μPD777 On-chip ROM Code Dump System Design

(A) Dump System Architectural Design

Dump Procedure

- "PA.txt" which combines dump patterns for both program ROM along with pattern ROM edited by Vim.
 The total number of patterns is 59,818 excluding comment lines. 512K bit (64K byte) EEPROM can store all the patterns in it.
- C application "rom_dump.cpp (object code : "rom_dump.exe")" refers to "PA.txt" and makes three binary files ("SST 27SF512-[3:1].bin") for three EEPROMs ("SST 27SF512-[3:1]") respectively to be programmed by multi-purpose EEPROM programmer ("TL866II Plus").
- C application "rom_dump.cpp" is compiled, debugged, and released by Microsoft Visual Studio Community.
- The logic implemented on breadboard is debugged by logic analyzer, "Hantek LA5034" (500 M/Sec samplings with 34 channel probes).

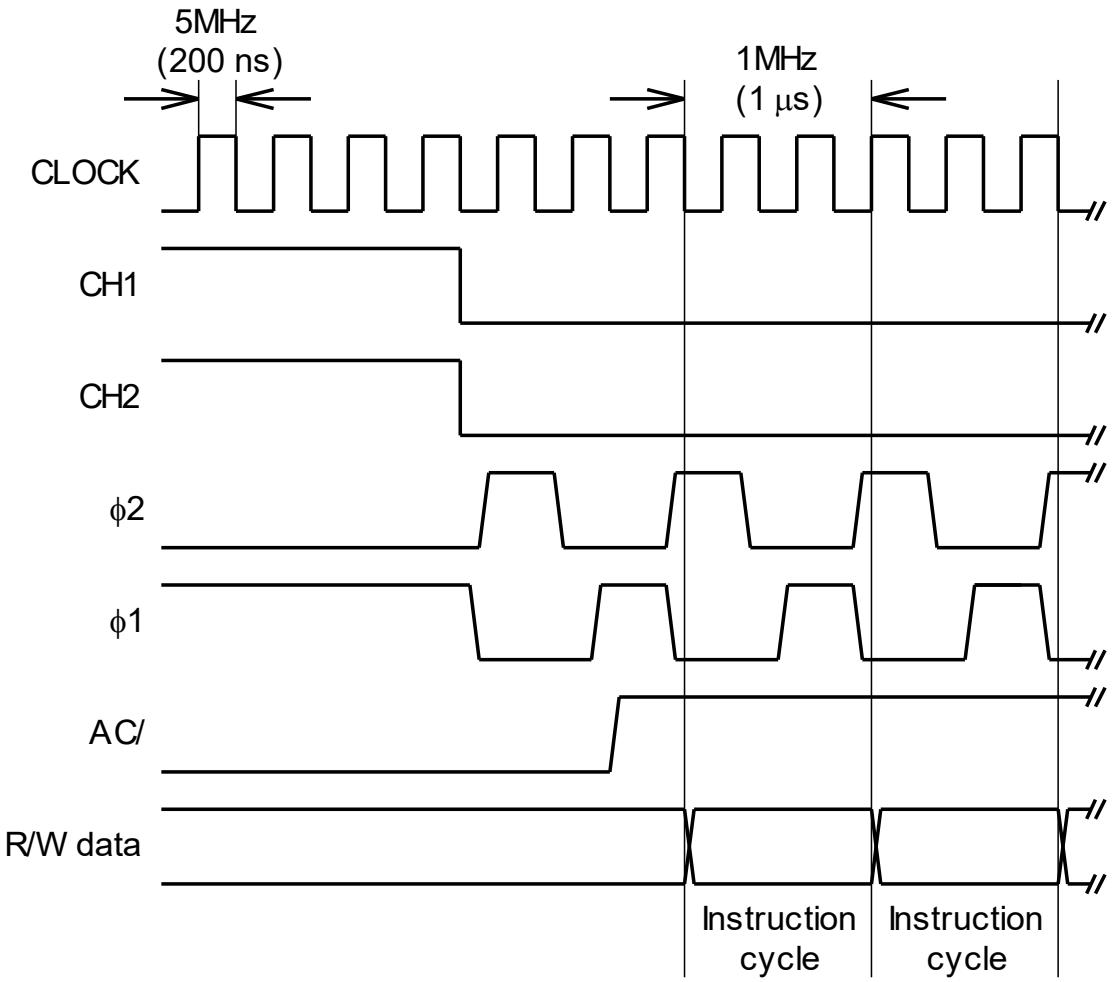
Sequence	Number of patterns (Decimal)	End byte address (Hexadecimal)
Clock initialization	13	0dh
Program ROM dump	10,484	2901h
Pattern ROM dump	49,321	0e9aah
Total	59,818	

Pattern for dump of on-chip program ROM and pattern ROM																								
EEPROM name	"SST 27SF512-3"								"SST 27SF512-2"								"SST 27SF512-1"							
Binary file name	"SST 27SF512-3.bin"								"SST 27SF512-2.bin"								"SST 27SF512-1.bin"							
μPD777 pins	-	OE/	CNT	WE	INC	CLK	AC/	CH6	CH4	CH3	CH2	CH1	K1/	K2/	K3/	K4/	K5/	K6/	K7/	GP/	PD4	PD3	PD2	PD1

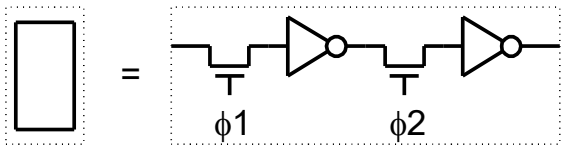
Dump result storage & pattern																	
SRAM (Temporary)	Name	"Sony CXK58256-2"								"Sony CXK58256-1"							
	Initial value	Cleared to 0h															
EEPROM	Name	"SST 27SF512-5"								"SST 27SF512-4"							
	Initial value	Erased to 0ffh															
Dump result	Data output	-	-	CH3	R	R012	R011	R010	R09	R08	R07	R06	R05	R04	R03	R02	R01
	Program ROM	0	0	0	-	*	*	*	*	*	*	*	*	*	*	*	*
	Pattern ROM	0	0	1	*	-	-	-	-	-	-	-	-	-	-	-	-

* : Data dumped
"CH3" discriminates dump result of program ROM or pattern ROM.

On-chip prescaler initialization & ROM dump timing

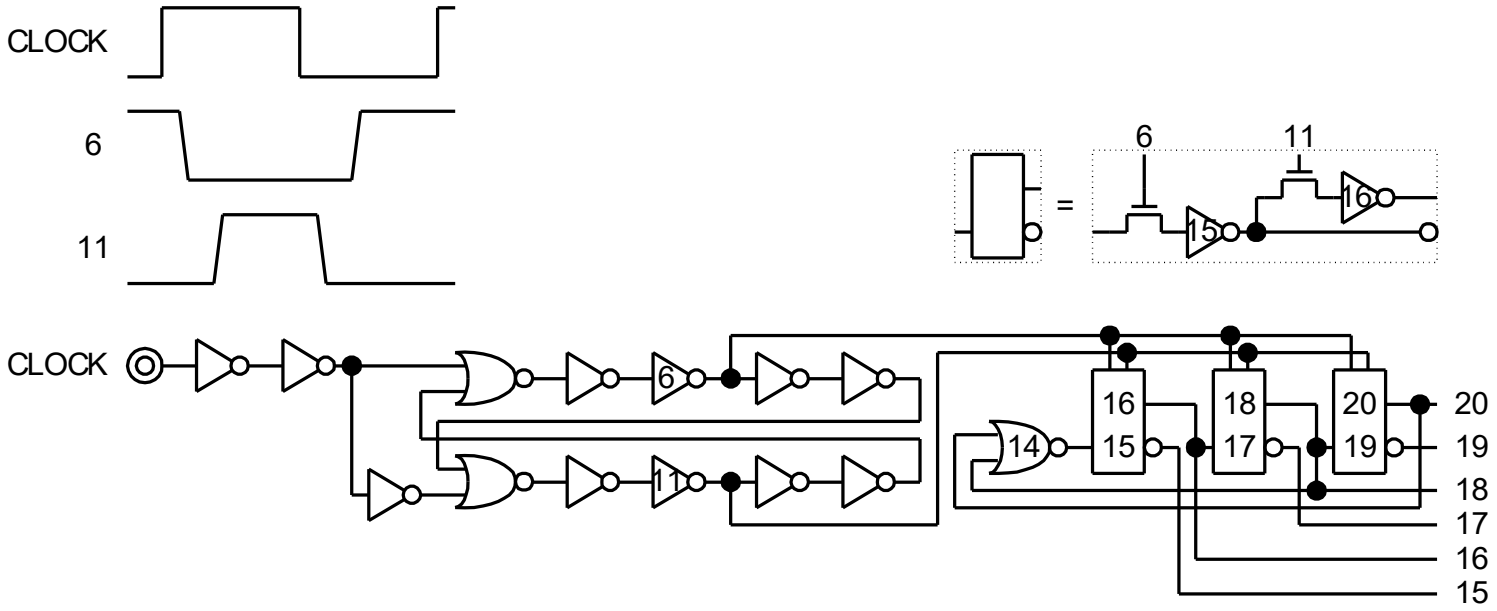


"PA.txt" (next page) precisely reflects basic timing relationship above.



Delayed Flip-Flop (DFF)

μPD777 On-chip Clock Generator with Prescaler Function



(1) 3 bit delayed flip-flops with NOR2 consists of a polynomial counter that produces 5 different values of "6, 4, 0, 1, 3" repeatedly.

(2) The polynomial counter enters an infinite loop of "6, 4, 0, 1, 3" and generates stable Φ[4:1] based upon the value.

(3) Instruction cycle (on-chip clock cycle) is defined as a cycle of rising edge of Φ2 to next rising edge of Φ2.

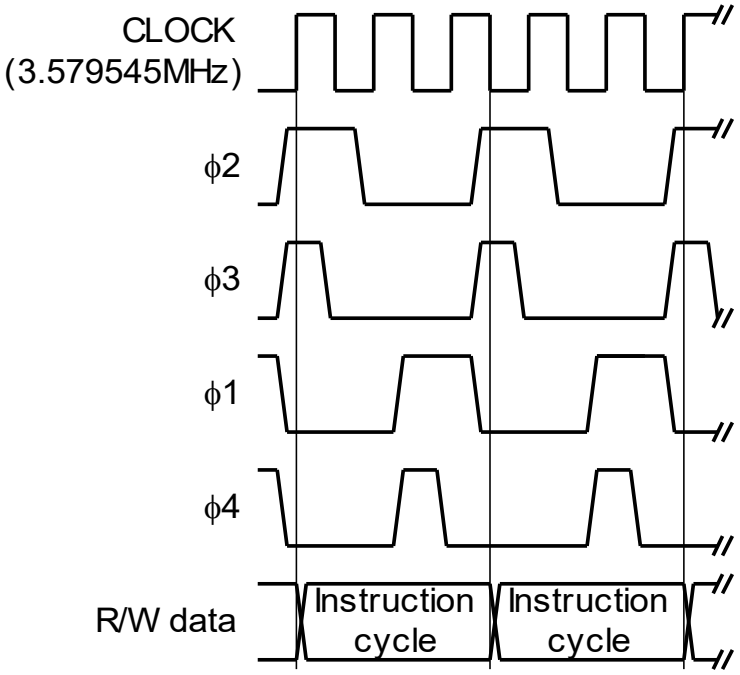
(4) One instruction cycle is equivalent of CLOCK (3.579545MHz) level transition of "0-1-0-1-0" or "1-0-1-0-1".

(5) Two instruction cycles (on-chip clocks) are equivalent of five CLOCKs as function of a prescalar implemented.

(6) On-chip logic is working at 1.431818MHz (one instruction cycle).

15	16	17	18	19	20	Φ2	Φ3	Φ1	Φ4
x	**0**	x	**1**	x	x	**1**	0	0	0
0	x	**1**	x	x	x	**1**	0	0	0
1	x	x	x	**1**	x	0	0	**1**	0
x	x	x	**1**	x	**0**	0	0	**1**	0
x	**0**	**0**	x	x	x	0	**1**	0	0
0	**0**	x	x	x	x	0	**1**	0	0
x	x	x	x	**1**	**1**	0	0	0	**1**
x	x	x	**1**	**1**	x	0	0	0	**1**

