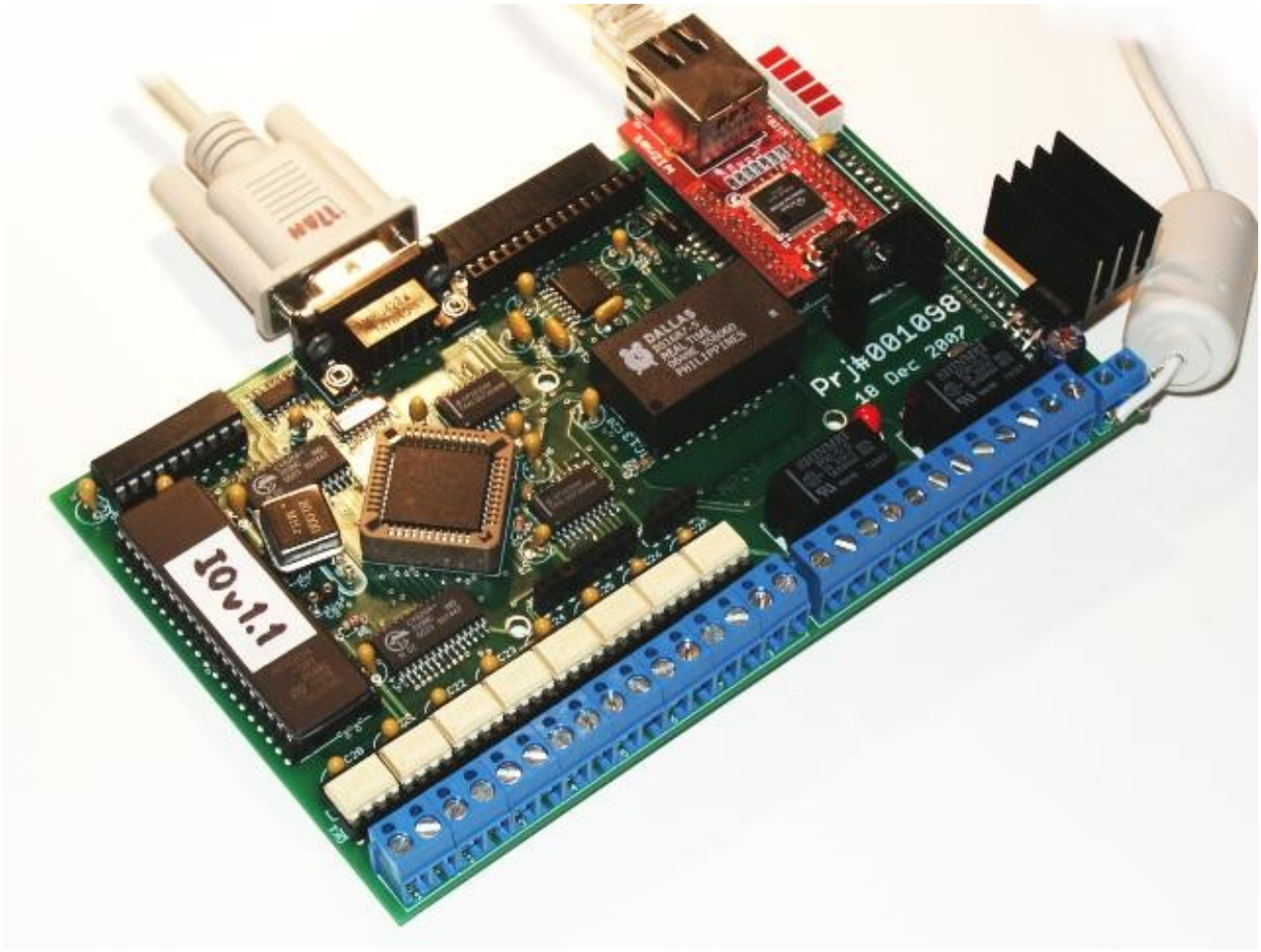
"Remote digital I/O board"

