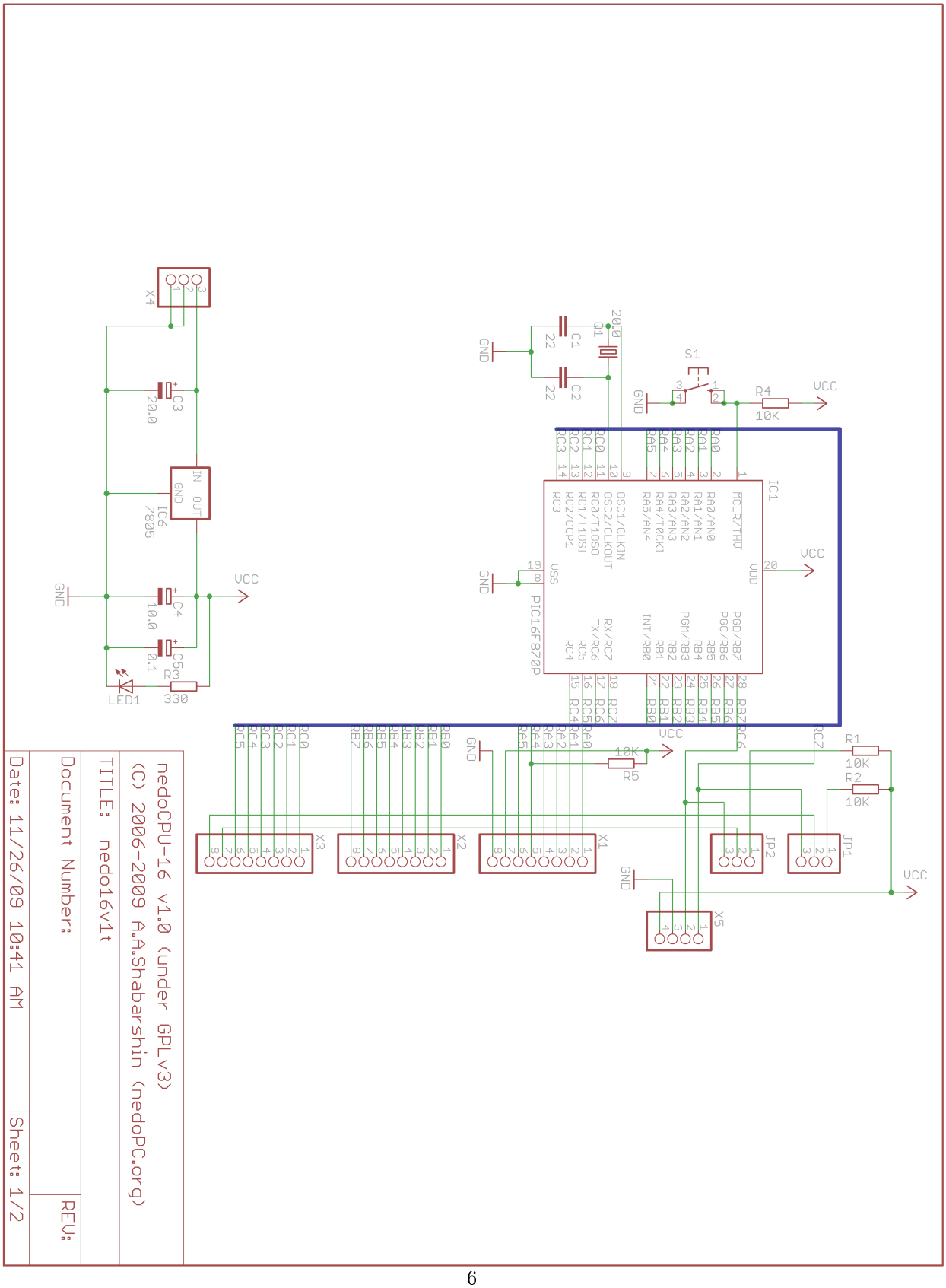
Одноплатный троичный компьютер "3niti alpha simu1" Rev.A является первым воплощением в жизнь архитектуры "3niti alpha", не являясь при этом троичным компьютером в полном смысле этого слова, потому что построен он на основе вполне двоичного микропроцессора фирмы Microchip PIC16F870, работающего на частоте 20 МГц. Для хранения троичных программ и троичных данных используется двоичная микросхема памяти с последовательным доступом фирмы Atmel AT24C32 (4K), AT24C64 (8K), AT24C128 (16K), AT24C256 (32K) или AT24C512 (64K).

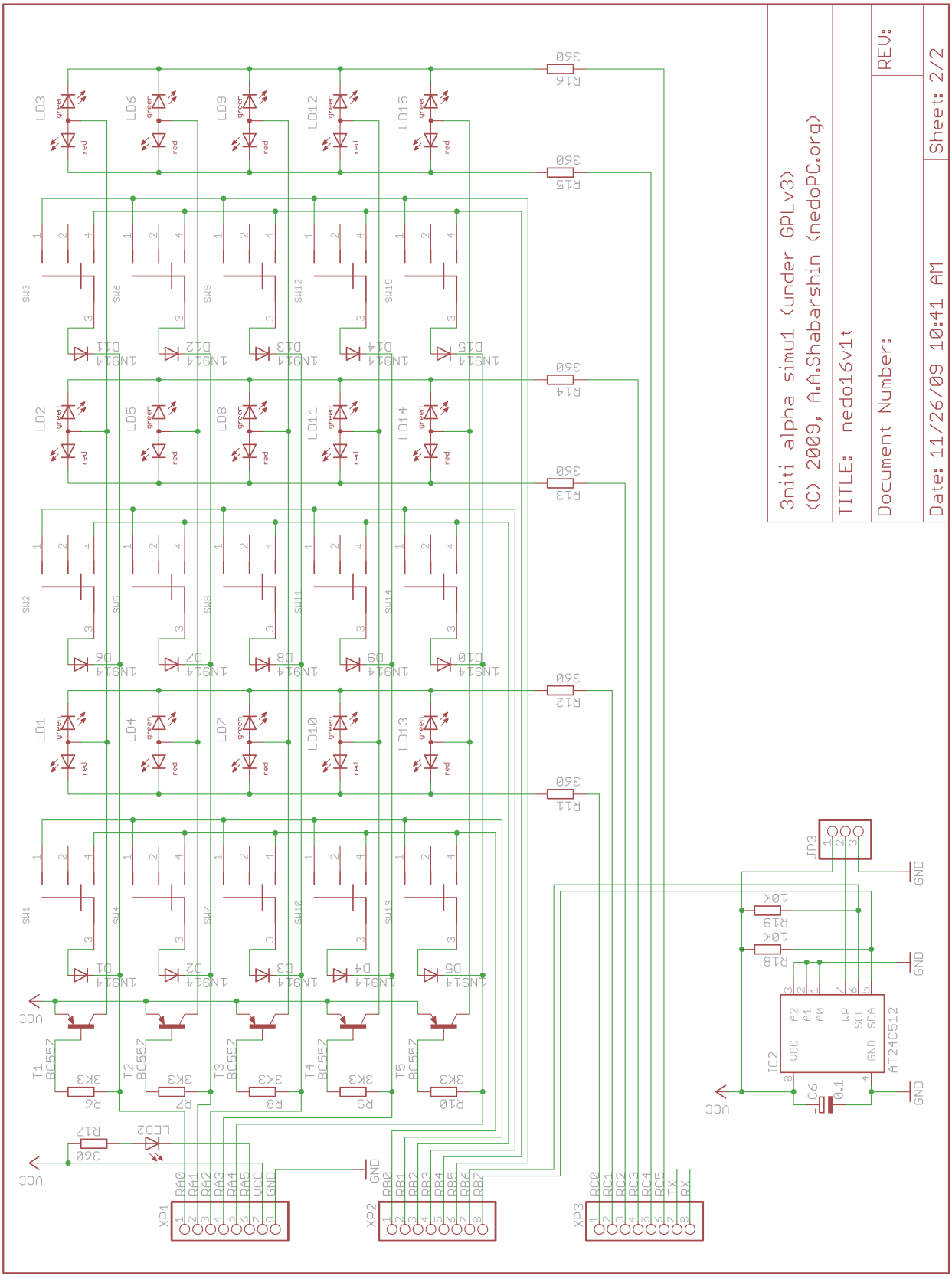
Условно "троичными" в схеме можно назвать трёх-позиционные переключатели (с четырмя выводами), используемые для ввода троичных данных, и красно-зелёные светодиоды (с тремя выводами и общим анодом), используемые для отображения троичных данных.

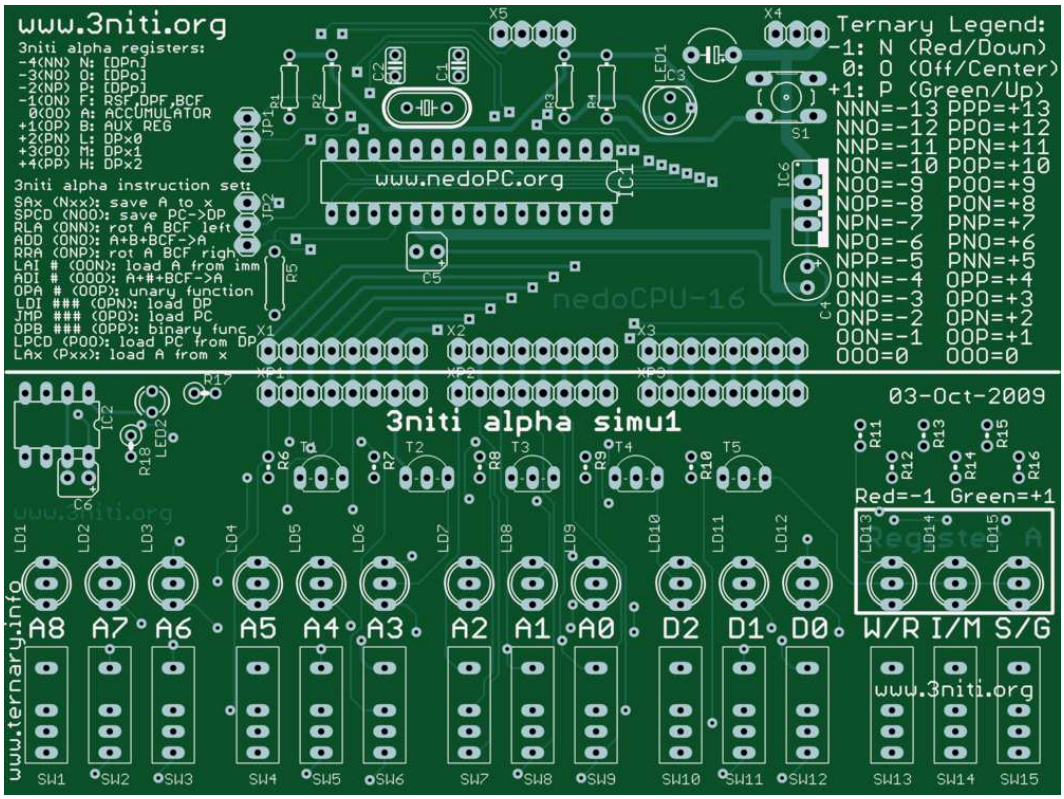
У платы также имеется последовательный TTL-порт (уровни 0-5В) для будущих расширений, который с помощью внешнего адаптера (на основе микросхемы MAX232 или аналогичной) может быть превращён в обычный COM-порт для связи с персональным компьютером, с которого можно будет вводить троичные программы большого объёма.