Truth table of clock generation logic

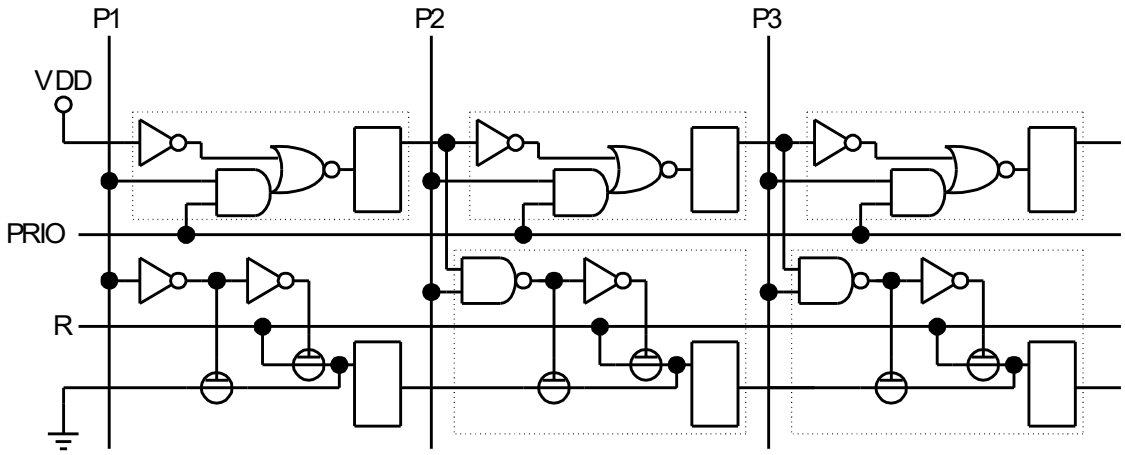


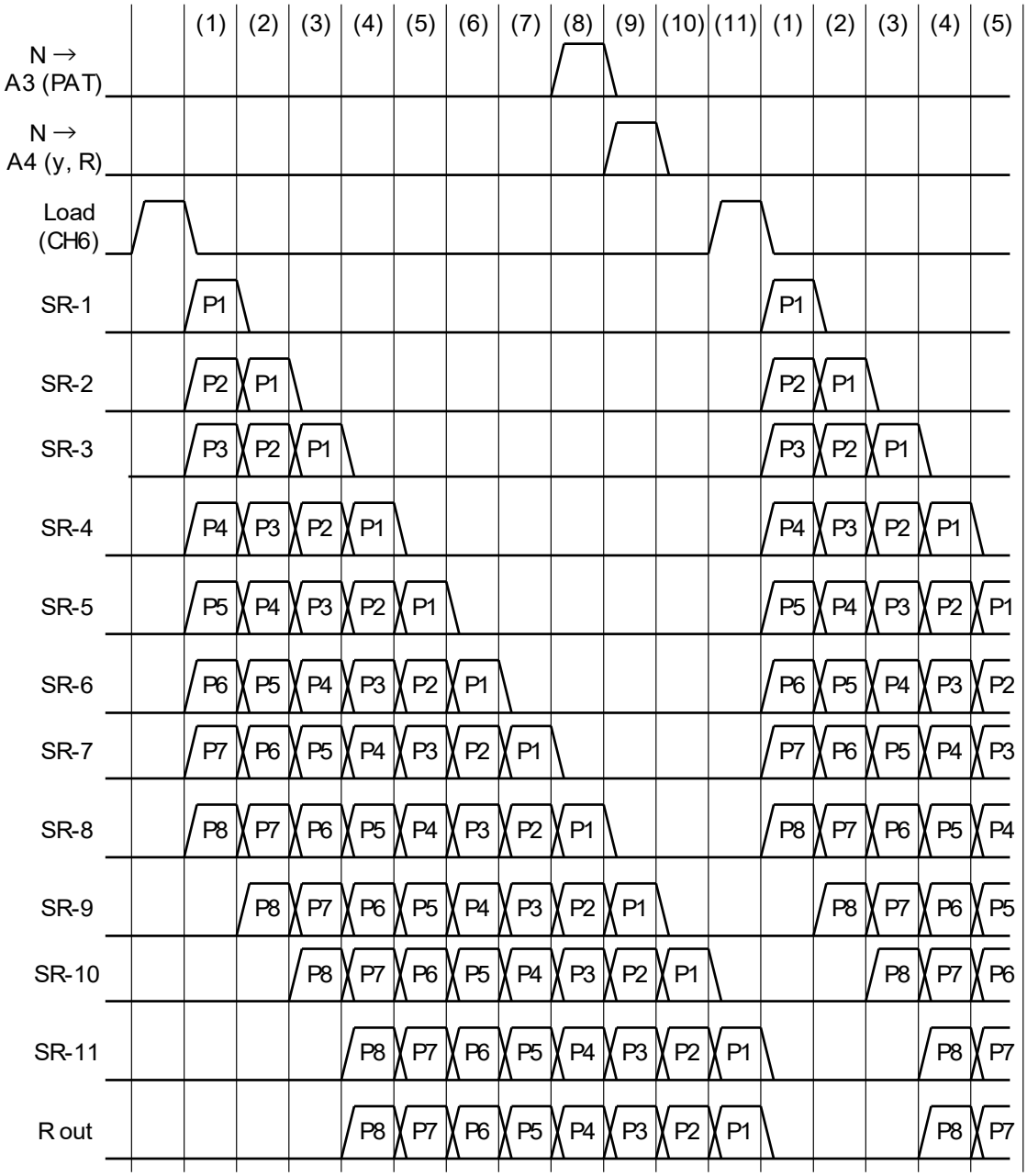
On-chip clock (Φ2, Φ3, Φ1, Φ4) waveform

Refer to a table at next page.

Abstract timing relationship of CLOCK, Polynomial counter, and On-chip clocks

CLOCK	clock		Polynomial counter								On-chip clock				Polynomial counter sequence
	6	11	14	15	16	17	18	19	20	[20,18,16]	Φ2	Φ3	Φ1	Φ4	
0	1	0	1	1	0	1	0	1	0	0	1	1	0	0	<Any sequence possible>
1	0	1	1	1	0	1	0	1	0	0	0	0	**1**	0	
0	1	0	1	0	0	1	0	1	0		**1**	**1**	0	0	
1	0	1	1	0	1	1	0	1	0	1	**1**	0	0	0	
0	1	0	1	0	1	0	0	1	0		0	0	0	0	
1	0	1	0	0	1	0	1	1	0	3	0	0	**1**	**1**	
0	1	0	0	1	1	0	1	0	0		0	0	**1**	0	
1	0	1	0	1	0	0	1	0	1	**6**	**1**	**1**	0	0	<Sequence fixed> 2 on-chip clocks = 5 external CLOCKs
0	1	0	0	1	0	1	1	0	1		**1**	0	0	0	
1	0	1	0	1	0	1	0	0	1	**4**	0	0	0	0	
0	1	0	0	1	0	1	0	1	1		0	0	**1**	**1**	
1	0	1	1	1	0	1	0	1	0	**0**	0	0	**1**	0	
0	1	0	1	0	0	1	0	1	0		**1**	**1**	0	0	
1	0	1	1	0	1	1	0	1	0	**1**	**1**	0	0	0	
0	1	0	1	0	1	0	0	1	0		0	0	0	0	
1	0	1	0	0	1	0	1	1	0	**3**	0	0	**1**	**1**	
0	1	0	0	1	1	0	1	0	0		0	0	**1**	0	
1	0	1	0	1	0	0	1	0	1	**6**	**1**	**1**	0	0	<Sequence fixed> 2 on-chip clocks = 5 external CLOCKs
0	1	0	0	1	0	1	1	0	1		**1**	0	0	0	
1	0	1	0	1	0	1	0	0	1	**4**	0	0	0	0	
0	1	0	0	1	0	1	0	1	1		0	0	**1**	**1**	
1	0	1	1	1	0	1	0	1	0	**0**	0	0	**1**	0	
0	1	0	1	0	0	1	0	1	0		**1**	**1**	0	0	
1	0	1	1	0	1	1	0	1	0	**1**	**1**	0	0	0	
0	1	0	1	0	1	0	0	1	0		0	0	0	0	
1	0	1	0	0	1	0	1	1	0	**3**	0	0	**1**	**1**	
0	1	0	0	1	1	0	1	0	0		0	0	**1**	0	





Pattern ROM Dump Timing

(B) Dump System Software Design

Excerpt of "PA.txt" (On-chip prescaler initialization & Dump Program ROM)

```
-OCWICACCCCCKKKKKKKGPPPP
-ENENLCHHHHH1234567PDDDD
-/T CK/64321////////4321
------------------------
- On-chip prescaler initialization
011001000011000000000000
011000000011000000000000
011001000011000000000000
011000000011000000000000
011001000011000000000000
011000000011000000000000
011001000011000000000000
011000000001000000000000
011001000001000000000000
011000000001000000000000
011001000001000000000000
011000100001000000000000
011001100001000000000000
- VBLK Clear 49 clocks
011000100100000000000000 -
011001100100000000000000
011000100100000000000000
011001100100000000000000
011000100100000000000000 -



011001100100000000000000 -
- VBLK Clear 49th clock
011000100100000000000000 -
011001100100000000000000
011000100100000000000000
011001100100000000000000
011000100100000000000000 -
- 0 --> A11, 0 --> DGKS
001001100001010001000000 -
001000100001010001000000
001001100001010001000000
001000100001010001000000
001001100001010001000000 -
- JP 000
001000100001100000000000 -
001001100001100000000000
001000100001100000000000
001001100001100000000000
001000100001100000000000 -
- Dump Program ROM (000) 127 addresses
011001100100101010101101 -
011000100010101010101010
011001100100101010101101
011100100010101010101010
011011100100101010101101 -
```

Hyphen ('-') at first column defines "comment line".
A pair of hyphen ('-') at 26th column appeared every 5 patterns exhibits on-chip clock cycle (instruction cycle).

Excerpt of “PA.txt” (Dump Pattern ROM)

```
-OCWICACCCCCKKKKKKKGPPPP
-ENENLCHHHHH1234567PDDDD
-/T CK/64321////////4321
------------------------
- Dump Pattern ROM
- 7F --> A1 (Y = 7F)
001001100101011001111111 -
001000100101011001111111
001001100101011001111111
001000000101011001111111
001001000101011001111111 -
- 7F --> A2 (X = 7F) (AC/ = 0)
001000000101011011111111 -
001001000101011011111111
001000000101011011111111
001001100101011011111111
001000100101011011111111 -
- 00 --> A3 (PTN[7:1] = 00)
001001100101011100000000 -
001000100101011100000000
001001100101011100000000
001000100101011100000000
001001100101011100000000 -
- 18 --> A4 (y = 1, R = 1)
001000100101011110011000 -
001001100101011110011000
001000100101011110011000
001001100101011110011000
001000100101011110011000 -
- 40 --> L,H
001001100101010111000000 -
001000100101010111000000
001001100101010111000000
001000100101010111000000
001001100101010111000000 -
- A --> MA
001000100101000001010100 -
001001100101000001010100
001000100101000001010100
001001100101000001010100
001000100101000001010100 -
- M --> MODE, 3 --> L (MODE = 0)
001001100101001110001011 -
001000100101001110001011
001001100101001110001011
001000100101001110001011
001001100101001110001011 -
- H <--> X
-  (X4[7:1] = X3[7:1] = 0)
-  (X1' = A1' = 0)
-  (L[2:1] <--> L' [2:1])
001000100101000000011000 -
001001100101000000011000
001000100101000000011000
001001100101000000011000
001000100101000000011000 -
- 0 --> DGKS, 0 --> A11 (DISP = 0)
001001100101010001000000 -
001000100101010001000000
001001100101010001000000
```

```
00100011010101000100000
00100111010101000100000 -
- NOP (Load 11 bits pattern)
00100011010100000000000 -
00100111010100000000000
00100011010100000000000
00100110010100000000000
00100010010100000000000 -
- NOP (1)
00100110010100000000000 -
00100010010100000000000
00100110010100000000000
00110010010100000000000
00101110010100000000000 -
- NOP (2)
```

- All of “PA.txt”

Source code of "rom_dump.cpp"

```
/*****************************************************************************
Program name:   rom_dump
Module name:    rom_dump.cpp
Description:    The program converts "PA.txt" and makes "SST 27SF512-*.bin" for
                dumping NEC uPD777 program ROM and pattern ROM contents.
                Input files : "PA.txt" which includes dump patterns by texts
                Output files : "SST 27SF512-3.bin", "SST 27SF512-2.bin",
                               "SST 27SF512-1.bin" byte packed binary files
                               to be programmed by EEPROM programmer
Usage:          rom_dump <enter>
Version:        1.0
Date:           June, 2, 2021
Programmer:     Tetsuji Oguchi
(C) Oguchi R&D  2021
*****************************************************************************/

#include   <stdio.h>
#include   <string.h>
#include   <process.h>
// #pragma warning(disable: 4996)

// #define RELEASE

static char   infile[] = "PA.txt", out3file[] = "SST 27SF512-3.bin";
static char   out2file[] = "SST 27SF512-2.bin", out1file[] = "SST 27SF512-1.bin";
static char   ibuf[40], obuf[3][40];
static int    i, j[8] = {0x80, 0x40, 0x20, 0x10, 0x08, 0x04, 0x02, 0x01}, k;
static int    block, length;

FILE   *patifp, *bin3ofp, *bin2ofp, *bin1ofp;


int main()
{
    printf("NEC uPD777 program ROM & pattern ROM dump program\n");
    printf("                              (C) Oguchi R&D  2021\n\n");

// Check & specify file I/O
    if (fopen_s(&patifp, "PA.txt", "r"))
    {
        printf("Input file PA.txt doesn't exist\n");
        exit(0);
    }
    if (fopen_s(&bin3ofp, "SST 27SF512-3.bin", "wb"))
    {
        printf("Output file SST 27SF512-3.bin open error...\n");
        exit(0);
    }
    if (fopen_s(&bin2ofp, "SST 27SF512-2.bin", "wb"))
    {
        printf("Output file SST 27SF512-2.bin open error...\n");
        exit(0);
    }
    if (fopen_s(&bin1ofp, "SST 27SF512-1.bin", "wb"))
    {
        printf("Output file SST 27SF512-1.bin open error...\n");
        exit(0);
    }
// Start dump
    fgets(ibuf, 40, patifp);
```

```
    while (!feof(patifp))
    {
        if (ibuf[0] != '-')
        {
            i = 0;
            k = 0;
            length = 8;
            block = 0;
            while (block < 3)
            {
                while (i < length)
                {
                    if (ibuf[(block * 8) + i] == '1')  k += j[i];
                    i++;
                }
                switch (block)
                {
                case 0: fputc(k, bin3ofp); break;
                case 1: fputc(k, bin2ofp); break;
                case 2: fputc(k, bin1ofp); break;
                }
                i = 0;
                k = 0;
                block++;
            }
            }
        fgets(ibuf, 40, patifp);
    }

// End dump
printf("NEC uPD777 program ROM & pattern ROM dump completed\n");

fclose(patifp);
fclose(bin3ofp);
fclose(bin2ofp);
fclose(bin1ofp);
}
```