Alexander Shabarshin

http://www.nedopc.org

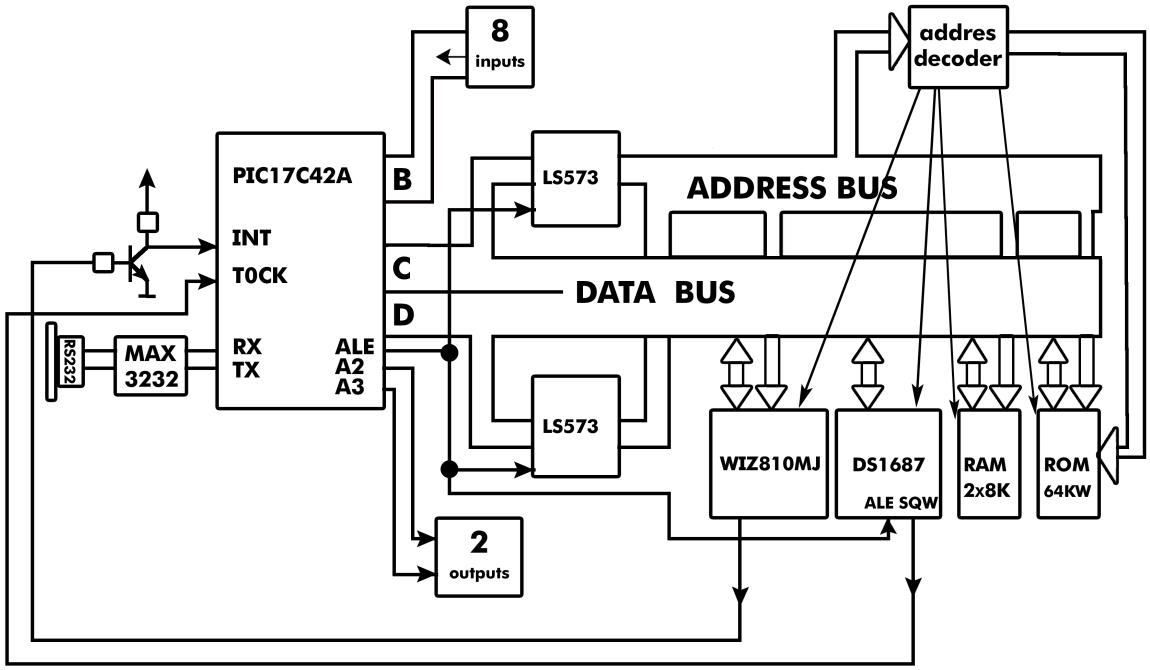
PIXPUTER PROTOTYPE

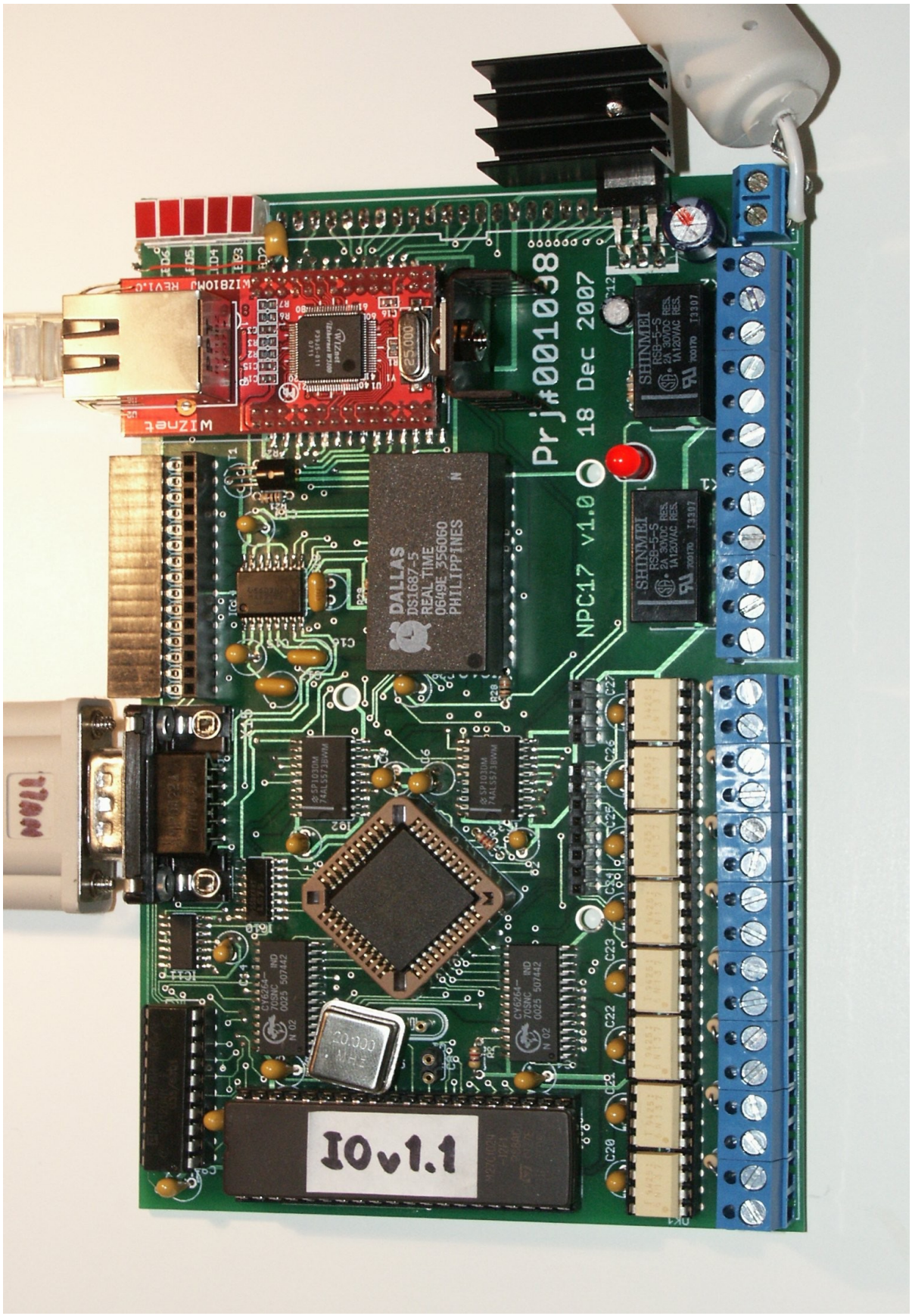


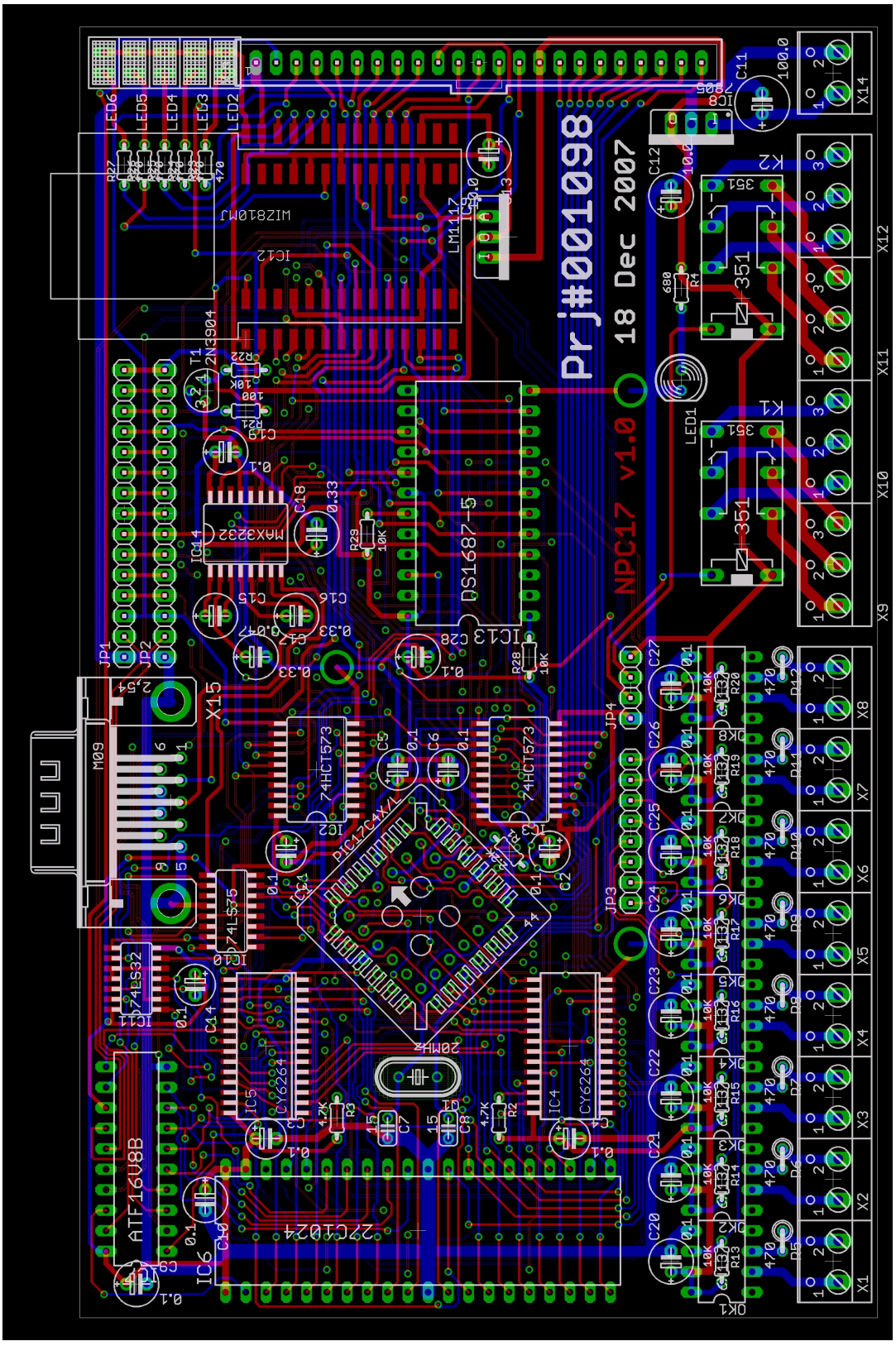
2008

Digital I/O-board with memory (2048 x 8 digital samples with full timestamps for each). Simple Web-interface to get collected data and to control the board (set time, change sample frequency, control outputs) through network. CPU will be PIC17C44 in microprocessor mode with external memory. Onboard RTC chip (real time clock) for collecting timestamps. I/O channels: 8 opto-isolated inputs and 2 high-voltage relays as outputs.