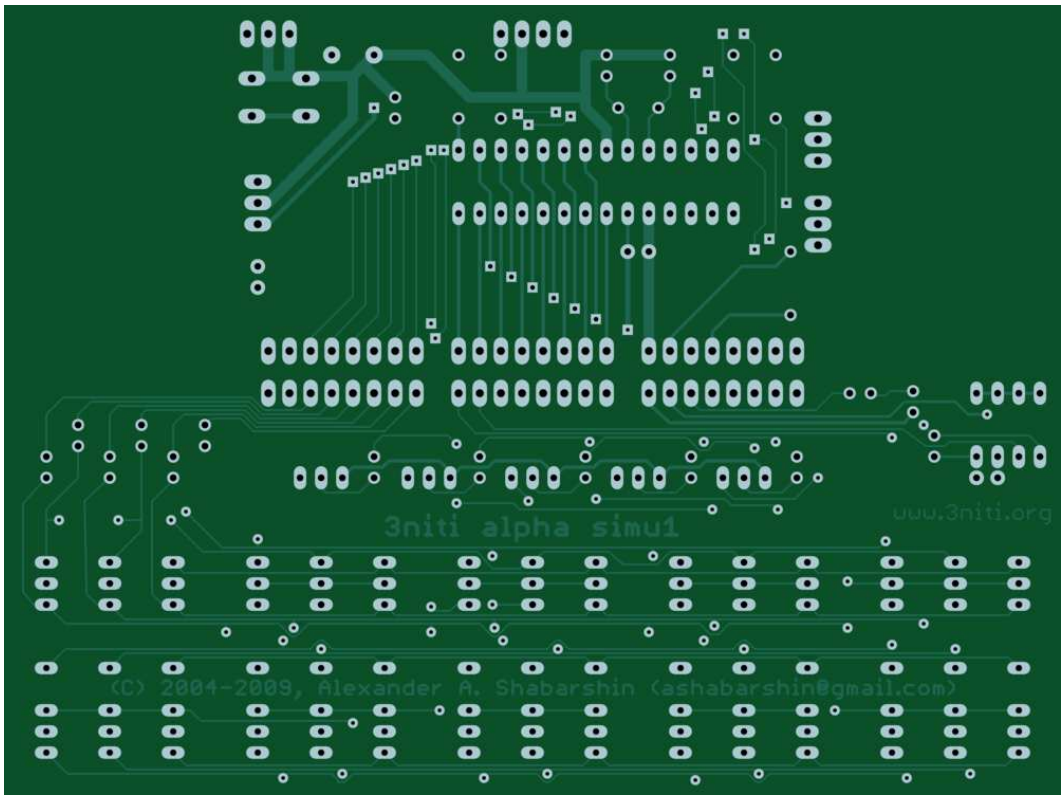
Список компонентов для сборки платы представлен в следующей таблице:

Наименование	Обозначение	Кол-во	Где купить	Аналог
переключатель SP3T	SW1...SW15	15	mouser.com 611-OS103012MU2QP1	
резистор 10 КОм	R4,R5,R18	3	jameco.com 691104	
резистор 330 Ом	R3,R11...R17	8	jameco.com 690742	
резистор 3.3 КОм	R6...R10	5	jameco.com 690988	
SIP-30	для IC1	1	jameco.com 78642	
сокет DIP-8	для IC2	1	jameco.com 390261-2	
терминал питания	X4	1	jameco.com 152347	
4-пиновый разъём	X5	1	digikey.com 609-2234-ND	
кнопка	S1	1	jameco.com 153251	
конденсатор 22пФ	C1,C2	2	jameco.com 15405	15-33 пФ
конденсатор 20мФ	C3	1	jameco.com 207811	
конденсатор 10мФ	C4	1	jameco.com 1946287	
конденсатор 0.1мФ	C5,C6	2	jameco.com 33486	
регулятор 7805	IC6	1	jameco.com 51262	К142ЕН5
кристалл 20МГц	Q1	1	mouser.com 815-AB-20-B2 1	
красный светодиод 5мм	LED1	1	mouser.com 696-SSL-LX5093HD	
зелёный светодиод 3мм	LED2	1	mouser.com 604-WP7104LGD	
R-G светодиод (CA)	LD1...LD15	15	mouser.com 604-WP59EGW/CA	
транзистор BC557	T1...T5	5	mouser.com 512-BC557BTF	КТ3107АМ
диод 1N914	D1...D15	15	mouser.com 512-1N914	КД521
микроконт. PIC16F870	IC1	1	jameco.com 246895	
послед.память 24C32	IC2	1	jameco.com 200563	


nedoCPU-16 v1.0 (under GPLv3)
(C) 2006-2009 A.A.Shabarshin (nedoPC.org)
TITLE: nedo16v1t
Document Number:
REV:
Date: 11/26/09 10:41 AM
Sheet: 1/2
PIC16F870P
IC1
IC6
7805
VCC
GND
LED1
330
10K
20.0


3niti alpha simu1 (under GPLv3)
(C) 2009, A.A.Shabarshin (nedoPC.org)
TITLE: nedo16v1t
Document Number:
REV:
Date: 11/26/09 10:41 AM
Sheet: 2/2
LD1
LD2
LD3
LD4
LD5
LD6
LD7
LD8
LD9
LD10
LD11
LD12
LD13
LD14
LD15
SW1
SW2
SW3
SW4
SW5
SW6
SW7
SW8
SW9
SW10
SW11
SW12
SW13
SW14
SW15
D1
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D12
D13
D14
D15
1N914
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15
R16
360
T1
T2
T3
T4
T5
BC557
3K3
R17
LED2
R18
R19
10K
JP3
IC2
AT24C512
C6
0.1
GND
VCC
XP1
XP2
XP3
RA0
RA1
RA2
RA3
RA4
RA5
RB0
RB1
RB2
RB3
RB4
RB5
RB6
RB7
RC0
RC1
RC2
RC3
RC4
RC5
TX
RX


www.3niti.org
3niti alpha registers:
-4(NN) N: [DPn]
-3(NO) O: [DPo]
-2(NP) P: [DPp]
-1(ON) F: RSF,DPF,BCF
0(OO) A: ACCUMULATOR
+1(OP) B: AUX REG
+2(PN) L: DPx0
+3(PO) M: DPx1
+4(PP) H: DPx2
3niti alpha instruction set:
SAx (Nxx): save A to x
SPCD (NOO): save PC->DP
RLA (ONN): rot A BCF left
ADD (ONO): A+B+BCF->A
RRA (ONP): rot A BCF right
LAI # (OON): load A from imm
ADI # (OOO): A+#+BCF->A
OPA # (OOP): unary function
LDI ### (OPN): load DP
JMP ### (OPO): load PC
OPB ### (OPP): binary func
LPCD (POO): load PC from DP
LAx (Pxx): load A from x
www.nedoPC.org
nedoCPU-16
Ternary Legend:
-1: N (Red/Down)
0: O (Off/Center)
+1: P (Green/Up)
NNN=-13 PPP=+13
NNO=-12 PPO=+12
NNP=-11 PPN=+11
NON=-10 POP=+10
NOO=-9 POO=+9
NOP=-8 PON=+8
NPN=-7 PNP=+7
NPO=-6 PNO=+6
NPP=-5 PNN=+5
ONN=-4 OPP=+4
ONO=-3 OPO=+3
ONP=-2 OPN=+2
OON=-1 OOP=+1
OOO=0 OOO=0
03-Oct-2009
3niti alpha simu1
Red=-1 Green=+1
www.ternary.info
A8 A7 A6 A5 A4 A3 A2 A1 A0 D2 D1 D0 W/R I/M S/G
www.3niti.org

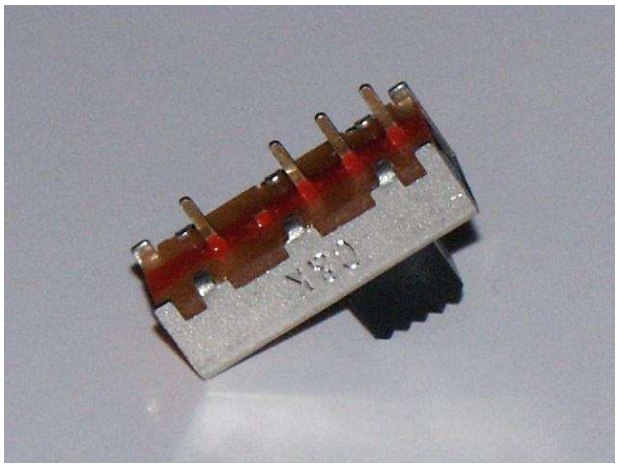

3niti alpha simu1
www.3niti.org
(C) 2004-2009, Alexander A. Shabarshin (ashabarshin@gmail.com)

Методика сборки платы Rev.A

Эта методика предназначена для сборки платы "3niti alpha simu1" из самой первой партии (так называемая "Rev.A" 03-Oct-2009) и не может быть применена ни для каких других версий платы. Данная партия плат была произведена с ошибкой (не были предусмотрены диоды для переключателей), поэтому платы Rev.A надо собирать специальным образом, описанным ниже (при этом не требуется модификация платы или прошивки). После такой специальной сборки получившаяся конструкция будет соответствовать принципиальной схеме платы "3niti alpha sumu1" Rev.B (см. страницы 3 и 4) - за исключением способа подключения светодиода, сигнализирующего об использовании внешней памяти, а также джампера JP3 (отсутствующего у Rev.A).

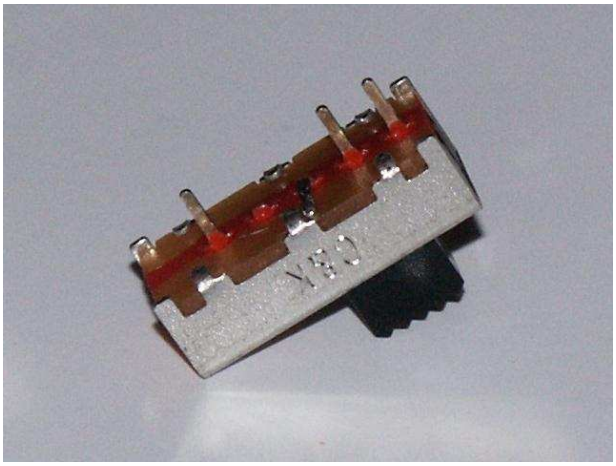
Переключатели

Для начала каждый из 15 трёх-позиционных переключателей[1] должен быть подготовлен перед установкой в плату - для этого берём переключатель так, чтобы справа оказалось больше ножек чем слева:

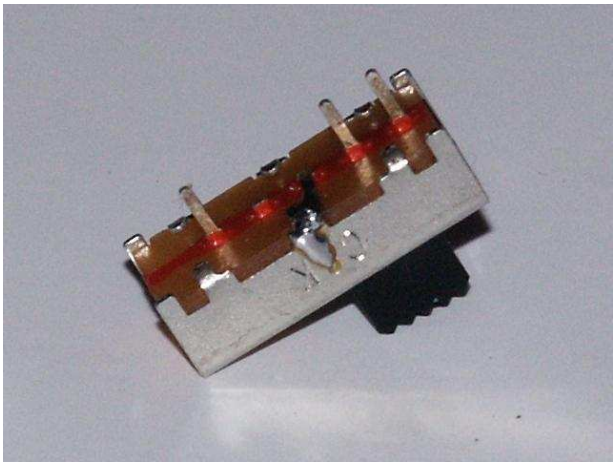


[1] mouser.com 611-OS103012MU2QP1

Далее загибаем среднюю ножку к корпусу:

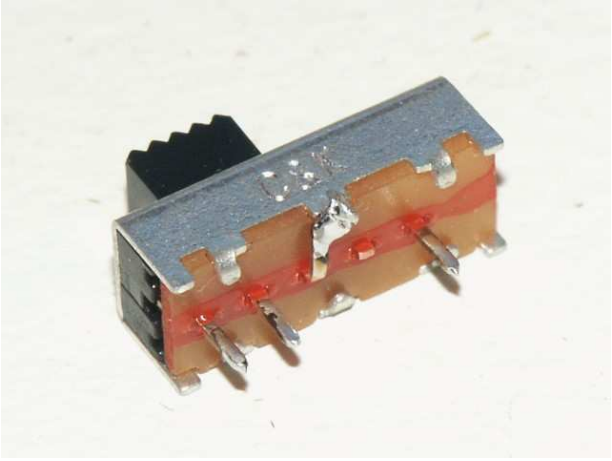


И припаиваем её к лепестку корпуса:

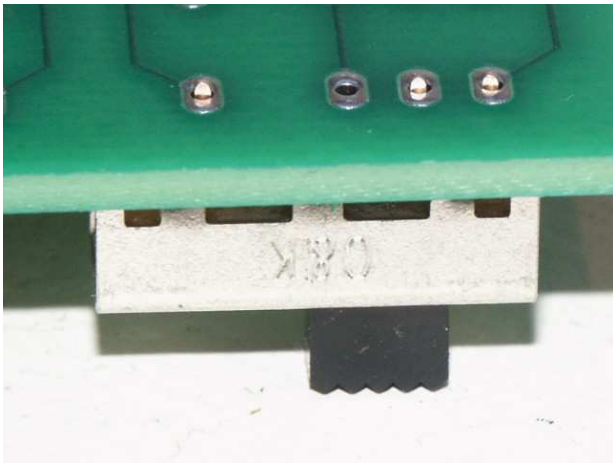


Следует сразу же убедиться, что контакт получился крепкий. Потому что если после запаивания переключателя в плату выяснится, что он не работает из-за плохого контакта в лепестке, переключатель придётся выпаивать из платы, что чревато его разрушением.

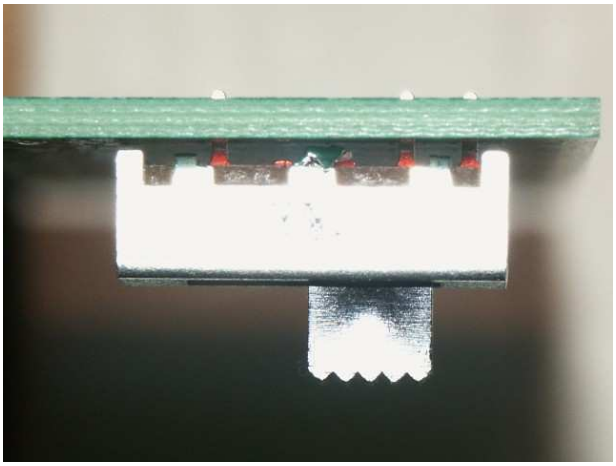
Не забываем, что оставшиеся три ножки должны быть зачищены (если нужно) и залужены:



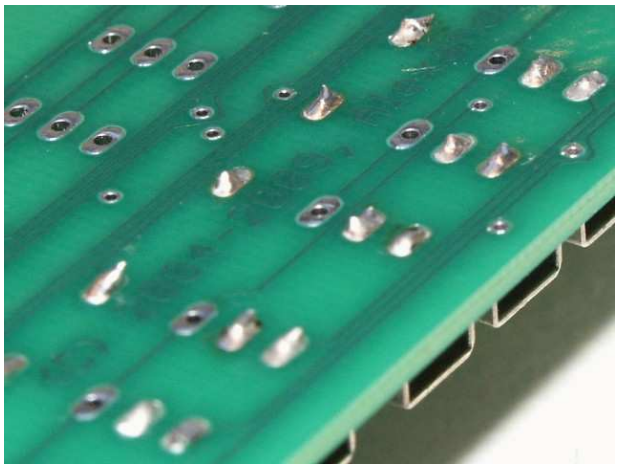
Далее вставляем каждый переключатель так, чтобы его ножки только чуть-чуть выглядывали из отверстий:



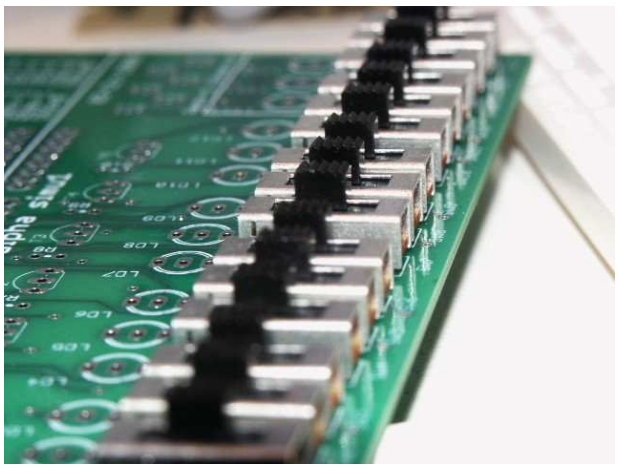
Это делается для того, чтобы средний (загнутый) контакт, не касался контактной площадки под ним (можно увидеть на просвет):



Таким образом вставляем и припаиваем в трёх местах каждый переключатель:



После этого мы получили 15 припаянных переключателей:



Следы канифоли со стороны пайки можно убрать ватной палочкой, пропитанной изопропиловым спиртом.

Резисторы

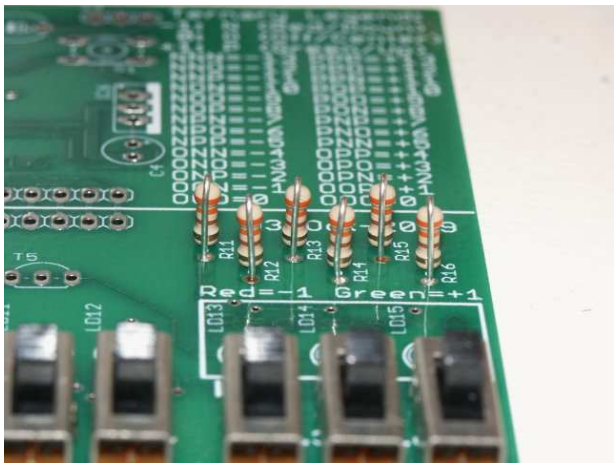
Теперь будем устанавливать резисторы - нам понадобятся 3 штуки 10 кОм, 6 штук 3.3 кОм, 8 штук 330 Ом (все 0.25 Вт):

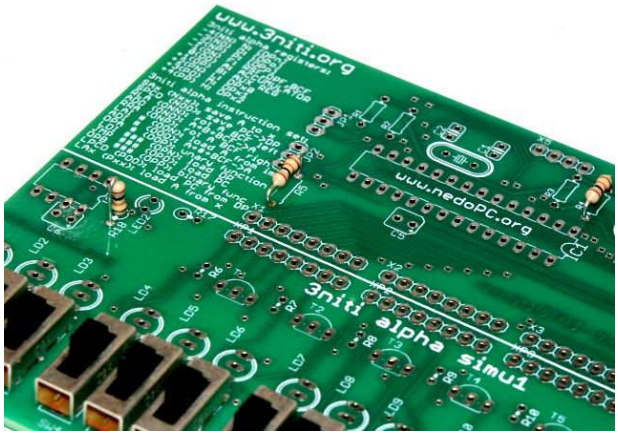
- R1,R2 - не устанавливаются ни на Rev.A, ни на Rev.B (они могут потребоваться только если верхняя плата будет отделена от нижней);
- R3 - 330 Ом (на схеме обозначено как 360);

- R4,R5 - 10 кОм;
- R6,R7,R8,R9,R10 - 3.3 кОм;
- R11,R12,R13,R14,R15,R16 - 330 Ом;
- R17 - 330 Ом;
- R18 - 10 кОм;
- R19 - на плате Rev.A отсутствует.

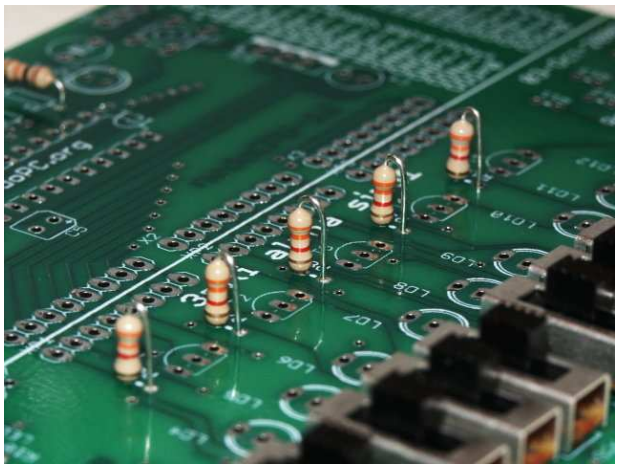
Причём R3-R5 устанавливаются горизонтально (верхняя часть платы), а R6-R18 устанавливаются вертикально (нижняя часть платы).

Итак, три резистора по 10 кОм (R4,R5,R18):



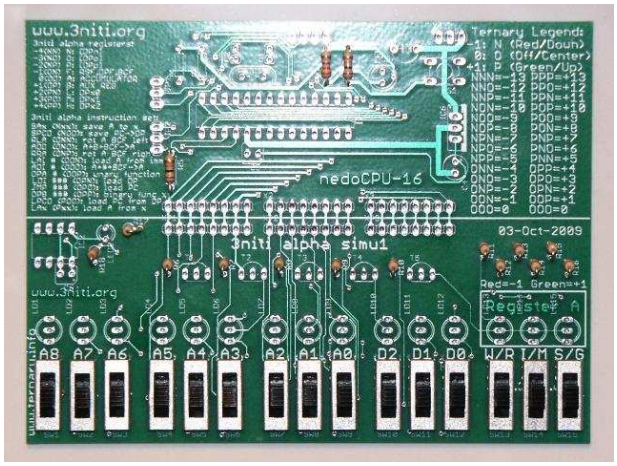


Пять резисторов по 3.3 кОм (R6,R7,R8,R9,R10) устанавливаются в центре платы:



Шесть резисторов по 330 Ом (R11,R12,R13,R14,R15,R16) устанавливаются с правого края платы:

Оставшиеся два резистора по 330 Ом устанавливаются сверху и слева (R3,R17):

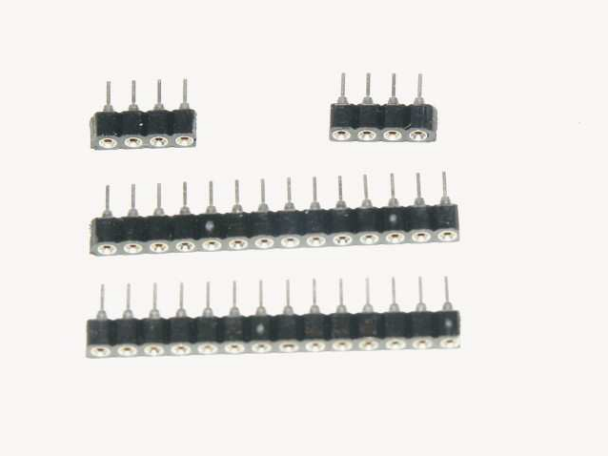


Теперь у нас все резисторы припаяны.