(C) Dump System Breadboard Design

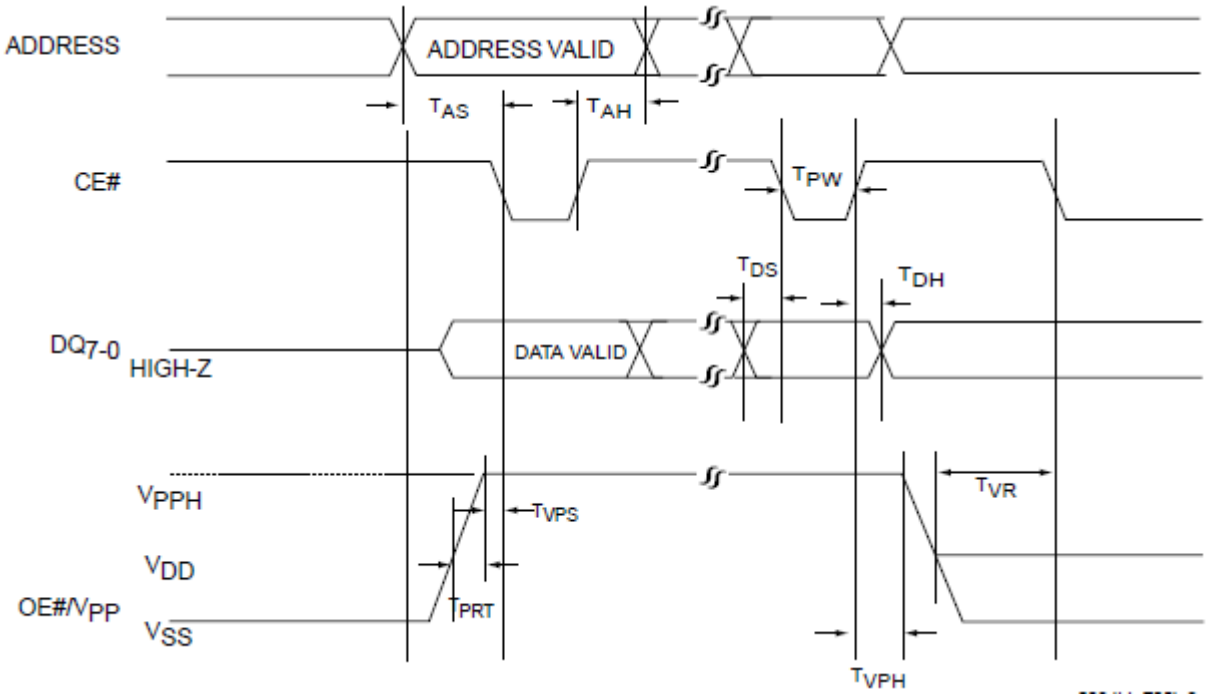
Devices		Signals	Switches		
			0 → SRAM	**777 → SRAM**	**SRAM → EEPROM**
16 bit pattern EEPROM address counter ("74LS161-[8:5]") One pulse signal clocked by 19.53125 KHz triggered by **Start** switch clears counter to zero		Clock	Low	5 MHz	Low
		Increment	Low	"CNT" = High	Low
		A[15:0]	0h	0h ~ 0e9aah	0h
15 bit dump SRAM&EEPROM address counter ("74LS161-[4:1]") One pulse signal clocked by 19.53125 KHz triggered by **Start** switch clears counter to zero		Clock	5 MHz	5 MHz	19.53125 KHz
		Increment	"A14" = Low	"INC" = High	"A14" = Low
		A[13:0]	0h ~ 4000h	0h ~ 2e6fh	0h ~ 4000h
EPROM Pattern Generator ("SST 27SF512-[3:1]")		OE#/VPP	Low		
		CE#	Low		
SRAM Dump Data Storage ("Sony CXK58256-[2:1]")		OE/	High	High	Low
		WE/	5 MHz/	"WE" = High	High
		CE/	Low		
EPROM SRAM Data Copy ("SST 27SF512-[5:4]")		OE#/VPP	High	High	+12V
		CE#	High	High	Timing Low*
Tristate Buffer	ROM code to 777 ("74LS245-[6:5]")	G/	"OE/" = Low	"OE/" = Low	"OE/" = Low
	777 to SRAM ("74LS245-[4:3]")	G/	High	Low	High
	Zero to SRAM ("74LS245-[2:1]")	G/	Low	High	High
END light		END	"A14" = High	"CNT" = Low	"A14" = High

19.53125 = (2500 / 128). Timing Low*: Refer to "SST 27SF512 EEPROM Program Cycle Timing (CE#)"

Bill Of Materials (BOM)

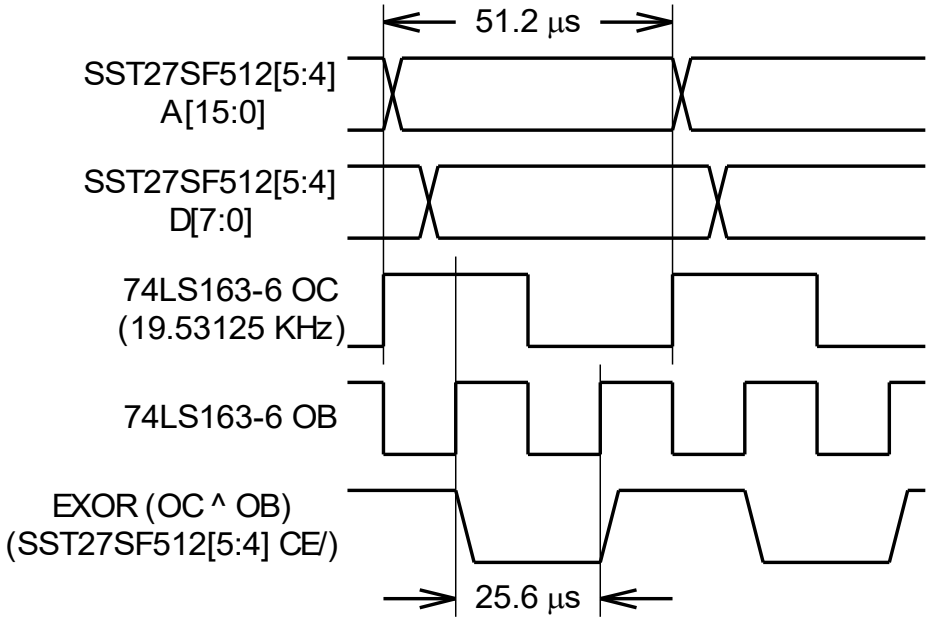
Name	Pins	Function	Piece(s)
TTL 74HCT00	14	Quad (2 input NAND)	1
TTL 74HCT04	14	Hex (Inverter)	1
TTL 74HCT08	14	Quad (2 input AND)	1
TTL 74HCT14	14	Hex (Schmitt-trigger Inverter)	1
TTL 74LS51	14	Dual (AND – NOR) 1Y=/((1A·1B·1C)+(1D·1E·1F)), 2Y=/((2A·2B)+(2C·2D))	1
TTL 74LS54	14	(AND – NOR) Y=/(AB+CDE+FGH+IJ)	2
TTL 74HCT74	14	Dual (Delayed Flipflop)	2
TTL 74HCT86	14	Exclusive OR	1
TTL 74HCT161	16	4 bit binary counter, direct clear	8
TTL 74HCT163	16	4 bit binary counter, synchronous clear	2
TTL 74HCT245	20	Octal (bi-directional tri-state driver) DIR=H; A to B, DIR=L; B to A	6
SST 27SF512	28	512K x 8 (EEPROM)	5
Sony CXK58256	28	256K x 8 (SRAM)	2
Crystal	16	5 MHz (Crystal Oscillator)	1

"SST 27SF512" EEPROM Program Cycle Timing (CE#)
(A) Specification

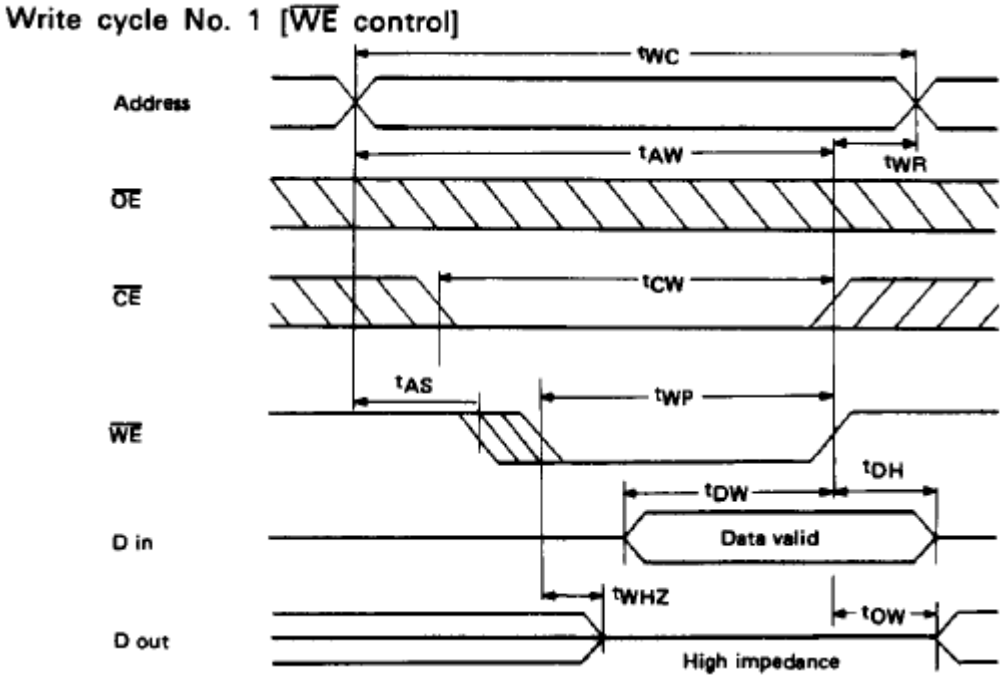


TPW = 20 µs min, 30 µs max
TAS = 1 µs min
TAH = 1 µs min
TDS = 1 µs min
TDH = 1 µs min