Коннекторы и кнопка

Теперь переходим к сокетам, коннекторам и кнопке. Джамперы JP1 и JP2 на эту плату не устанавливаются, а JP3 в Rev.A отсутствует.

Микросхемы IC1 (28 ног) и IC2 (8 ног) надо установить в соответствующие DIP-сокеты. Самый простой способ - отрезать части нужной длины от однорядного длинного сокета с шагом 0.1 дюймов:



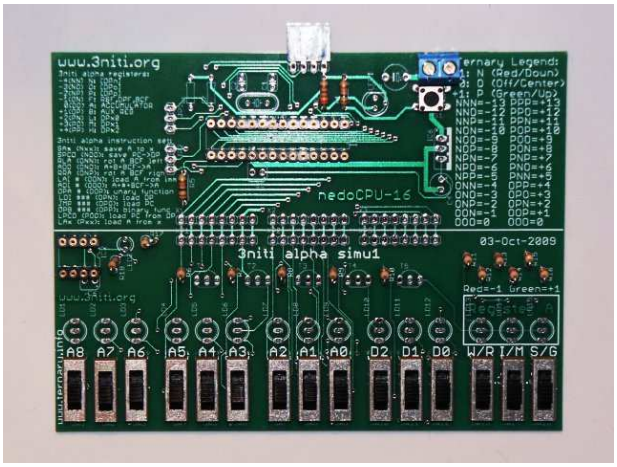
Коннекторы на схеме:

- X1,X2,X3 и XP1,XP2,XP3 - не устанавливаются (нужны только если верх и низ разделяются);
- X4 - коннектор питания (на фотографии справа);
- X5 - коннектор внешней платы последовательного порта (на фотографии слева).

Коннектор последовательного порта может быть прямым или угловым (как на фотографии):

Кнопка S1, предназначенная для перезагрузки (RESET) берётся стандартная (без неё схема тоже будет работать):

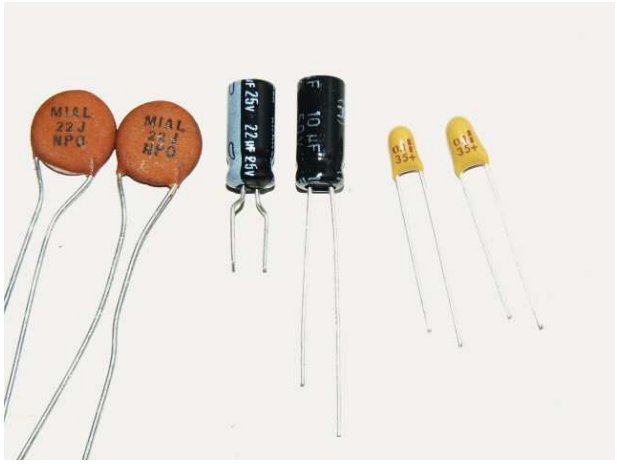
Вот так будет выглядеть плата с припаянными коннекторами и кнопкой:



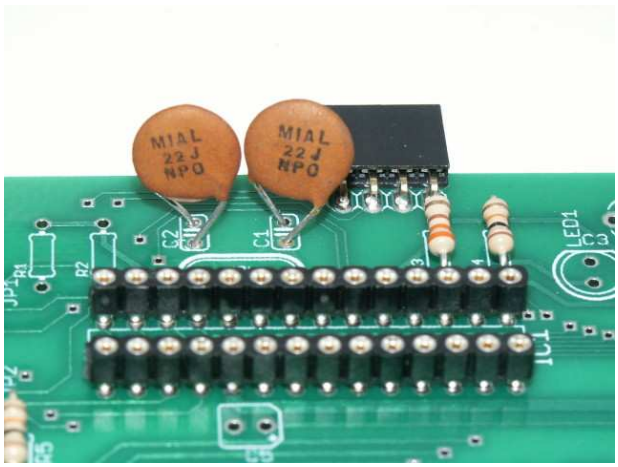
Конденсаторы

Теперь принимаемся за конденсаторы:

- C1,C2 - 22 пФ (по даташиту для 20 МГц должно быть от 15 до 33);
- C3 - 20 мкФ на напряжение более 25В (слева от регулятора напряжения - на фотографии 22 мкФ);
- C4 - 10 мкФ на напряжение 10В или выше (справа от регулятора напряжения - должно быть меньше чем C3);
- C5,C6 - 0.1 мкФ (на фотографии показаны танталовые электролитические конденсаторы на напряжение 35 вольт).

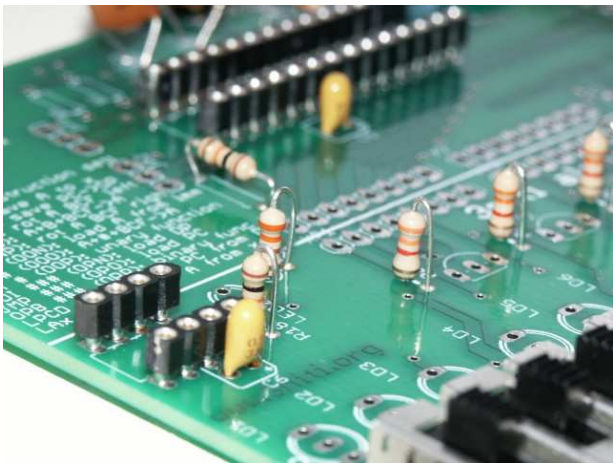


Сначала устанавливаем C1 и C2:

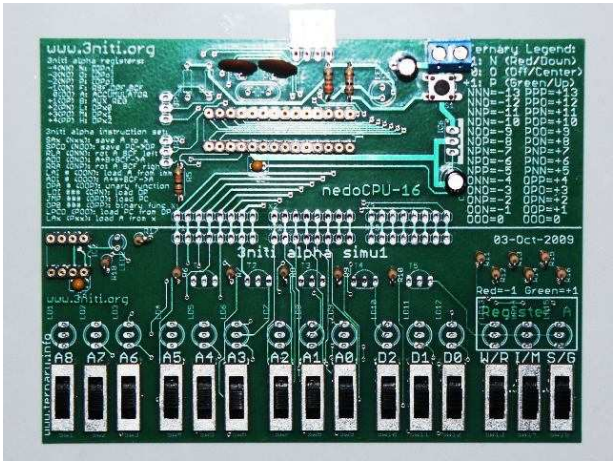


Потом C3 и C4 (следите за полярностью!):

И далее C5 и C6 (если выбраны электролитические конденсаторы, то также следим за полярностью):

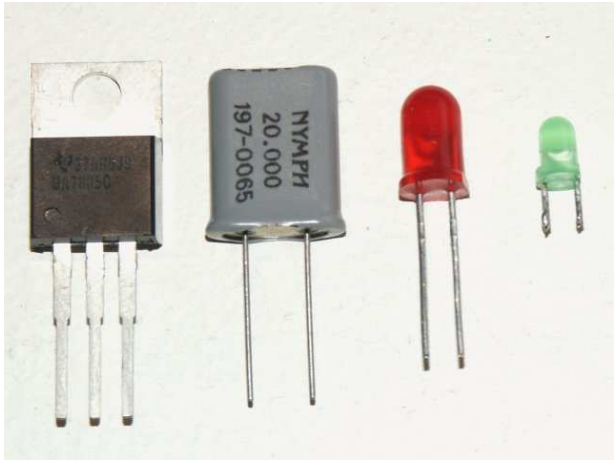


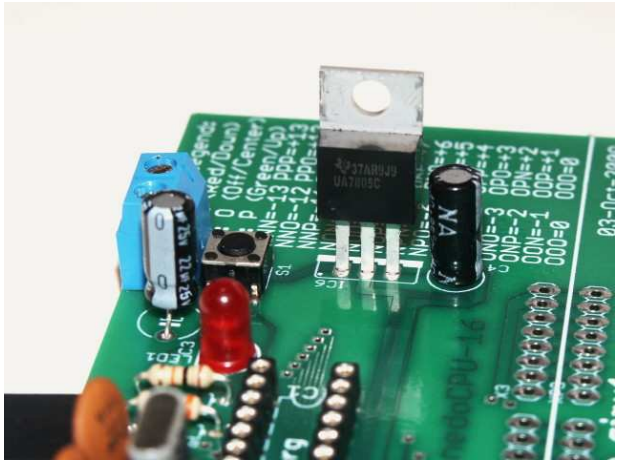
Теперь у нас установлены все конденсаторы:



Регулятор напряжения, кварц и светодиоды индикации

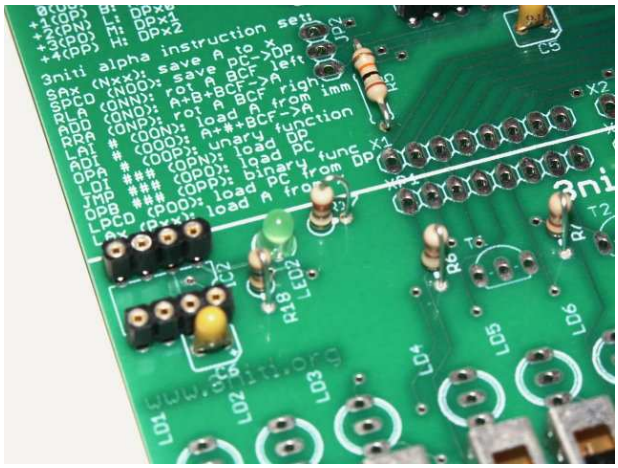
Следующий шаг - припаивание регулятора напряжения 5В (на плате обозначен как IC6), кварцевого резонатора 20 МГц (Q1) и двух светодиодов - красный (5 мм) индикатор питания LED1 и зелёный (3 мм) индикатор работы троичного вычислителя LED2:

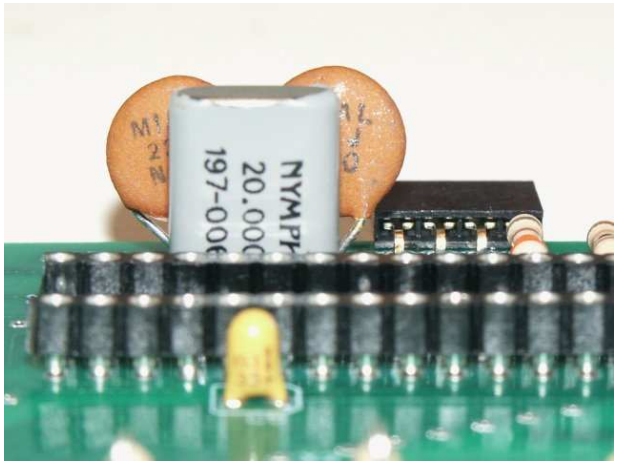




Зелёный светодиод LED2 припаивается рядом с IC2 сточенной стороной вниз:

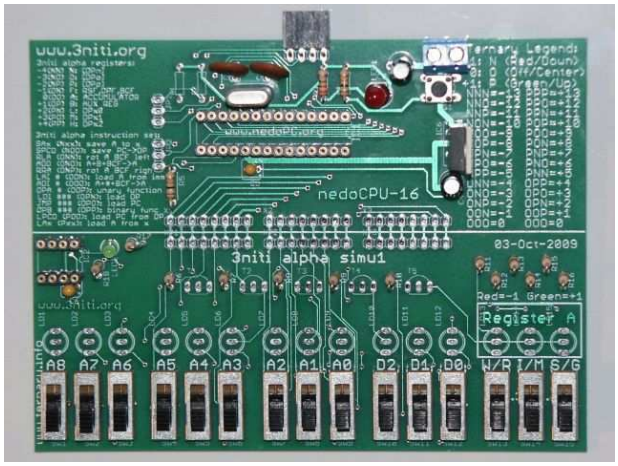
Кварцевый резонатор припаивается на своё место около IC1:





Вот так должна выглядеть плата после этого шага:

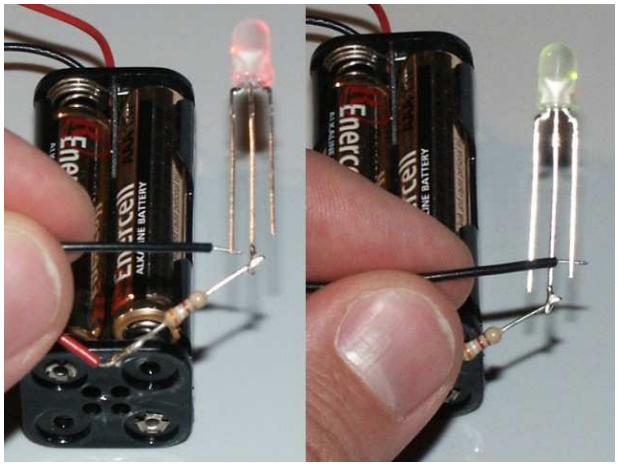
Чуть правее припаивается регулятор напряжения 5В - это 7805 или его российский аналог "КРЕН5" (КР142ЕН5). Рядом припаивается красный светодиод LED1 сточенной стороной вверх (как нарисовано на плате):



Если на плату планируется подавать уже стабилизированное напряжение 5В, то можно обойтись без регулятора напряжения, а также без конденсатора C3, однако первую и третью (последнюю) контактные площадки "IC6" надо соединить проводником.

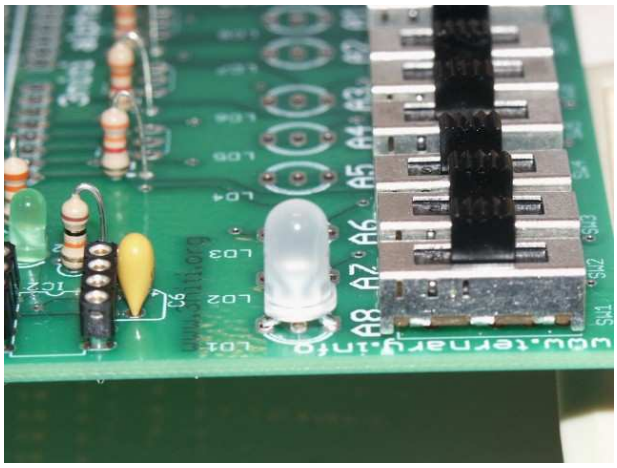
Красно-зелёные светодиоды

Теперь приступаем к припаиванию двух-цветных светодиодов с общим анодом[2] LD1...LD15. Чтобы удостовериться что у вас именно то, что нужно - можно провести следующий тест:

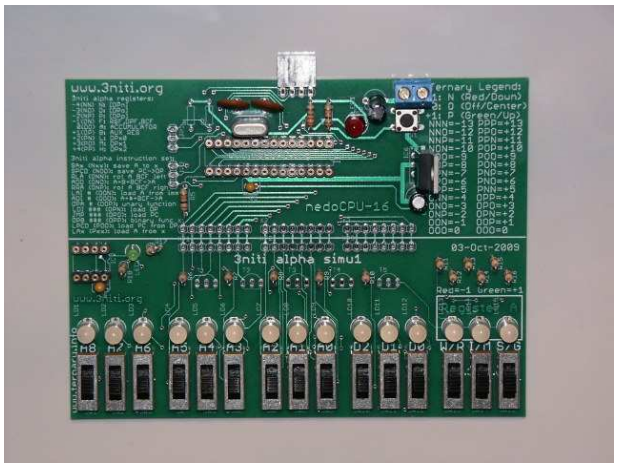


Берём источник питания около 5В (в данном случае 4 пальчиковые батарейки - 6В) и подаём "+" через резистор 300 Ом на среднюю ножку светодиода. Располагаем светодиод скошенной стороной влево - при этом если подавать "-" на левую ножку - будет гореть красный свет, а если на правую - зелёный. Если всё так, то припаиваем светодиод скошенной стороной вверх (а если наоборот - то вниз):

[2] mouser.com 604-WP59EGW/CA

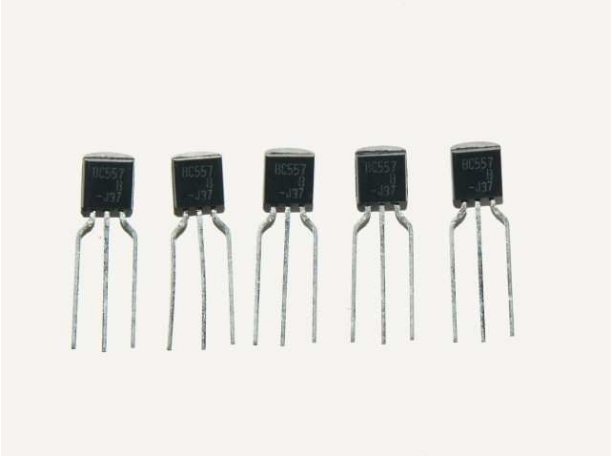


С установленными светодиодами плата будет выглядеть так:

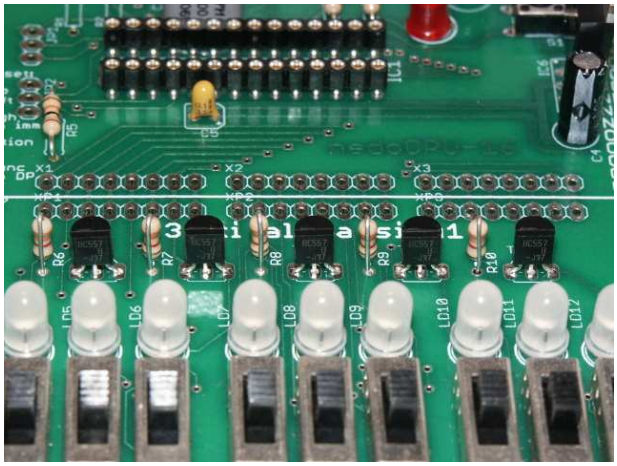


Транзисторы

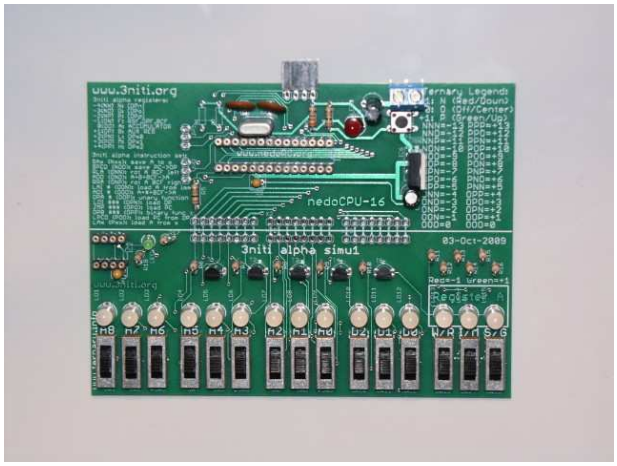
Теперь принимаемся за транзисторы T1...T5. В данном случае это PNP-транзисторы BC557B (можно взять любой аналогичный - например их российский аналог KT3107AM...KT3107KM):



Транзисторы припаиваются в нижней части платы лицевой стороной вниз (как и нарисовано на плате):



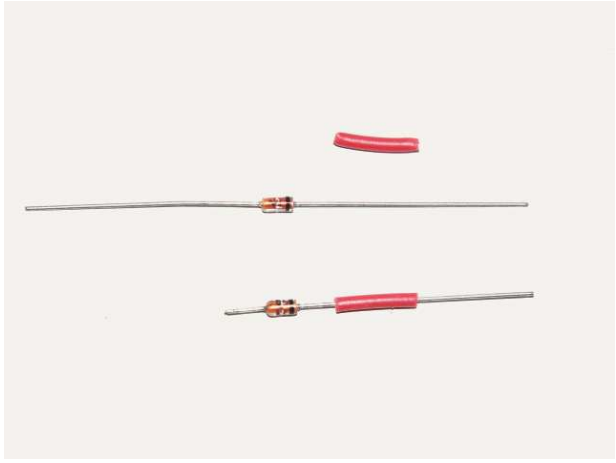
Плата с припаянными транзисторами должна выглядеть так:



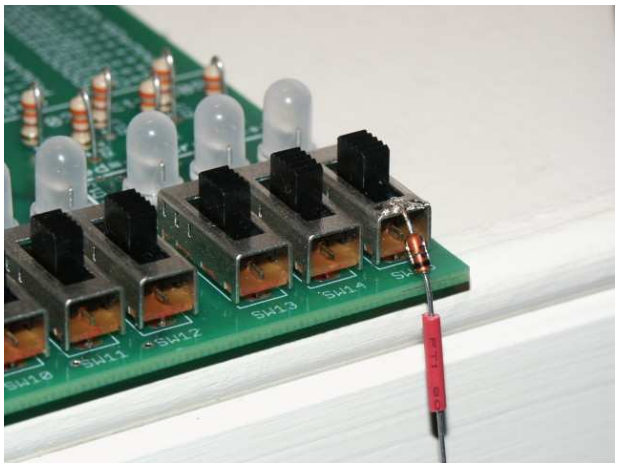
Диоды

Теперь переходим к установке диодов, которые не были предусмотрены в Rev.A - на схеме это D1...D15 (1N914 или аналогичный, например российский КД521):

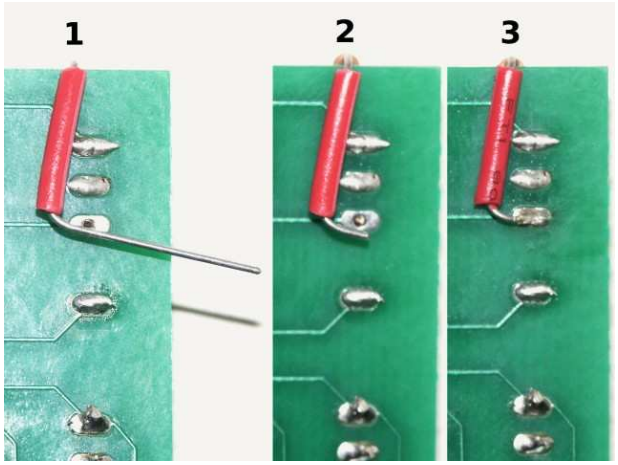
Диоды надо припаять между средней ножкой каждого переключателя и тем местом куда она должна была втыкаться - именно поэтому мы в самом начале средние ножки переключателей загибали и припаивали к металлическому корпусу - чтобы затем припаять диод туда же. Для этого надо отрезать вывод идущий от анода на расстоянии примерно 5 мм от диода, а со стороны катода (там где чёрная полоска) надеть на вывод резиновую трубку длиной около 10 мм:



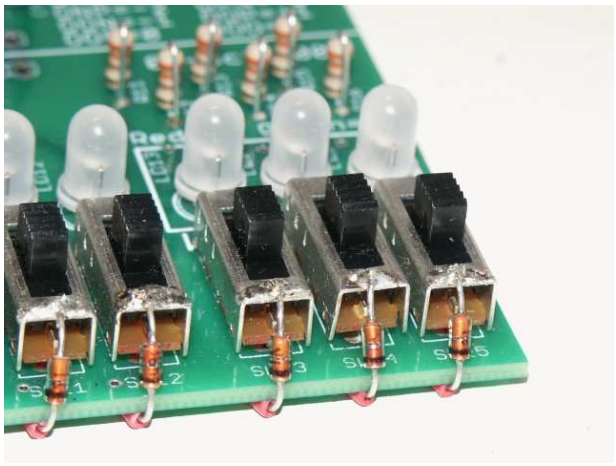
Далее каждый диод припаивается коротким концом (анодом) к металлическому корпусу своего переключателя со стороны нижнего края платы (следите чтобы диод оставаля над поверхностью платы):



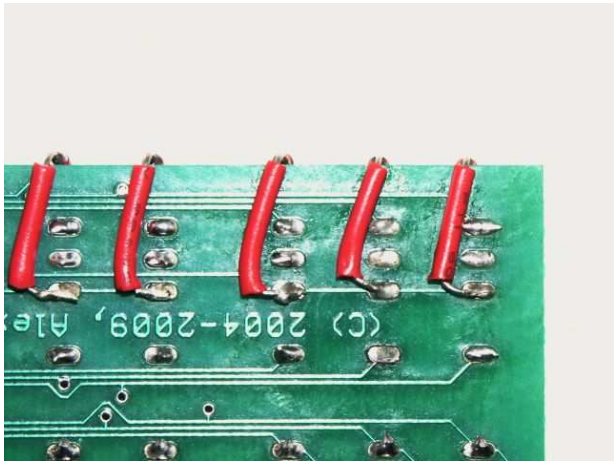
Далее свободный вывод загибаем за плату, расположив резиновую трубку у самого сгиба. Выставляющуся часть вывода также загибаем вправо - в сторону той контактной площадки, куда должна была войти средняя ножка переключателя (1), затем отрезаем лишнее, чтобы оставить только несколько миллиметров свободными от резиновой трубки (2), слегка подгибаем к плате, вставляем неглубоко в отверстие контактной площадки и припаиваем (3):



Тоже самое делаем с каждым диодом и переключателем (всего 15 раз):

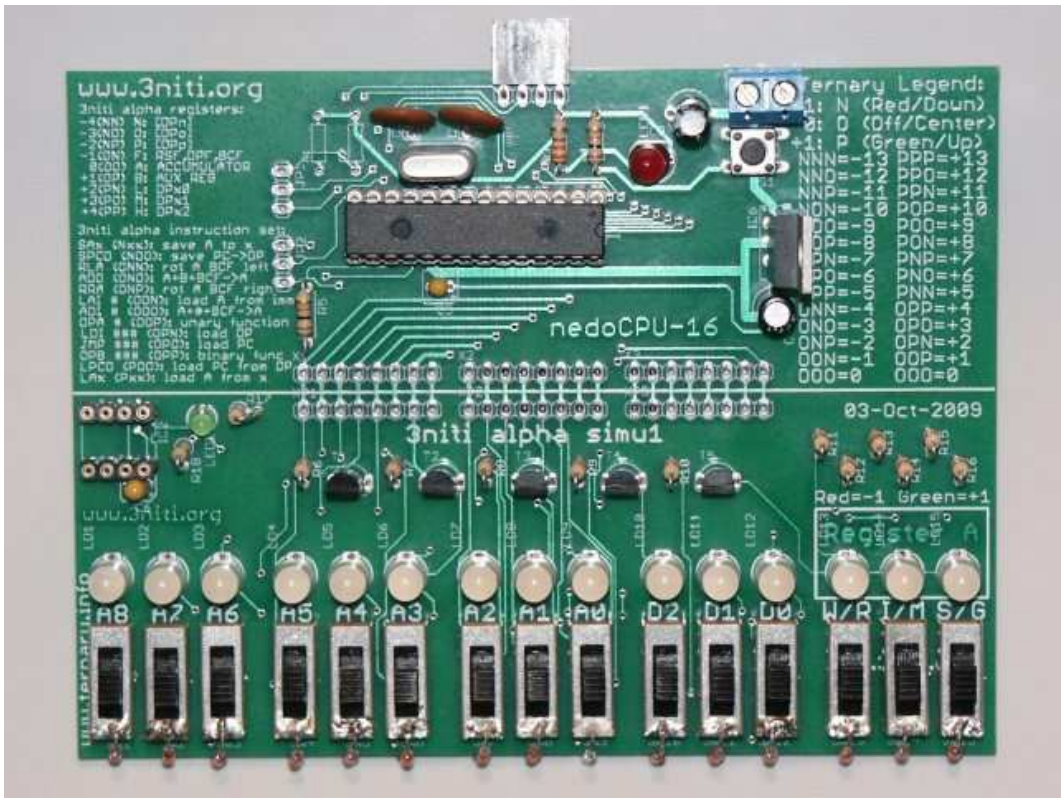


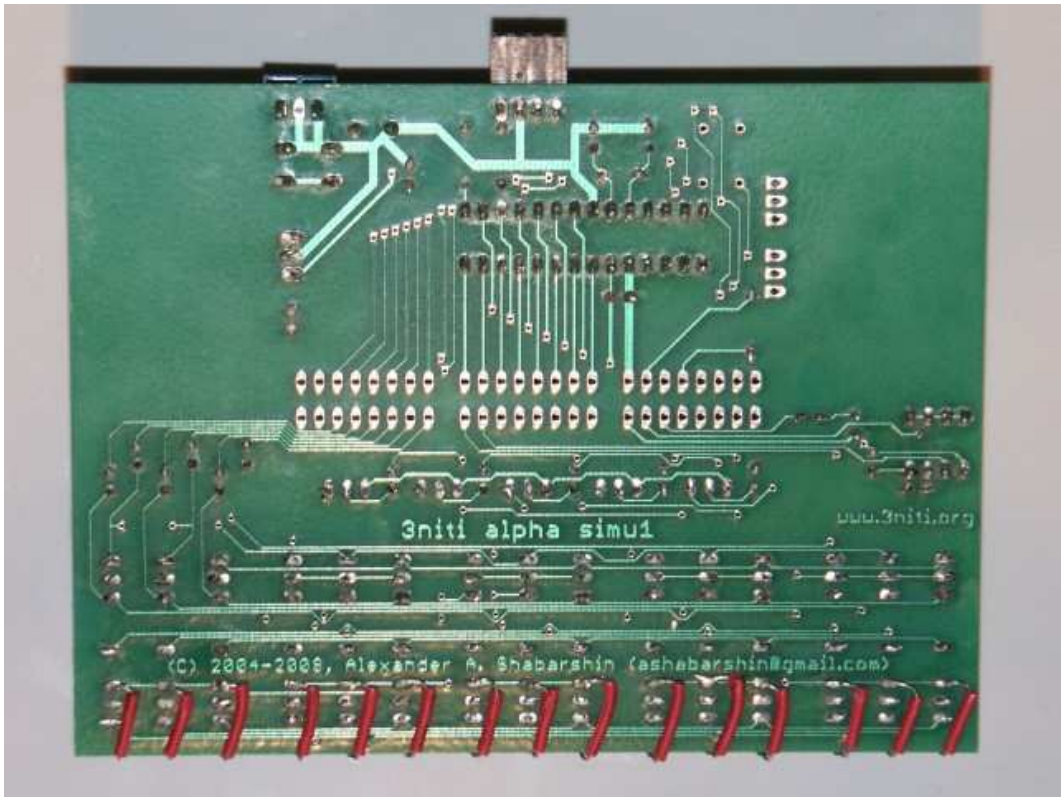
С нижней стороны это должно выглядеть примерно так:



После припаивания всех диодов, нужно прозвонить каждый и удостовериться, что ток через диод течёт только в одну сторону - если на каком-то из диодов ток течёт в обе стороны, значит средняя ножка всё таки коснулась своей контактной площадки и закоротила диод - переключатель придётся перепаивать.

На следующей странице вы можете увидеть готовую плату сверху и снизу. А программу для тестирования правильности сборки можно увидеть в разделе *Тестовая прошивка для проверки платы "3niti alpha simu1"* на странице 39.


www.3niti.org
nedoCPU-16
3niti alpha simu1
03-Oct-2009
Red=-1 Green=+1
Register A
A8 A7 A6 A5 A4 A3 A2 A1 A0 D2 D1 D0 W/R I/M S/G


3niti alpha simu1
www.3niti.org
(C) 2004-2008, Alexander A. Shabarshin (ashabarshin@gmail.com)