(B) CE/ Timing Generation Design

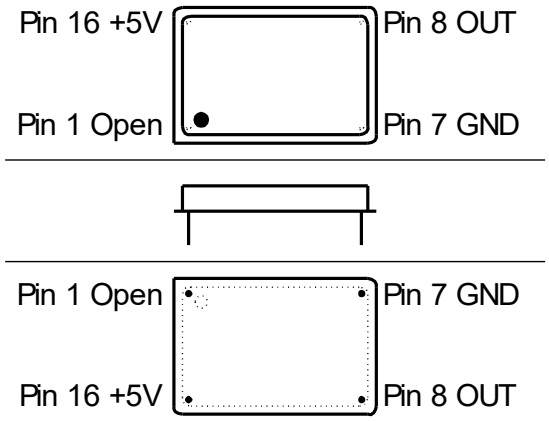


"Sony CXK58256" SRAM Write Cycle Timing (WE/)
(A) Specification

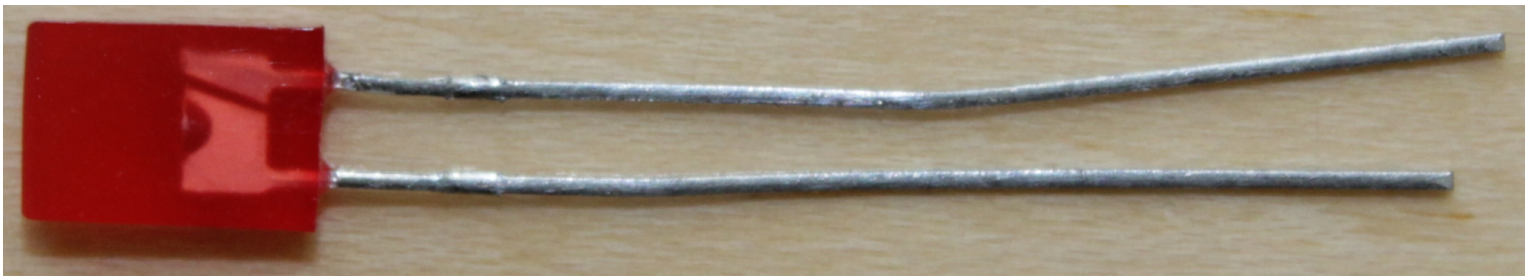


tWP = 70 ns min
tDW = 40 ns min
tDH = 0 ns min

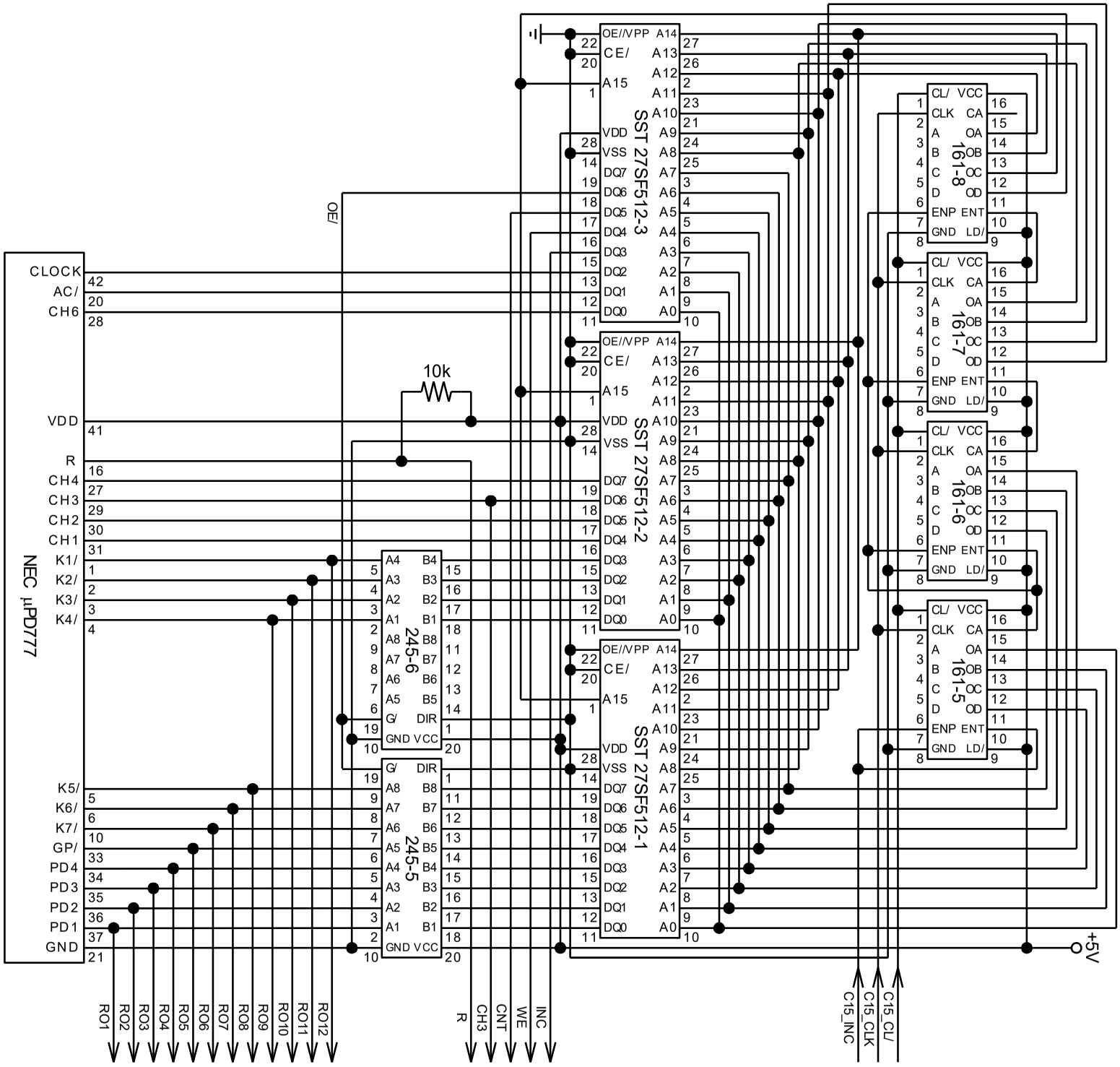
DIP Type Crystal Oscillator

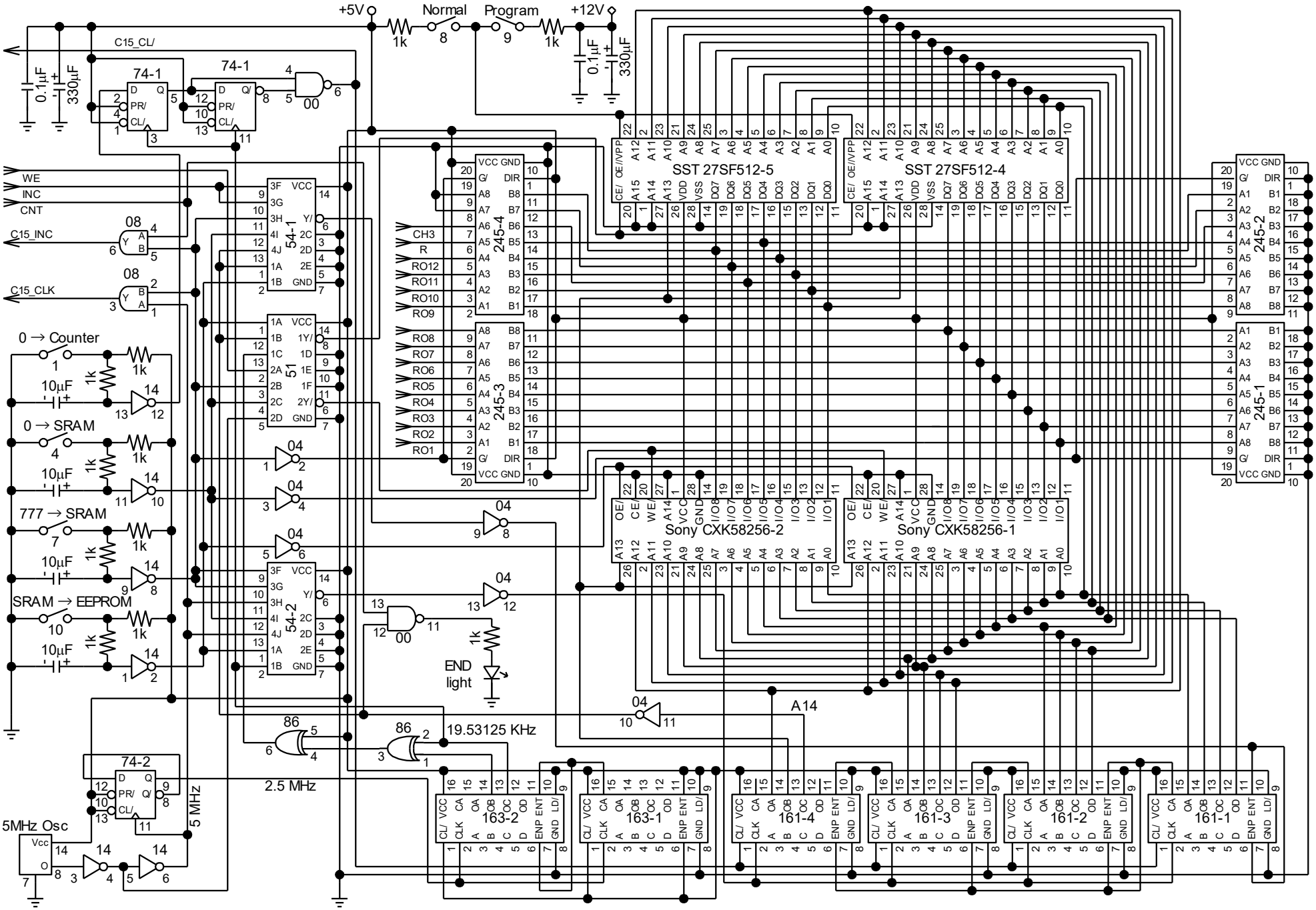


LED (Light Emission Diode)



Longer wire is anode. To verify;
Choose diode/beep position on voltmeter. Touch red probe to anode and black probe to cathode. LED turns on.


NEC μPD777
CLOCK
AC/
CH6
VDD
R
CH4
CH3
CH2
CH1
K1/
K2/
K3/
K4/
K5/
K6/
K7/
GP/
PD4
PD3
PD2
PD1
GND
OE/
10k
SST 27SF512-3
SST 27SF512-2
SST 27SF512-1
OE//VPP
CE/
A15
VDD
VSS
DQ7
DQ6
DQ5
DQ4
DQ3
DQ2
DQ1
DQ0
A14
A13
A12
A11
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
245-6
245-5
G/
DIR
GND
VCC
161-8
161-7
161-6
161-5
CL/
VCC
CLK
CA
A
OA
B
OB
C
OC
D
OD
ENP
ENT
GND
LD/
+5V
RO1
RO2
RO3
RO4
RO5
RO6
RO7
RO8
RO9
RO10
RO11
RO12
R
CH3
CNT
WE
INC
C15_INC
C15_CLK
C15_CL/


+5V
Normal
Program
+12V
1k
8
9
1k
0.1µF
330µF
C15_CL/
74-1
74-2
00
WE
INC
CNT
C15_INC
C15_CLK
08
54-1
54-2
51
04
86
0 → Counter
0 → SRAM
777 → SRAM
SRAM → EEPROM
10µF
END light
19.53125 KHz
2.5 MHz
5 MHz
5MHz Osc
A14
245-4
245-3
245-2
245-1
CH3
R
RO12
RO11
RO10
RO9
RO8
RO7
RO6
RO5
RO4
RO3
RO2
RO1
SST 27SF512-5
SST 27SF512-4
Sony CXK58256-2
Sony CXK58256-1
163-2
163-1
161-4
161-3
161-2
161-1

Dump Procedure

(A) Programming EEPROMs
Program dump pattern in "SST 27SF512[3:1]" and erase "SST 27SF512[5:4]" by EEPROM programmer and install "SST 27SF512[5:1]" on breadboard.

(B) Breadboard switch initialization

(1) Keep +5V & +12V power supply OFF

0 → Counter	0 → SRAM	777 → SRAM	Normal	Program	SRAM → EEPROM
"SW1" = OFF	"SW4" = OFF	"SW7" = OFF	"SW8" = ON(+5V)	"SW9" = OFF	"SW10" = OFF

(2) Power supply ON

(C) Dump procedure (N: switch sequence)

(1) SRAM clear
Clear SRAM ("Sony CXK58256-[2:1]") contents to zeros.

0 → Counter	0 → SRAM
1: "SW1" = ON	2: "SW4" = ON
Wait until "END" light turns on	
2: "SW1" = OFF	1: "SW4" = OFF

(2) μPD777 ROM dump to SRAM
Dump μPD777 on-chip program ROM and pattern ROM contents and stores in SRAM ("Sony CXK58256-[2:1]").

0 → Counter	777 → SRAM
1: "SW1" = ON	2: "SW7" = ON
Wait until "END" light turns on	
2: "SW1" = OFF	1: "SW7" = OFF

(3) SRAM copy to EEPROM
Copy dump result stored in SRAM ("Sony CXK58256-[2:1]") to EEPROM ("SST 27SF512-[5:4]").

0 → Counter	Normal	Program	SRAM → EEPROM
1: "SW1" = ON	2: "SW8" = OFF	3: "SW9" = ON(+12V)	4: "SW10" = ON
Wait until "END" light turns on			
4: "SW1" = OFF	3: "SW8" = ON(+5V)	2: "SW9" = OFF	1: "SW10" = OFF

From	"Sony CXK58256-2"								"Sony CXK58256-1"							
Signal Name	0	0	CH3	R	R012	R011	R010	R09	R08	R07	R06	R05	R04	R03	R02	R01
To	"SST 27SF512-5"								"SST 27SF512-4"							

(D) Analysis the ROM dump results on PC
- Turn +5V & +12V power supply OFF and remove "SST 27SF512[5:4]".
- Read the dump result and save as "SST 27SF512-[5:4].bin" by EEPROM programmer.
- Run "dump_analysis.exe" and analyze "program_ROM.csv" and "pattern_ROM.csv" on PC.
 Refer to "Dump Analysis" more in details.

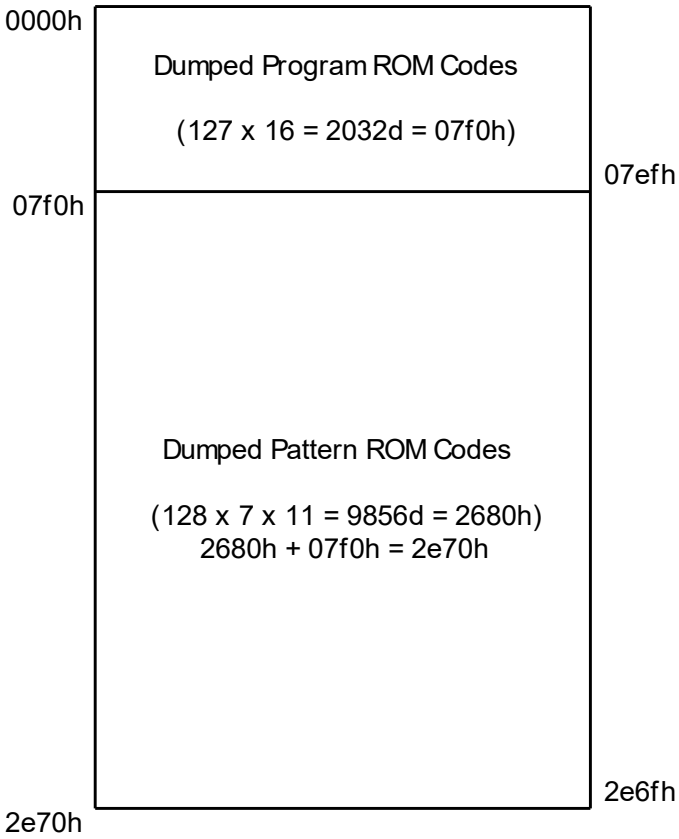
If necessary, update "PA.txt" and "SST 27SF512[3:1].bin". Repeat (A) to (D) until getting right ROM dump result.

Source code of "dump_analysis.cpp"

```
/*****************************************************************************
Program name:   dump_analysis
Module name:    dump_analysis.cpp
Description:    The program converts "SST 27SF512-5.bin" and "SST 27SF512-4.bin"
                to "program_ROM.csv" and "pattern_ROM.csv".
                Input files :
                 "SST 27SF512-5.bin" ROM dump binary file (high byte)
                 "SST 27SF512-4.bin" ROM dump binary file (low byte)
                Output files :
                 "program_ROM.csv" program ROM code tab-delimited text file
                 "pattern_ROM.csv" pattern ROM code tab-delimited text file
Usage:          dump_analysis <enter>
Version:        1.0
Date:           July, 10, 2021
Programmer:     Tetsuji Oguchi
(C) Oguchi R&D  2021
*****************************************************************************/

#include   <stdio.h>
#include   <string.h>
#include   <process.h>
#include   <clocale>

static char   inhbfile[] = "SST 27SF512-5.bin", inlbfile[] = "SST 27SF512-4.bin";
static char   out1file[] = "program_ROM.csv", out2file[] = "pattern_ROM.csv";
static int    hb, lb;
static int    i, j, r;
static int    pcode, patcode;

FILE* inhbfp, * inlbfp, * txt1ofp, * txt2ofp;

void pat_out()
{
    setlocale(LC_ALL, "en_US.UTF-8");
    wchar_t blank[] = L"\u2000\u2000";
    wchar_t ones[] = L"\u2588\u2588";

    r = fgetc(inhbfp);
    r &= 0x10;
    if (r == 0) fprintf(txt2ofp, "%ls", blank);
    else fprintf(txt2ofp, "%ls", ones);
}

int main()
{
    printf("NEC uPD777 program ROM & pattern ROM dump program 2\n");
    printf("                              (C) Oguchi R&D  2021\n\n");

    // Check & specify file I/O
    if (fopen_s(&inhbfp, "SST 27SF512-5.bin", "rb"))
    {
        printf("Input file SST 27SF512-5.bin doesn't exist\n");
        exit(0);
    }
    if (fopen_s(&inlbfp, "SST 27SF512-4.bin", "rb"))
    {
        printf("Input file SST 27SF512-4.bin open error...\n");
        exit(0);
    }
```

```
    if (fopen_s(&txt1ofp, "program_ROM.csv", "w"))
    {
        printf("Output file program_ROM.csv open error...\n");
        exit(0);
    }
    if (fopen_s(&txt2ofp, "pattern_ROM.csv", "w"))
    {
        printf("Output file pattern_ROM.csv open error...\n");
        exit(0);
    }

    // Start dumped program ROM code conversion
    for (j = 0; j < 16; j++)
    {
        for (i = 0; i < 127; i++)
        {
            hb = fgetc(inhbfp);
            lb = fgetc(inlbfp);
            hb &= 0xf;
            hb <<= 8;
            pcode = hb + lb;
            fprintf(txt1ofp, "%03X\t\n", pcode);
        }
        fprintf(txt1ofp, "\t\n");
    }

    // Start dumped pattern ROM code conversion
    for (j = 0; j < 128; j++)
    {
        for (i = 0; i < 7; i++)
        {
            r = fgetc(inhbfp);
            r = fgetc(inhbfp);
            r = fgetc(inhbfp);
            pat_out();
            pat_out();
            pat_out();
            pat_out();
            pat_out();
            pat_out();
            pat_out();
            pat_out();
            fprintf(txt2ofp, "\t\n");
        }
        fprintf(txt2ofp, "--\t\n");
    }

    // End dump
    printf("NEC uPD777 program ROM & pattern ROM dump completed\n");

    fclose(inhbfp);
    fclose(inlbfp);
    fclose(txt1ofp);
    fclose(txt2ofp);
}
```

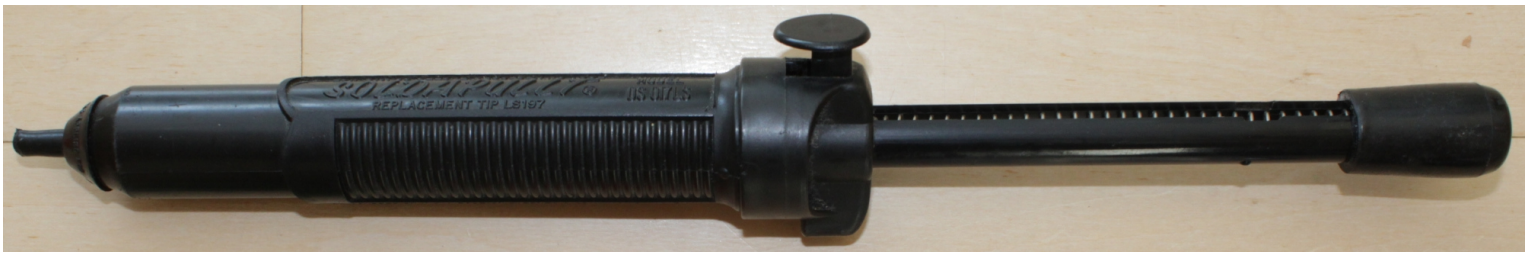
Contents of "SST 27SF512-5 (upper byte)" and "SST 27SF512-4 (lower byte)"



Address Map

Desoldering µPD777 from cartridge board

Plug a µPD777 desoldered & detached from cartridge board into 48 pin ZIF (Zero Insertion Force) socket significantly accelerates the ROM dump of multiple µPD777s.

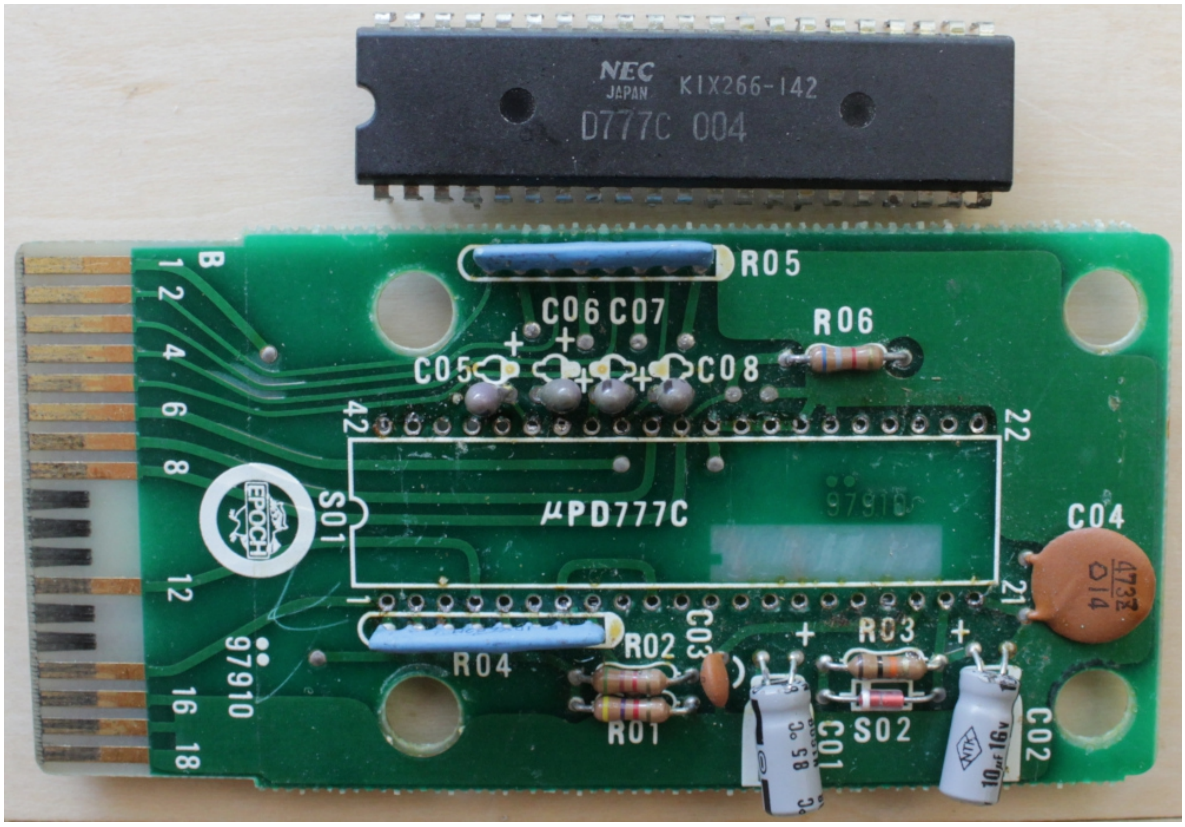


13" Desoldering Pump

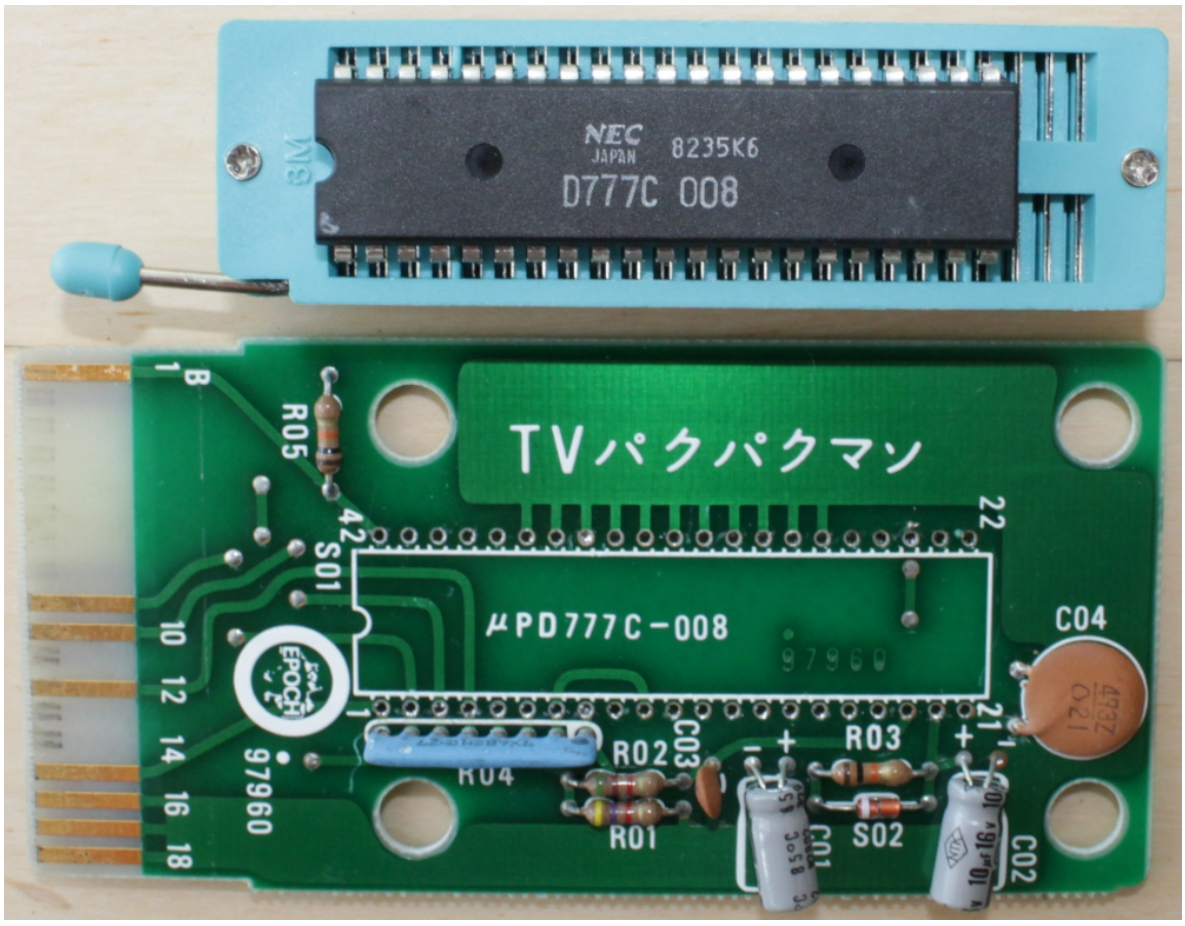
Longer desoldering pump works better than shorter one.

Tips (Through-hole):

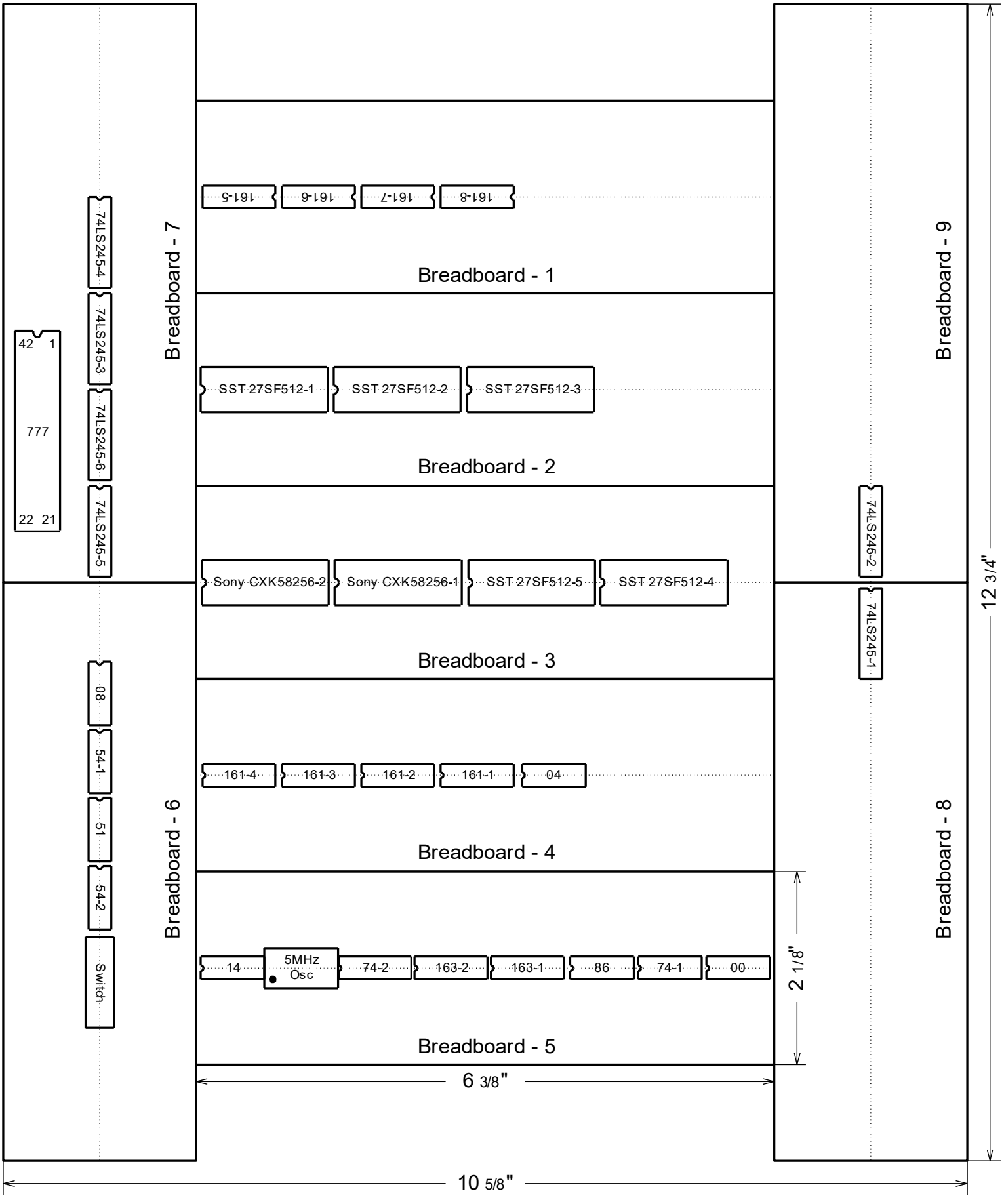
(1) Melt solder sufficiently.
(2) Handle a desolder pump on solder side melting solder on parts side, at the same time.