Исходный код эмулятора для PC

```
// Ternary.java - Alexander Shabarshin (30.10.2008-13.11.2008)
// See http://ternary.info and http://3niti.org

public class Ternary
{

 public static final int N =  -1;
 public static final int O =   0;
 public static final int P =   1;

 public static final int NN =  -4;
 public static final int NO =  -3;
 public static final int NP =  -2;
 public static final int ON =  -1;
 public static final int OO =   0;
 public static final int OP =   1;
 public static final int PN =   2;
 public static final int PO =   3;
 public static final int PP =   4;

 public static final int NNN = -13;
 public static final int NNO = -12;
 public static final int NNP = -11;
 public static final int NON = -10;
 public static final int NOO =  -9;
 public static final int NOP =  -8;
 public static final int NPN =  -7;
 public static final int NPO =  -6;
 public static final int NPP =  -5;
 public static final int ONN =  -4;
 public static final int ONO =  -3;
 public static final int ONP =  -2;
 public static final int OON =  -1;
 public static final int OOO =   0;
 public static final int OOP =   1;
 public static final int OPN =   2;
 public static final int OPO =   3;
 public static final int OPP =   4;
 public static final int PNN =   5;
 public static final int PNO =   6;
 public static final int PNP =   7;
```

```
public static final int PON =   8;
public static final int POO =   9;
public static final int POP =  10;
public static final int PPN =  11;
public static final int PPO =  12;
public static final int PPP =  13;

int value;

public static int fromTernary(String s)
{
   int n = 0;
   int k = 1;
   for(int i=s.length()-1;i>=0;i--)
   {
     char c = s.charAt(i);
     if(c!='N'&&c!='O'&&c!='P'&&c!='n'&&c!='o'&&c!='p') return 0;
     if(c=='N') n-=k;
     if(c=='P') n+=k;
     if(c=='n') n-=k;
     if(c=='p') n+=k;
     k *= 3;
   }
   return n;
}

public static String toTernary(int num)
{
   int n,m,k;
   String s = null;
   k = 387420489;
   for(int i=0;i<=18;i++)
   {
     n = num/k;
     num -= n*k;
     m = k/2;
     if(num>+m){n++;num-=k;}
     if(num<-m){n--;num+=k;}
     if(n!=0 && s==null) s="";
     if(n==-1) s+="N";
     if(n==0 && s!=null) s+="O";
     if(n==1) s+="P";
     k /= 3;
   }
   if(s==null) s="O";
   return s;
}

public String toString()
{
   return toTernary(value);
}
```

```
 public String toString(int r)
 {
    String s = toTernary(value);
    while(s.length() < r) s="0"+s;
    return s;
 }

 public void set(int n)
 {
    value = n;
 }

 public int get()
 {
    return value;
 }

 public Ternary()
 {
    set(0);
 }

 public Ternary(int n)
 {
    set(n);
 }

 public Ternary(int l, int m, int h)
 {
    set(l+3*m+9*h);
 }

 public Ternary(String s)
 {
    set(fromTernary(s));
 }

 public Ternary(Ternary t)
 {
    set(t.get());
 }

}
```

```
// Alpha1.java - Alexander Shabarshin  28.10.2008
// version 1.5 ( see http://3niti.org )

// 14.11.2008 (v1.5) - fixed register A and data display
// 12.11.2008 (v1.4) - fixed register A display
// 11.11.2008 (v1.3) - big endian LDI and JMP
// 07.11.2008 (v1.2) - correctly implemented flag RSF
// 06.11.2008 (v1.1) - fixed saving of zero address in case of interrupt
// 05.11.2008 (v1.0) - initial version (writing to ROM is allowed)

import java.applet.*;
import java.util.*;
import java.awt.*;
import java.awt.event.*;

public class Alpha1 extends Applet implements Runnable, MouseListener
{
 Thread thread = null;
 int[] leds;
 int[] swis;
 int[] swix;
 int gStep,PC;
 int rsf,dpf,bcf;
 Ternary A;
 Ternary B;
 Ternary[] ram;
 Ternary[] rom;
 int[] D;
 public static final int MEM_SZ = 2187;
 public static final int MEM_MX = 1094;

 public void init()
 {
   setBackground(new Color(255,255,255));
   leds = new int[15];
   swis = new int[15];
   swix = new int[15];
   rom = new Ternary[MEM_SZ];
   ram = new Ternary[MEM_SZ];
   for(int i=0;i<MEM_SZ;i++)
   {
     rom[i] = new Ternary(0);
     ram[i] = new Ternary(0);
   }
   A = new Ternary(0);
   B = new Ternary(0);
   D = new int[3];
   addMouseListener(this);
 }

 public void paint(Graphics g)
 {
```

```
   int i,x,y,k,w;
   int dx = size().width;
   int dy = size().height;
   int st = dx/18;
   gStep = st;
   x = st/2;
   for(i=0;i<15;i++)
   {
       y = st/2;
       switch(leds[i])
       {
         case -1: g.setColor(new Color(0xFF,0x00,0x00)); break;
         case  0: g.setColor(new Color(0xFF,0xFF,0xFF)); break;
         case  1: g.setColor(new Color(0x00,0xFF,0x00)); break;
       }
       g.fillOval(x,y,st,st);
       g.setColor(new Color(0,0,0));
       g.drawOval(x,y,st,st);
       y += st + st/2;
       k = x + st/4;
       w = st/2;
       g.setColor(new Color(255,255,255));
       g.fillRect(k,y,w,st);
       y--;
       g.setColor(new Color(0,0,0));
       g.drawRect(k,y,w,st);
       y++;
       switch(swis[i])
       {
         case -1: g.fillRect(k,y+2*st/3,w,st/3); break;
         case  0: g.fillRect(k,y+st/3,w,st/3); break;
         case  1: g.fillRect(k,y,w,st/3); break;
       }
       swix[i] = k;
       x += st;
       if((i%3)==2) x+= st/2;
   }
 }

 public void update(Graphics g)
 {
   paint(g);
 }

 public void run()
 {
   boolean fstep = false;
   boolean fsave = false;
   int SP = 0;
   while(true)
   {
/*
```

```
        int i = (int)(Math.random()*15);
        if(Math.random() > 0.5)
        {
           leds[i]++;
           if(leds[i]==2) leds[i] = -1;
        }
        else
        {
           leds[i]--;
           if(leds[i]==-2) leds[i] = 1;
        }
*/
        String s = "";
        Ternary t = new Ternary(swis[11],swis[10],swis[9]);
        int aa = 0;
        int k = 1;
        for(int j=8;j>=0;j--)
        {
          aa += swis[j]*k;
          k *= 3;
        }
        switch(swis[12])
        {
           case -1: s=get(aa).toString(); break;
           case  0: s=get(PC).toString(); break;
           case  1: set(aa,t);s=t.toString(); break;
        }
        k = s.length();
        for(int i=0;i<3;i++)
        {
           if(i < 3-k) leds[9+i] = 0;
           else
           {
              switch(s.charAt(i-3+k))
              {
                case 'N': leds[9+i] = -1; break;
                case 'O': leds[9+i] =  0; break;
                case 'P': leds[9+i] =  1; break;
              }
           }
        }
        t = new Ternary(PC);
        System.out.println("PC="+PC+" ("+t+")");
        s = t.toString();
        k = s.length();
        for(int j=0;j<9;j++)
        {
           if(j < 9-k) leds[j] = 0;
           else
           {
              switch(s.charAt(j-9+k))
              {
```

```
            case 'N': leds[j] = -1; break;
            case 'O': leds[j] =  0; break;
            case 'P': leds[j] =  1; break;
         }
      }
   }
   switch(swis[13])
   {
      case -1: if(fsave){
               fsave=false;
               PC=SP;
               SP=0;
               }
               break;
      case  1: if(!fsave){
               fsave=true;
               SP=PC;
               PC=aa;
               }
               break;
   }
   switch(swis[14])
   {
      case -1: step();
               break;
      case  0: fstep=false;
               break;
      case  1: if(!fstep){
               step();
               fstep=true;};
               break;
   }
   s = A.toString();
   k = s.length();
   for(int i=0;i<3;i++)
   {
      if(i < 3-k) leds[12+i] = 0;
      else
      {
         switch(s.charAt(i-3+k))
         {
            case 'N': leds[12+i] = -1; break;
            case 'O': leds[12+i] =  0; break;
            case 'P': leds[12+i] =  1; break;
         }
      }
   }
  try {
   repaint();
   thread.sleep(250);
  } catch(Exception e) {
   System.out.println("Exception "+e);
```

```
      }
  }
}

public void start()
{
  if(thread==null)
  {
     thread = new Thread(this);
     thread.start();
  }
}

public void stop()
{
  if(thread!=null)
  {
     thread.stop();
     thread = null;
  }
}

public void mousePressed(MouseEvent e)
{
}

public void mouseReleased(MouseEvent e)
{
}

public void mouseEntered(MouseEvent e)
{
}

public void mouseExited(MouseEvent e)
{
}

public void mouseClicked(MouseEvent e)
{
  int i;
  int x = e.getX();
  int y = e.getY();
  int st2 = gStep/2;
  int y0 = gStep*2;
  int y1 = y0+st2;
  int y2 = y0+gStep;
  if(y > y0 && y < y1)
  {
    for(i=0;i<15;i++)
    {
      if(x > swix[i] && x < swix[i]+st2)
```

```
      {
         if(swis[i] < 1) swis[i]++;
         break;
      }
    }
  }
  if(y > y1 && y < y2)
  {
    for(i=0;i<15;i++)
    {
      if(x > swix[i] && x < swix[i]+st2)
      {
         if(swis[i] > -1) swis[i]--;
         break;
      }
    }
  }
}

Ternary get(int a)
{
  Ternary t;
  if(a >= MEM_MX && a < MEM_MX+MEM_SZ)
  {
     t = ram[a-MEM_MX];
  }
  else if(a > -MEM_MX && a < MEM_MX)
  {
     t = rom[a+MEM_MX-1];
  }
  else
  {
     t = new Ternary(0);
  }
  System.out.println("get "+a+" -> "+t);
  return t;
}

void set(int a, Ternary t)
{
  System.out.println("set "+a+" <- "+t);

  if(a > -MEM_MX && a < MEM_MX)
  {
     rom[a+MEM_MX-1] = new Ternary(t);
  }

  if(a >= MEM_MX && a < MEM_MX+MEM_SZ)
  {
     ram[a-MEM_MX] = new Ternary(t);
  }
}
```

```
Ternary getA()
{
  return A;
}

void setA(Ternary t)
{
  A = t;
}

Ternary getB()
{
  return B;
}

void setB(Ternary t)
{
  B = t;
}

Ternary getF()
{
  return new Ternary(rsf*9+dpf*3+bcf);
}

void setF(Ternary t)
{
  String s = t.toString();
  int ibcf = s.length() - 1;
  int idpf = ibcf - 1;
  int irsf = idpf - 1;
  if(irsf < 0) rsf = 0;
  else
  {
    switch(s.charAt(irsf))
    {
      case 'N': rsf = -1; break;
      case 'O': rsf =  0; break;
      case 'P': rsf =  1; break;
    }
  }
  if(idpf < 0) dpf = 0;
  else
  {
    switch(s.charAt(idpf))
    {
      case 'N': dpf = -1; break;
      case 'O': dpf =  0; break;
      case 'P': dpf =  1; break;
    }
  }
```

```
  switch(s.charAt(ibcf))
  {
    case 'N': bcf = -1; break;
    case 'O': bcf =  0; break;
    case 'P': bcf =  1; break;
  }
}

int getDi()
{
  return D[dpf+1];
}

Ternary getD()
{
  return new Ternary(getDi());
}

void setD(Ternary t)
{
  D[dpf+1] = t.get();
}

void setD(int i)
{
  D[dpf+1] = i;
}

void setD(int h, int m, int l)
{
  setD(new Ternary(27*(27*h+m)+l));
}

Ternary getTriade(String s, int ll)
{
  int il = ll - 1;
  int im = il - 1;
  int ih = im - 1;
  int i = 0;
  if(ih>=0)
  {
    if(s.charAt(ih)=='P') i+=9;
    if(s.charAt(ih)=='N') i-=9;
  }
  if(im>=0)
  {
    if(s.charAt(im)=='P') i+=3;
    if(s.charAt(im)=='N') i-=3;
  }
  if(il>=0)
  {
    if(s.charAt(il)=='P') i+=1;
```

```
    if(s.charAt(il)=='N') i-=1;
  }
  return new Ternary(i);
}

Ternary getL()
{
  String s = getD().toString();
  return getTriade(s,s.length());
}

Ternary getM()
{
  String s = getD().toString();
  return getTriade(s,s.length()-3);
}

Ternary getH()
{
  String s = getD().toString();
  return getTriade(s,s.length()-6);
}

void setL(Ternary t)
{
  setD(getH().get(),getM().get(),t.get());
}

void setM(Ternary t)
{
  setD(getH().get(),t.get(),getL().get());
}

void setH(Ternary t)
{
  setD(t.get(),getM().get(),getL().get());
}

Ternary getN()
{
  return get(D[0]);
}

void setN(Ternary t)
{
  set(D[0],t);
}

Ternary getO()
{
  return get(D[1]);
}
```

```
void set0(Ternary t)
{
  set(D[1],t);
}

Ternary getP()
{
  return get(D[2]);
}

void setP(Ternary t)
{
  set(D[2],t);
}

void opa(Ternary f)
{
  String s = f.toString();
  int il = s.length() - 1;
  int im = il - 1;
  int ih = im - 1;
  int i = 0;
  int j = 0;
  int k = 0;
  if(ih>=0)
  {
    if(s.charAt(ih)=='P') i =  1;
    if(s.charAt(ih)=='N') i = -1;
  }
  if(im>=0)
  {
    if(s.charAt(im)=='P') j =  1;
    if(s.charAt(im)=='N') j = -1;
  }
  if(il>=0)
  {
    if(s.charAt(il)=='P') k =  1;
    if(s.charAt(il)=='N') k = -1;
  }
  s = A.toString();
  il = s.length() - 1;
  im = il - 1;
  ih = im - 1;
  int r = 0;
  if(ih>=0)
  {
    switch(s.charAt(ih))
    {
      case 'N': r+=9*i; break;
      case '0': r+=9*j; break;
      case 'P': r+=9*k; break;
```

```
    }
  }
  if(im>=0)
  {
    switch(s.charAt(im))
    {
      case 'N': r+=3*i; break;
      case 'O': r+=3*j; break;
      case 'P': r+=3*k; break;
    }
  }
  if(il>=0)
  {
    switch(s.charAt(il))
    {
      case 'N': r+=i; break;
      case 'O': r+=j; break;
      case 'P': r+=k; break;
    }
  }
  A = new Ternary(r);
}

void opb(Ternary fn, Ternary fo, Ternary fp)
{
  String s = fn.toString();
  int il = s.length() - 1;
  int im = il - 1;
  int ih = im - 1;
  int in = 0;
  int jn = 0;
  int kn = 0;
  if(ih>=0)
  {
    if(s.charAt(ih)=='P') in =  1;
    if(s.charAt(ih)=='N') in = -1;
  }
  if(im>=0)
  {
    if(s.charAt(im)=='P') jn =  1;
    if(s.charAt(im)=='N') jn = -1;
  }
  if(il>=0)
  {
    if(s.charAt(il)=='P') kn =  1;
    if(s.charAt(il)=='N') kn = -1;
  }
  s = fo.toString();
  il = s.length() - 1;
  im = il - 1;
  ih = im - 1;
  int io = 0;
```

```
int jo = 0;
int ko = 0;
if(ih>=0)
{
  if(s.charAt(ih)=='P') io =  1;
  if(s.charAt(ih)=='N') io = -1;
}
if(im>=0)
{
  if(s.charAt(im)=='P') jo =  1;
  if(s.charAt(im)=='N') jo = -1;
}
if(il>=0)
{
  if(s.charAt(il)=='P') ko =  1;
  if(s.charAt(il)=='N') ko = -1;
}
s = fp.toString();
il = s.length() - 1;
im = il - 1;
ih = im - 1;
int ip = 0;
int jp = 0;
int kp = 0;
if(ih>=0)
{
  if(s.charAt(ih)=='P') ip =  1;
  if(s.charAt(ih)=='N') ip = -1;
}
if(im>=0)
{
  if(s.charAt(im)=='P') jp =  1;
  if(s.charAt(im)=='N') jp = -1;
}
if(il>=0)
{
  if(s.charAt(il)=='P') kp =  1;
  if(s.charAt(il)=='N') kp = -1;
}
String sb = B.toString();
int bl = sb.length() - 1;
int bm = bl - 1;
int bh = bm - 1;
int ib = 0;
int jb = 0;
int kb = 0;
if(bh>=0)
{
  if(s.charAt(bh)=='P') ib =  1;
  if(s.charAt(bh)=='N') ib = -1;
}
if(bm>=0)
```

```
{
  if(s.charAt(bm)=='P') jb =  1;
  if(s.charAt(bm)=='N') jb = -1;
}
if(bl>=0)
{
  if(s.charAt(bl)=='P') kb =  1;
  if(s.charAt(bl)=='N') kb = -1;
}
s = A.toString();
il = s.length() - 1;
im = il - 1;
ih = im - 1;
int r = 0;
if(ih>=0)
{
  switch(s.charAt(ih))
  {
    case 'N':
      switch(ib)
      {
        case -1: r+=9*in; break;
        case  0: r+=9*io; break;
        case  1: r+=9*ip; break;
      }
      break;
    case 'O':
      switch(ib)
      {
        case -1: r+=9*jn; break;
        case  0: r+=9*jo; break;
        case  1: r+=9*jp; break;
      }
      break;
    case 'P':
      switch(ib)
      {
        case -1: r+=9*kn; break;
        case  0: r+=9*ko; break;
        case  1: r+=9*kp; break;
      }
      break;
  }
}
if(im>=0)
{
  switch(s.charAt(im))
  {
    case 'N':
      switch(jb)
      {
        case -1: r+=3*in; break;
```

```
        case  0: r+=3*io; break;
        case  1: r+=3*ip; break;
      }
      break;
    case 'O':
      switch(jb)
      {
        case -1: r+=3*jn; break;
        case  0: r+=3*jo; break;
        case  1: r+=3*jp; break;
      }
      break;
    case 'P':
      switch(jb)
      {
        case -1: r+=3*kn; break;
        case  0: r+=3*ko; break;
        case  1: r+=3*kp; break;
      }
      break;
  }
}
if(il>=0)
{
  switch(s.charAt(il))
  {
    case 'N':
      switch(kb)
      {
        case -1: r+=in; break;
        case  0: r+=io; break;
        case  1: r+=ip; break;
      }
      break;
    case 'O':
      switch(kb)
      {
        case -1: r+=jn; break;
        case  0: r+=jo; break;
        case  1: r+=jp; break;
      }
      break;
    case 'P':
      switch(kb)
      {
        case -1: r+=kn; break;
        case  0: r+=ko; break;
        case  1: r+=kp; break;
      }
      break;
  }
}
```

```
  A = new Ternary(r);

}

void signA()
{
  int ai = A.get();
  if(ai  > 0) rsf =  1;
  if(ai == 0) rsf =  0;
  if(ai  < 0) rsf = -1;
}

int step()
{
  int i,j,k,st=1;
  String s;
  Ternary tt, t = get(PC++);
  switch(t.get())
  {
    case Ternary.NNN: setN(A); break; // SAN
    case Ternary.NNO: setO(A); break; // SAO
    case Ternary.NNP: setP(A); break; // SAP
    case Ternary.NON: setF(A); break; // SAF
    case Ternary.NOO: setD(PC); break; // SPCD
    case Ternary.NOP: setB(A); break; // SAB
    case Ternary.NPN: setL(A); break; // SAL
    case Ternary.NPO: setM(A); break; // SAM
    case Ternary.NPP: setH(A); break; // SAH

    case Ternary.ONN: // RLA
      tt = new Ternary(A.get()*3+bcf);
      s = tt.toString();
      i = s.length()-3;
      if(i<0) bcf = 0;
      else
      {
        switch(s.charAt(i))
        {
          case 'N': bcf = -1; break;
          case 'O': bcf =  0; break;
          case 'P': bcf =  1; break;
        }
      }
      A = getTriade(s,i+3);
      break;

    case Ternary.ONO: // ADD
      tt = new Ternary(A.get()+B.get()+bcf);
      s = tt.toString();
      i = s.length()-3;
      if(i<0) bcf = 0;
      else
```

```
  {
    switch(s.charAt(i))
    {
      case 'N': bcf = -1; break;
      case 'O': bcf =  0; break;
      case 'P': bcf =  1; break;
    }
  }
  A = getTriade(s,i+3);
  signA();
  break;

case Ternary.ONP: // RRA
  j = 0;
  s = A.toString();
  switch(s.charAt(s.length()-1))
  {
    case 'N': j = -1; break;
    case 'O': j =  0; break;
    case 'P': j =  1; break;
  }
  A = new Ternary(A.get()/3+bcf*9);
  bcf = j;
  break;

case Ternary.OON: // LAI #
  A = get(PC++);
  st++;
  break;

case Ternary.OOO: // ADI #
  tt = get(PC++);
  tt = new Ternary(A.get()+tt.get()+bcf);
  s = tt.toString();
  i = s.length()-3;
  if(i<0) bcf = 0;
  else
  {
    switch(s.charAt(i))
    {
      case 'N': bcf = -1; break;
      case 'O': bcf =  0; break;
      case 'P': bcf =  1; break;
    }
  }
  A = getTriade(s,i+3);
  signA();
  st++;
  break;

case Ternary.OOP: // OPA #
  opa(get(PC++));
```

```
        signA();
        st++;
        break;

      case Ternary.OPN: // LDI # # #
        i = get(PC++).get();
        j = get(PC++).get();
        k = get(PC++).get();
        setD(i,j,k);
        st+=3;
        break;

      case Ternary.OPO: // JMP # # #
        i = get(PC++).get();
        j = get(PC++).get();
        k = get(PC++).get();
        PC = 27*(27*i+j)+k;
        st+=3;
        break;

      case Ternary.OPP: // OPB # # #
        Ternary tn = get(PC++);
        Ternary to = get(PC++);
        Ternary tp = get(PC++);
        opb(tn,to,tp);
        signA();
        st+=3;
        break;

      case Ternary.PNN: A=getN(); break; // LAN
      case Ternary.PNO: A=getO(); break; // LAO
      case Ternary.PNP: A=getP(); break; // LAP
      case Ternary.PON: A=getF(); break; // LAF
      case Ternary.POO: PC=getDi(); break; // LPCD
      case Ternary.POP: A=getB(); break; // LAB
      case Ternary.PPN: A=getL(); break; // LAL
      case Ternary.PPO: A=getM(); break; // LAM
      case Ternary.PPP: A=getH(); break; // LAH
    }
    System.out.println("step "+t);
    return st;
  }

}
```