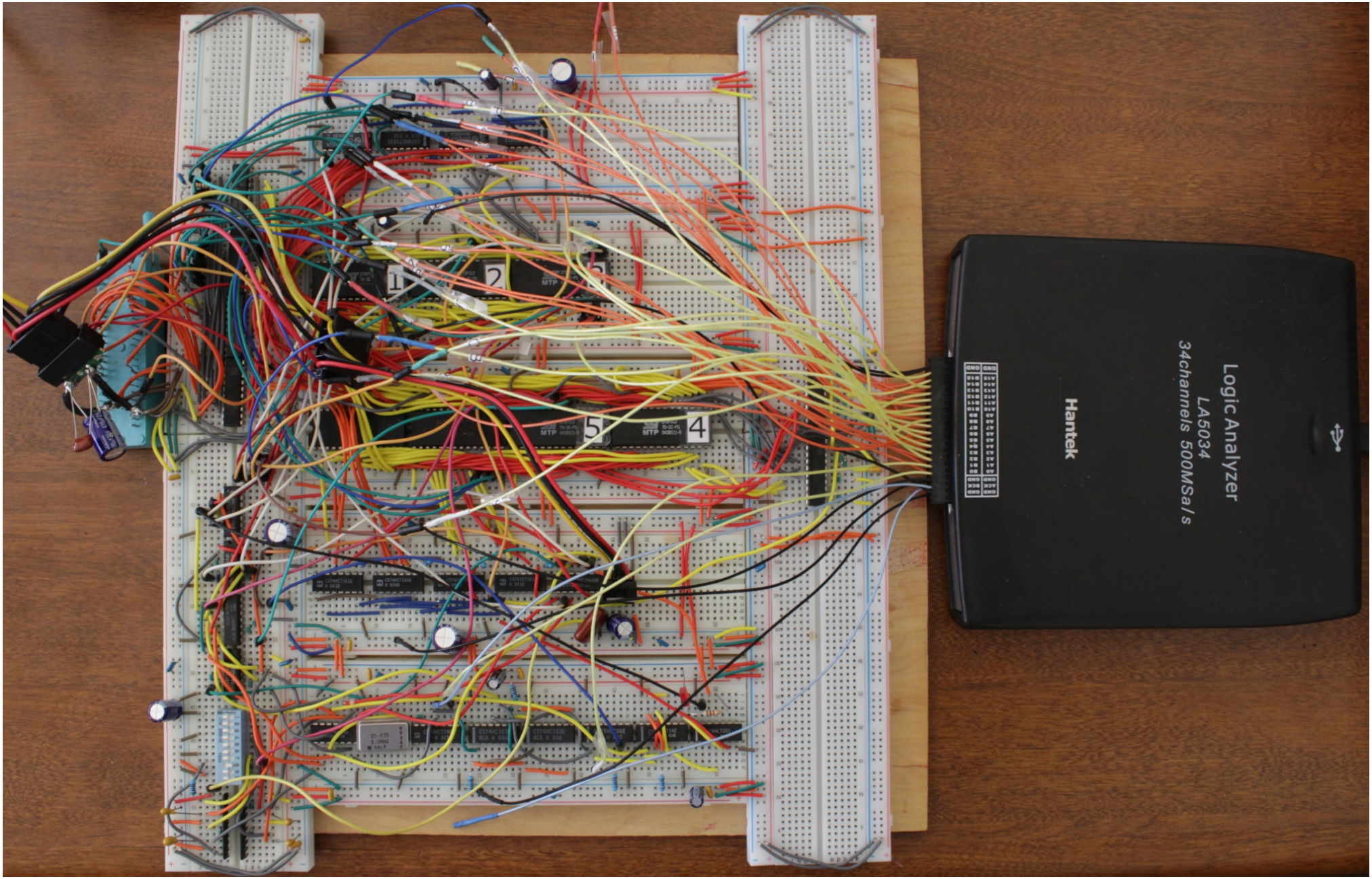
μPD777 004 (Big Sports 12) Detached



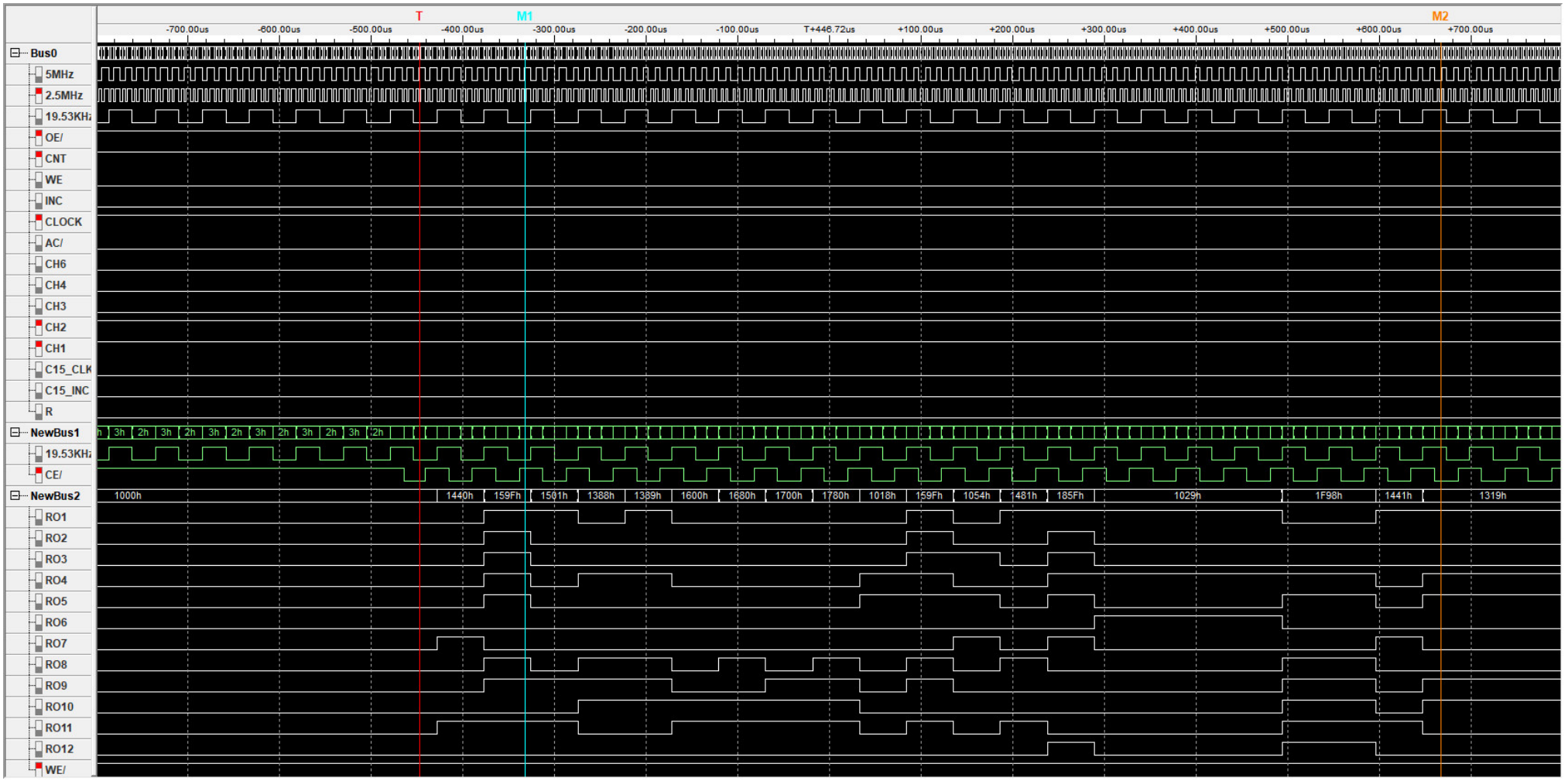
μPD777 008 (Pakpak Monster) Detached



Breadboard & Parts Layout

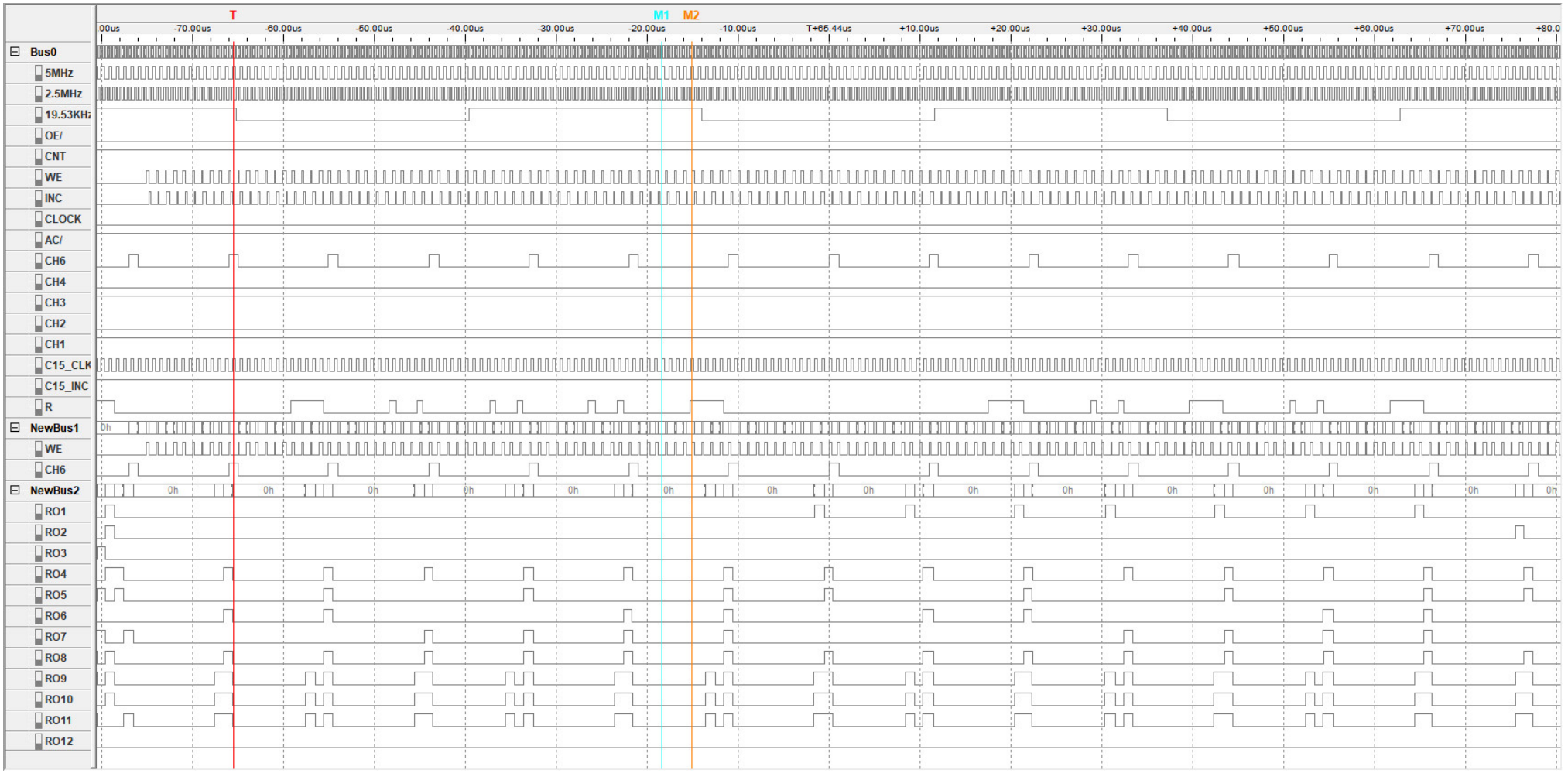


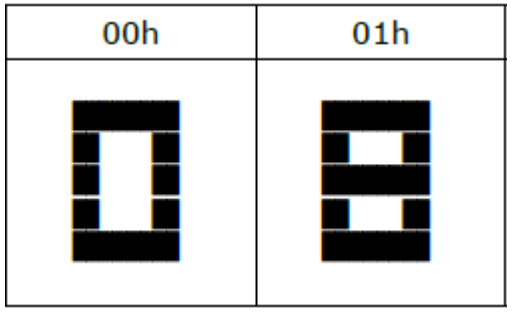
Breadboard Connected with Logic Analyzer under Debug & Dump



Program ROM Dump Wave Form of First 19 Steps
(000h, 440h, 59F, 501h, 388h, 389h, 600h, 680h, 700h, 780h, 018h, 59Fh, 054h, 481h, 85Fh, 029h, 029h, 029h, 029h)
Captured by Logic Analyzer