Тестовая прошивка для проверки платы "3niti alpha simu1"

Данная HEX-прошивка микроконтроллера PIC16F870 предназначена для проверки правильности сборки платы - программа читает состояния "троичных" переключателей и отправляет эти сосотяния для отображения на соответствующие "троичные" светодиоды:

```
:020000040000FA
:020000000528D1
:0800080009008316031306300 2
:100010009F00831603130030850 03F3086000030B8
:1000200087008312031 3FF308500FF308600FF3006
:10003000870083160313 8B01811301178312031 3A7
:1000400064002D28F8005539F9007808AA39F8001D
:100050000310F90D780C79040800FE30FD0005301E
:10006000FE007D0885000A30F500F50B35280608EE
:1000700022208700003 0F500F50B3C280314FD0D0D
:06008000FE0B31282D28C3
:02400E00723D01
:00000001FF
```

Если при перемещении каждого переключателя вверх, соответствующий светодиод загорается зелёным светом, а при перемещении вниз - красным, то сборка прошла успешно. Также полезно попробовать разнообразные комбинации переключателей, чтобы убедиться, что переключатели и светодиоды не слишком сильно влияют друг на друга. О том как должен выглядеть процесс тестирования платы "3niti alpha simu1" можно посмотреть вот в этом видео http://www.youtube.com/watch?v=9axwRv3koZk

Ternary computer "3niti alpha simu1" Rev.A