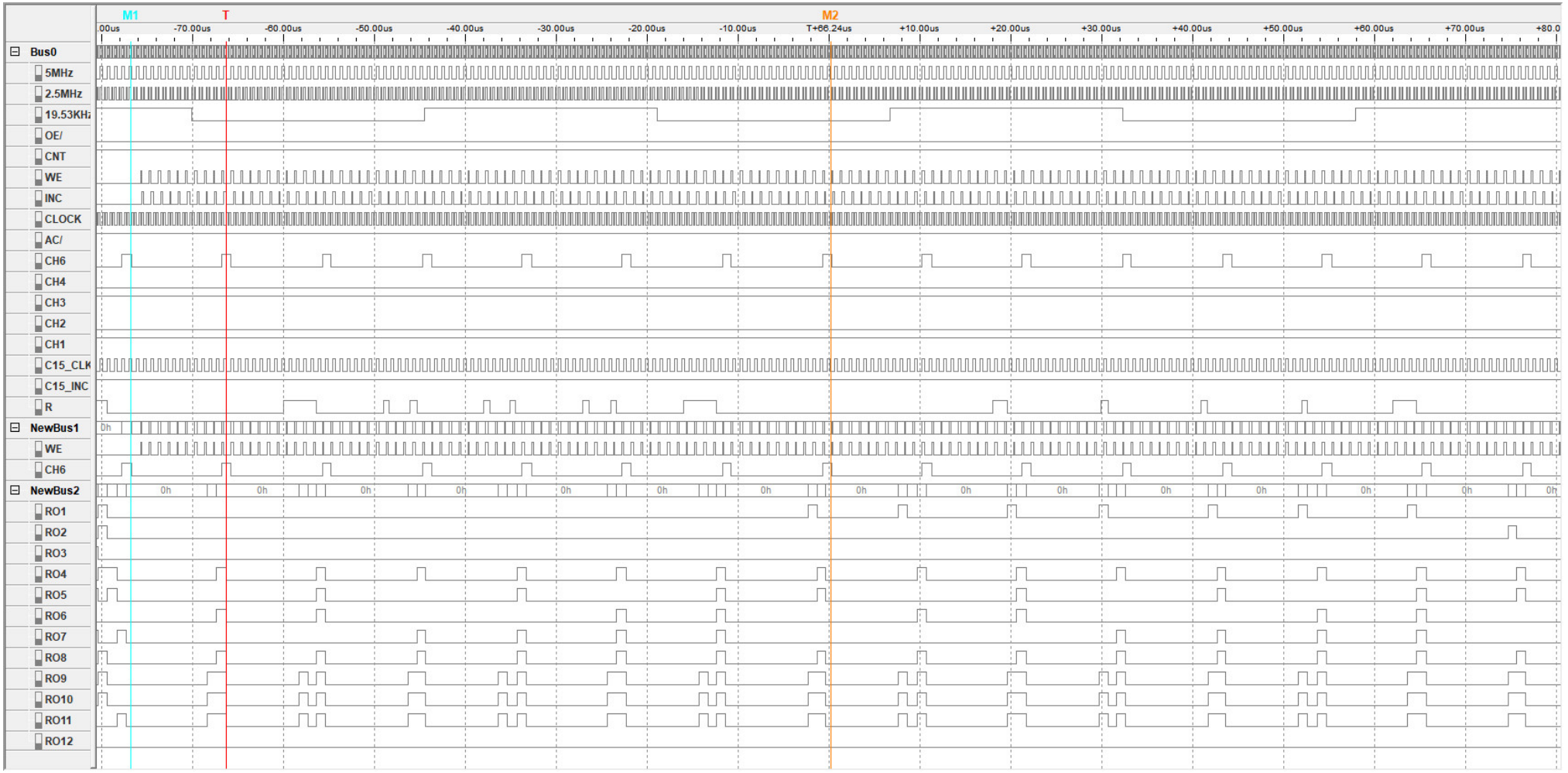
(μPD777 009 for Epoch Video Cassette “Monster Mansion”)

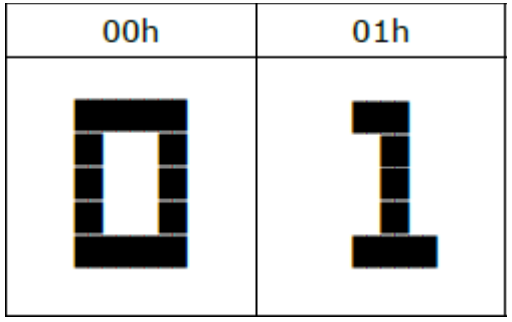




Pattern ROM Dump Wave Form of PAT[00h] (A3=00h) & PAT[01h] (A3=01h) under y=1 (A4=18h) through 7 (A4=78h) Captured by Logic Analyzer

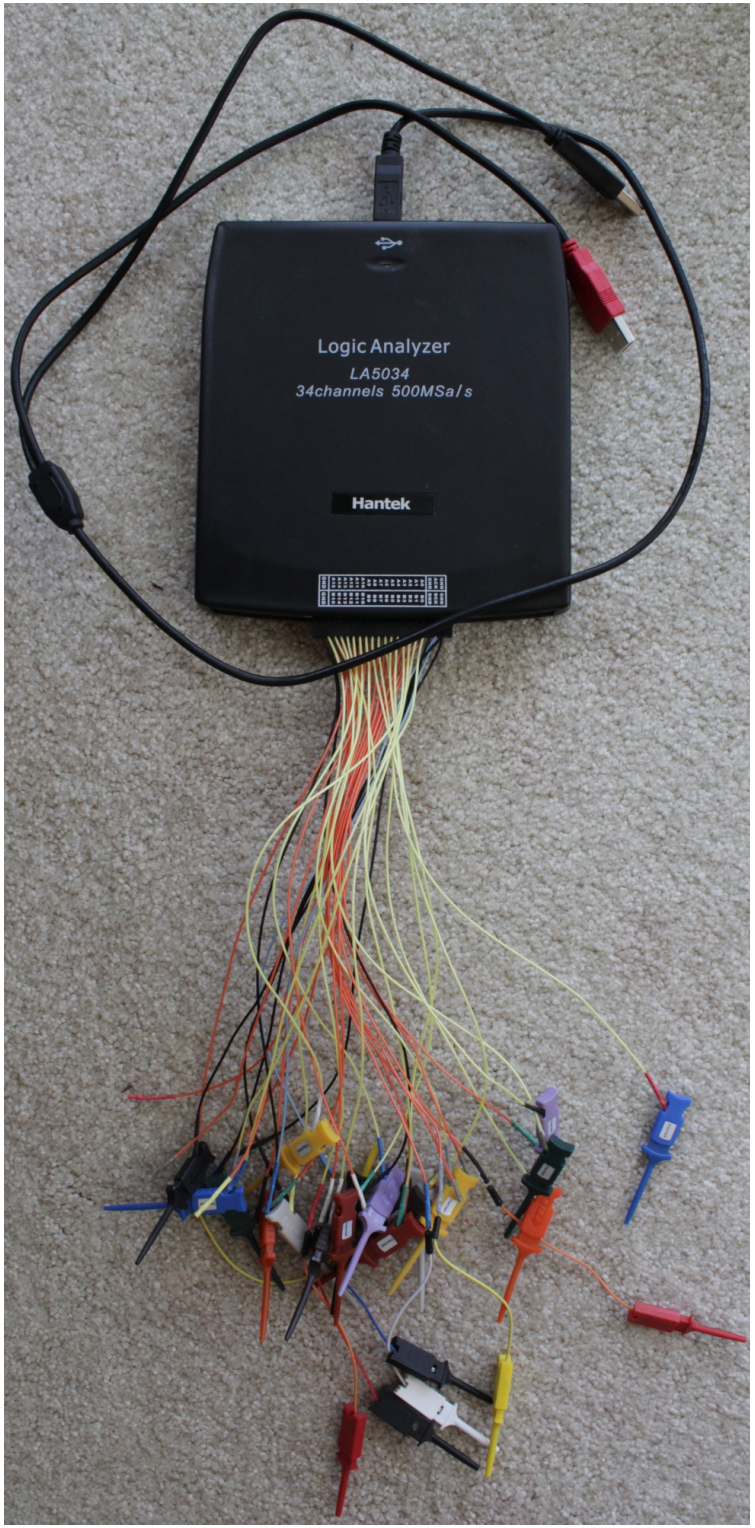
(μPD777 008 for Epoch Video Cassette “Pakpak Monster”)



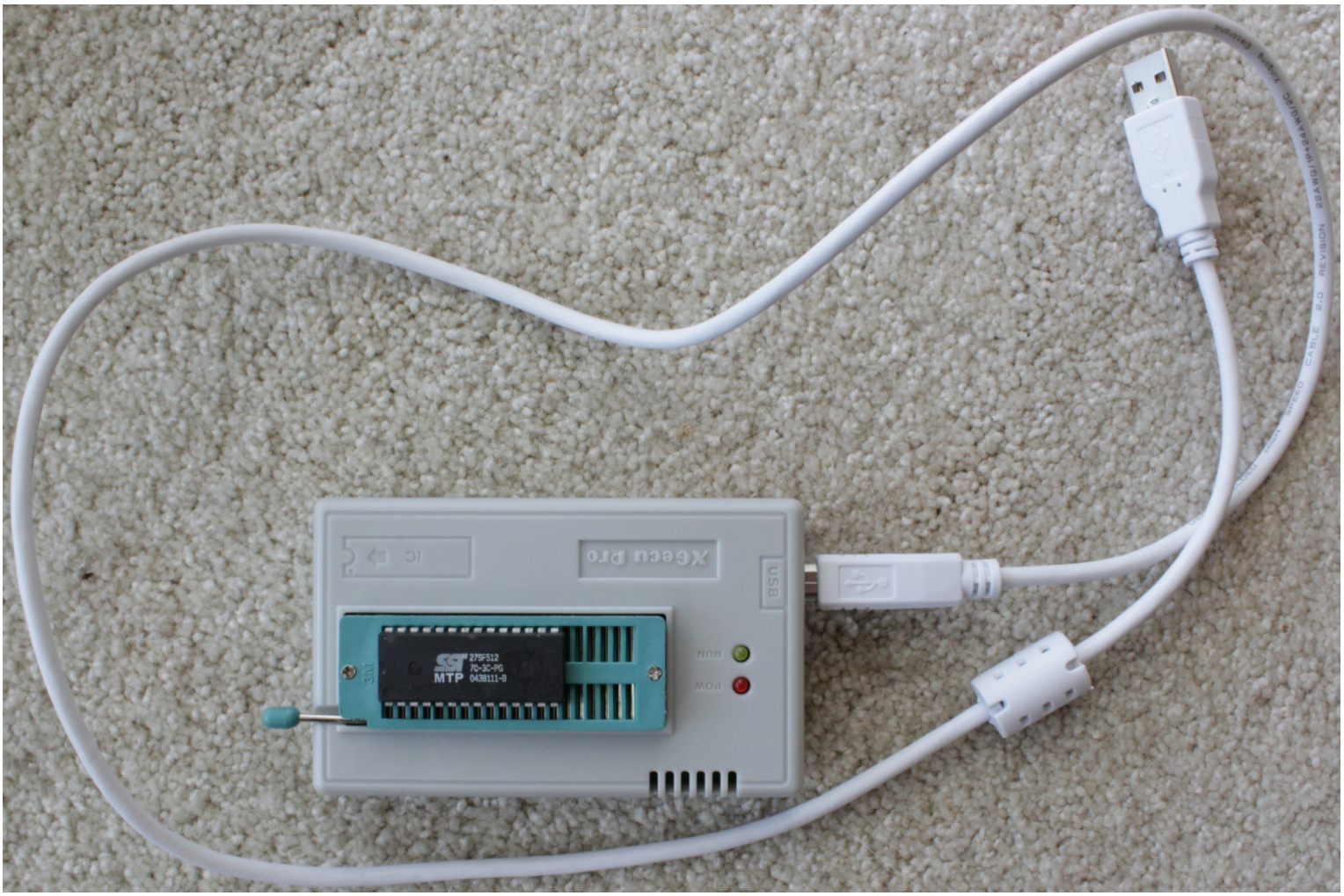


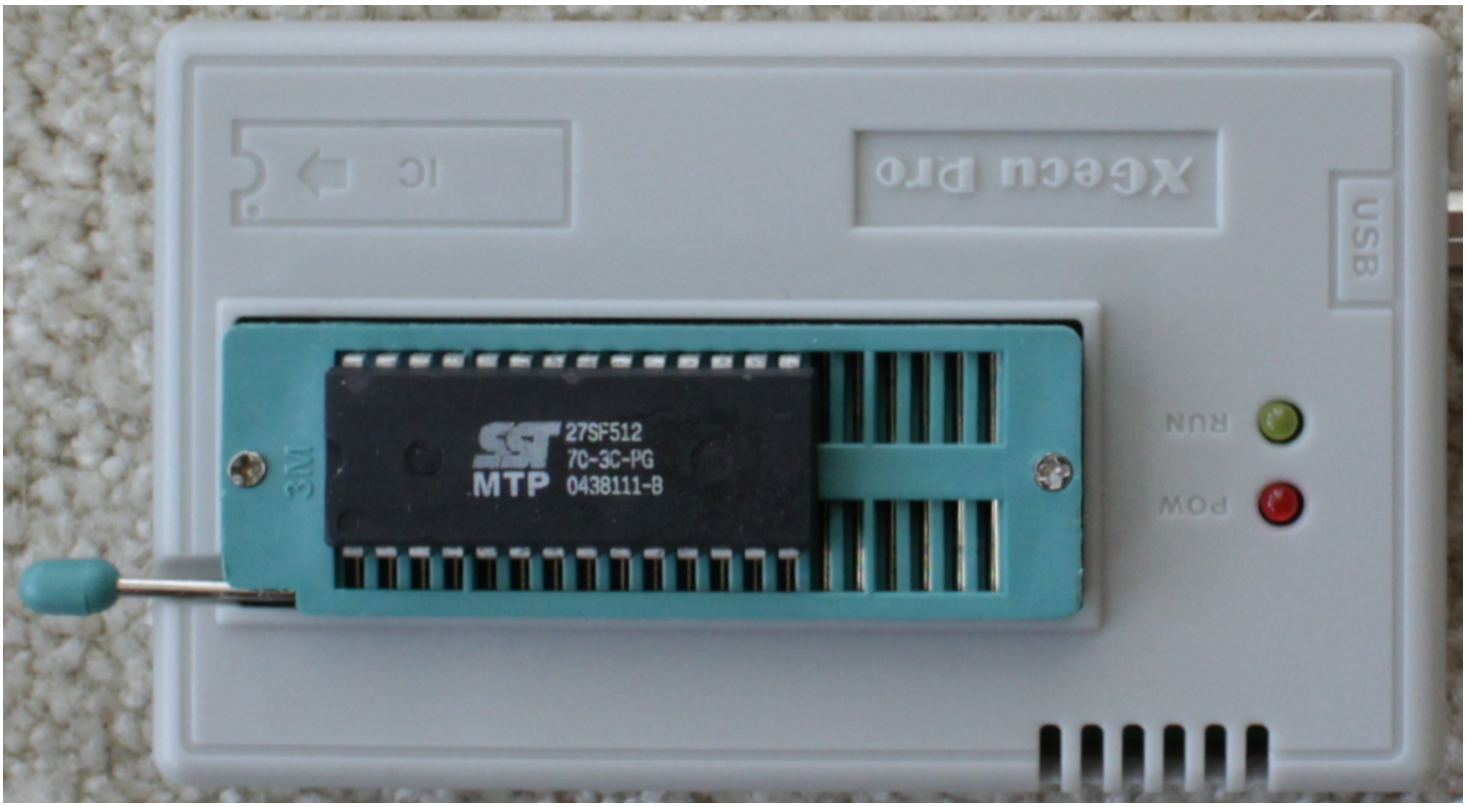
Pattern ROM Dump Wave Form of PAT[00h] (A3=00h) & PAT[01h] (A3=01h) under y=1 (A4=18h) through 7 (A4=78h) Captured by Logic Analyzer

(μPD777 009 for Epoch Video Cassette "Monster Mansion")



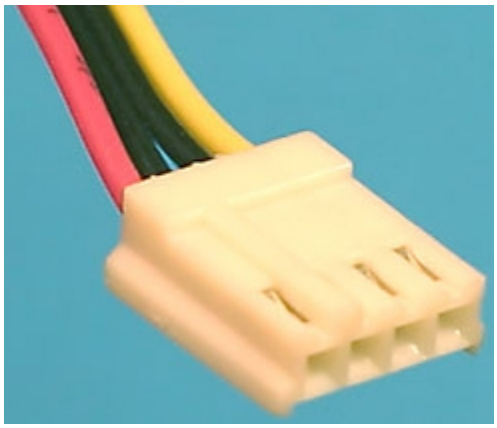
Hantek Logic Analyzer (34 channels, 500M Samples per Second)





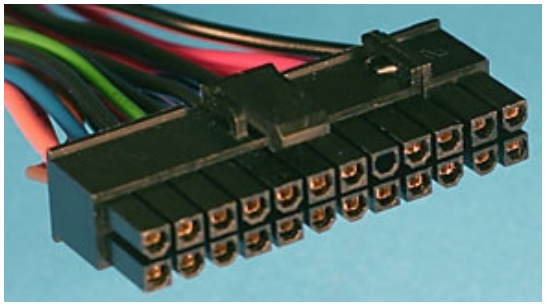
Xgecu TL866II Plus Programmer with 40 Pin Dual Inline Zero Insertion Force Socket

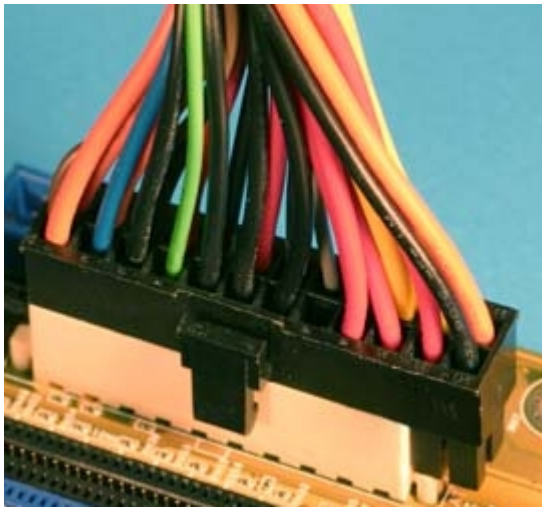
Power Supply for IBM PC



Pin #	Color	Power
1	Yellow	+12V
2	Black	0V
3	Black	0V
4	Red	+5V

4 Pin Floppy Disk Drive Power Cable Connector

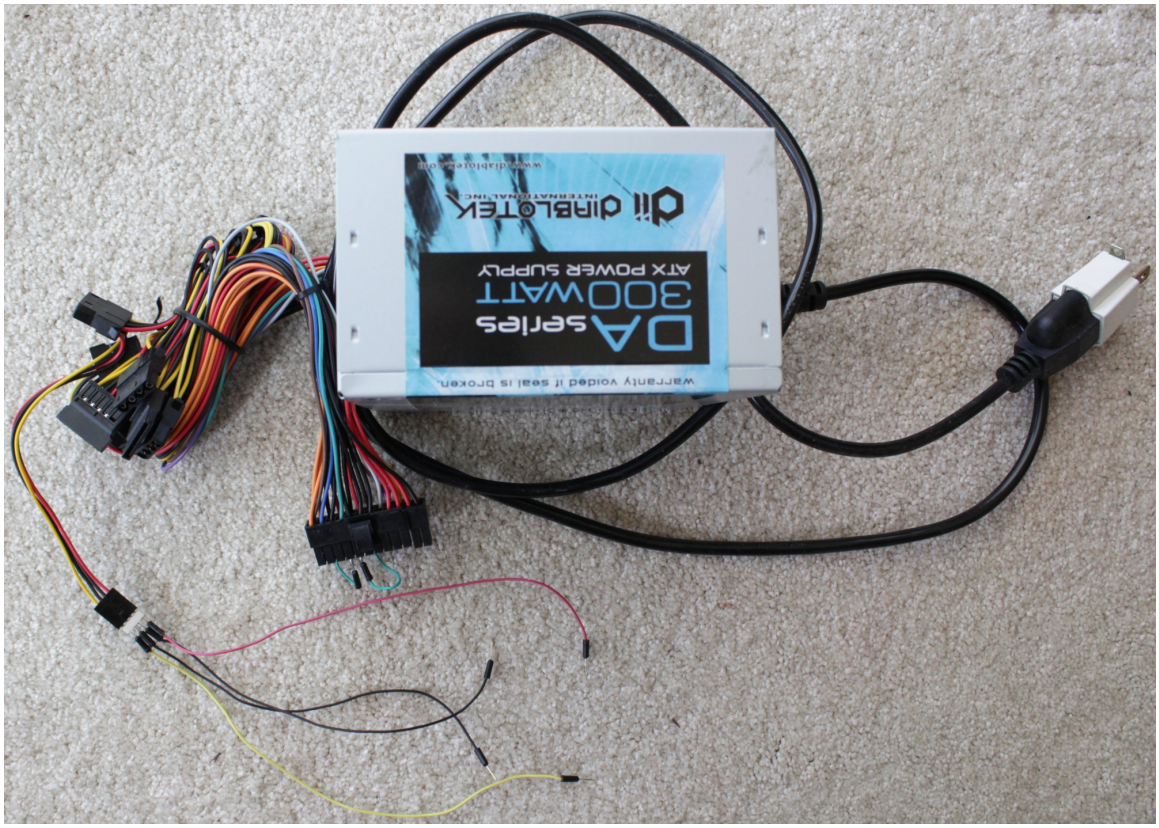




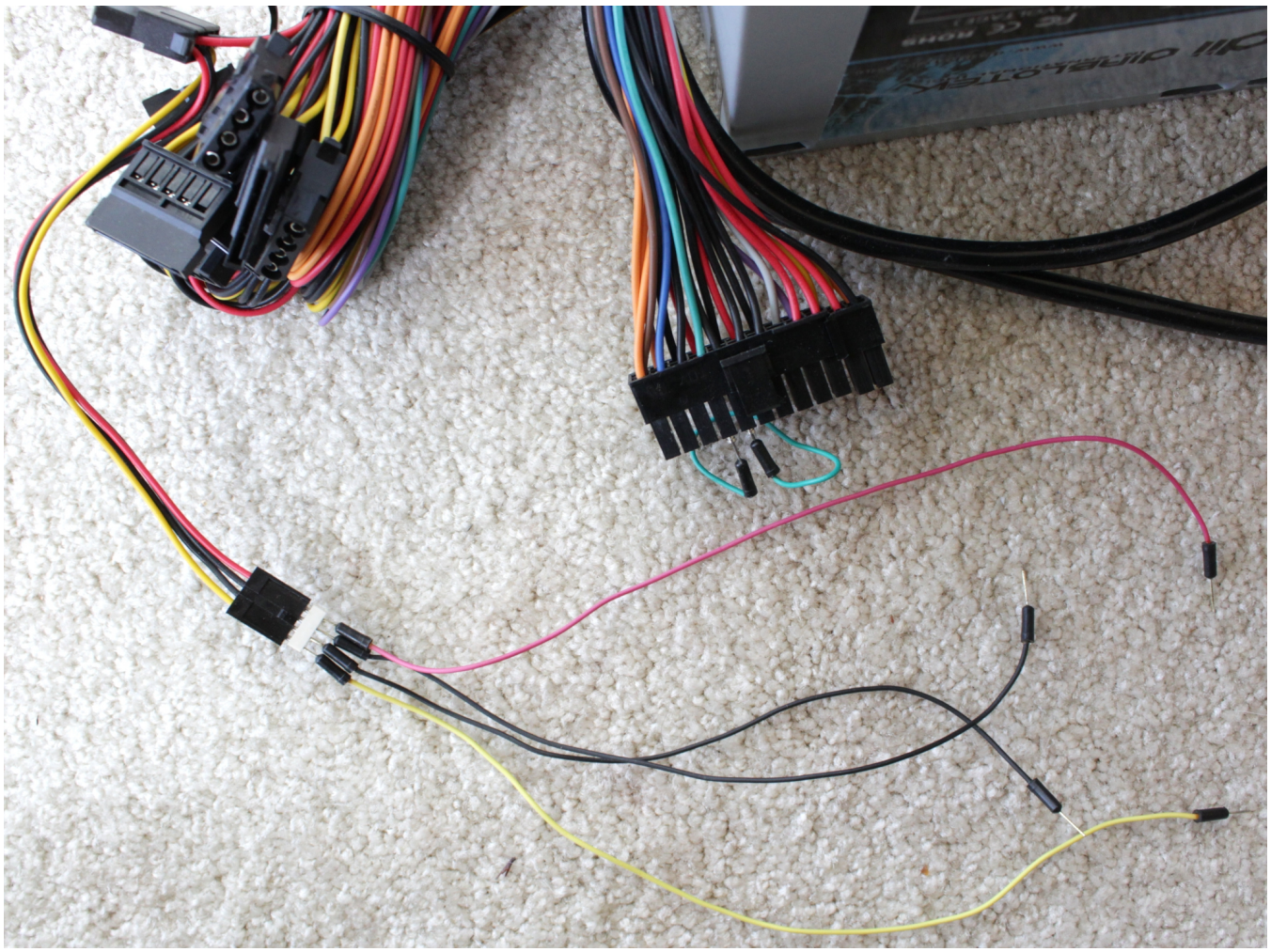
Pin #	Color	Power	Pin #	Color	Power
1	Orange	+3.3V	13	Orange	+3.3V
2	Orange	+3.3V	14	Blue	-12V
3	Black	0V	15	Black	0V
4	Red	+5V	16	Green	PS_ON#
5	Black	0V	17	Black	0V
6	Red	+5V	18	Black	0V
7	Black	0V	19	Black	0V
8	Gray	PWR_OK	20		
9	Purple	STBy +5V	21	Red	+5V
10	Yellow	+12V	22	Red	+5V
11	Yellow	+12V	23	Red	+5V
12	Orange	+3.3V	24	Black	0V

Power supply is enabled by shorting Pin 16 & 17.

24 Pin Main Power Cable Connector



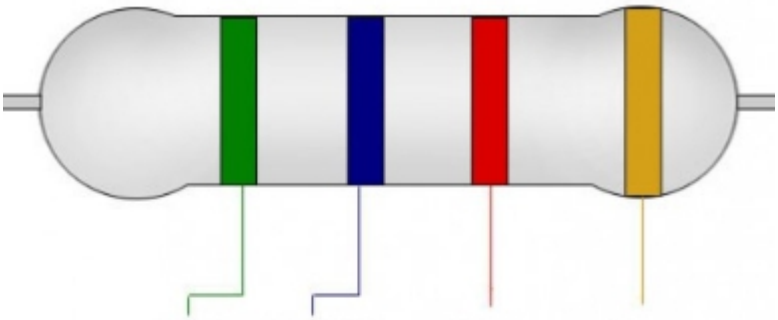
Applying a 300W ATX Power Supply for IBM PC



Shorting Pin 16 & 17 of 24 Pin Main Power Cable, +5V and +12V from Floppy Disk Drive Power Cable

Resistor Assortment & Color Codes

1Ω	1.2Ω	1.5Ω	1.8Ω	2.2Ω	2.7Ω	3.3Ω	3.9Ω	4.7Ω	5.6Ω	6.8Ω	8.2Ω
10Ω	12Ω	15Ω	18Ω	22Ω	27Ω	33Ω	39Ω	47Ω	56Ω	68Ω	82Ω
100Ω	120Ω	150Ω	180Ω	220Ω	270Ω	330Ω	390Ω	470Ω	560Ω	680Ω	820Ω
1kΩ	1.2kΩ	1.5kΩ	1.8kΩ	2.2kΩ	2.7kΩ	3.3kΩ	3.9kΩ	4.7kΩ	5.6kΩ	6.8kΩ	8.2kΩ
10kΩ	12kΩ	15kΩ	18kΩ	22kΩ	27kΩ	33kΩ	39kΩ	47kΩ	56kΩ	68kΩ	82kΩ
100kΩ	120kΩ	150kΩ	180kΩ	220kΩ	270kΩ	330kΩ	390kΩ	470kΩ	560kΩ	680kΩ	820kΩ
1MΩ	1.2MΩ	1.5MΩ	1.8MΩ	2.2MΩ	2.7MΩ	3.3MΩ	3.9MΩ	4.7MΩ	5.6MΩ	6.8MΩ	8.2MΩ



	First Digit	Second Digit	Multiplier	Tolerance
Black	Nil	0	1	Nil
Brown	1	1	10	±1%
Red	2	2	100	±2%
Orange	3	3	1000	±3%
Yellow	4	4	10000	±4%
Green	5	5	100000	±0.5%
Blue	6	6	1M	±0.25%
Violet	7	7	10M	±0.10%
Grey	8	8	100M	±0.05%
White	9	9	1G	Nil
Gold	Nil	Nil	÷10	±5%
Silver	Nil	Nil	÷100	±10%

NEC µPD777 Related PDF Files (Extracted)

- 777 Design Note
- Epoch Cassette Vision Cartridge
- Anatomy of Cassette Vision
- 777 to YUV System Design