The Digital I/O-board is designed around plug-in module **WIZ810MJ** (with chip WIZnet **W5100**) and microcontroller PIC17C4X (here **PIC17C42A**, but it can be easily changed to PIC17C43 or PIC17C44) in microprocessor mode (it is default mode for fresh not programmed chip). Main purpose of this board is remotely accessible digital inputs and outputs (here 8 optoisolated inputs and 2 high-voltage outputs). HTTP is chosen as network protocol for connection to this board because it's supported in many systems by default. Real-time clock (RTC) DS1687 is used to support timestamps (with precision of seconds). Also RTC provides periodical interrupts for data sampling (with frequency from 1 Hz to 128 Hz) and also gives us ability to save some data (as network settings) to not-volatile memory (NVM) that is kept in power off mode because of embedded battary. The board has 2 x CY6264 RAM (8K) that is used for saving data samples itself with time stamps (8 bytes per sample) in power on mode only. Program is written to one ultraviolet eraseble ROM IC M27C1024 (64Kwords). Address decoder cosnsists of 16V8 reprogrammable IC and couple of small TTL chips. Also RS-232 is used for settings and debug from PC (IC MAX3232) through NULL-modem cable:

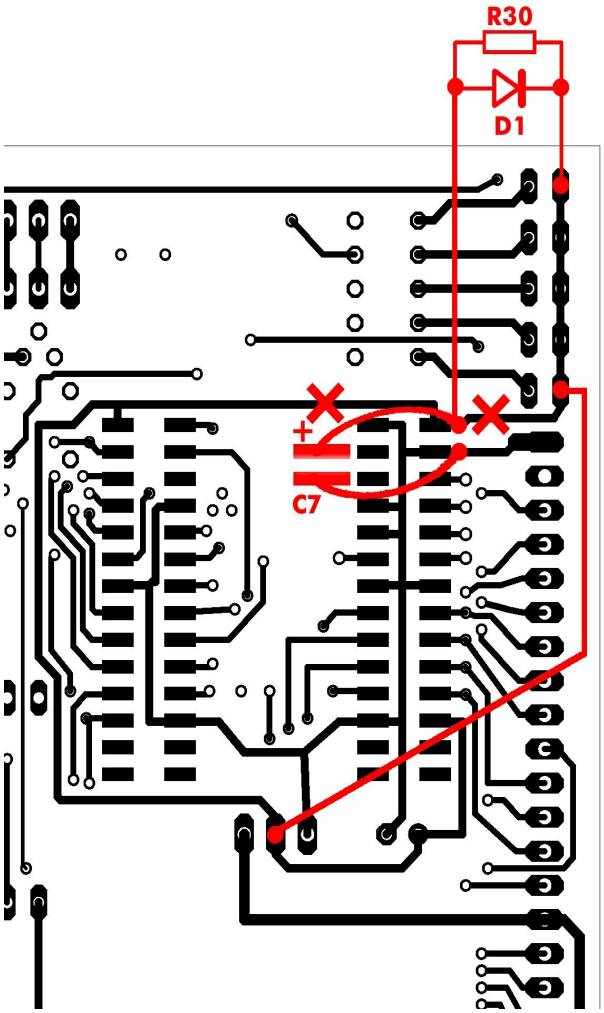


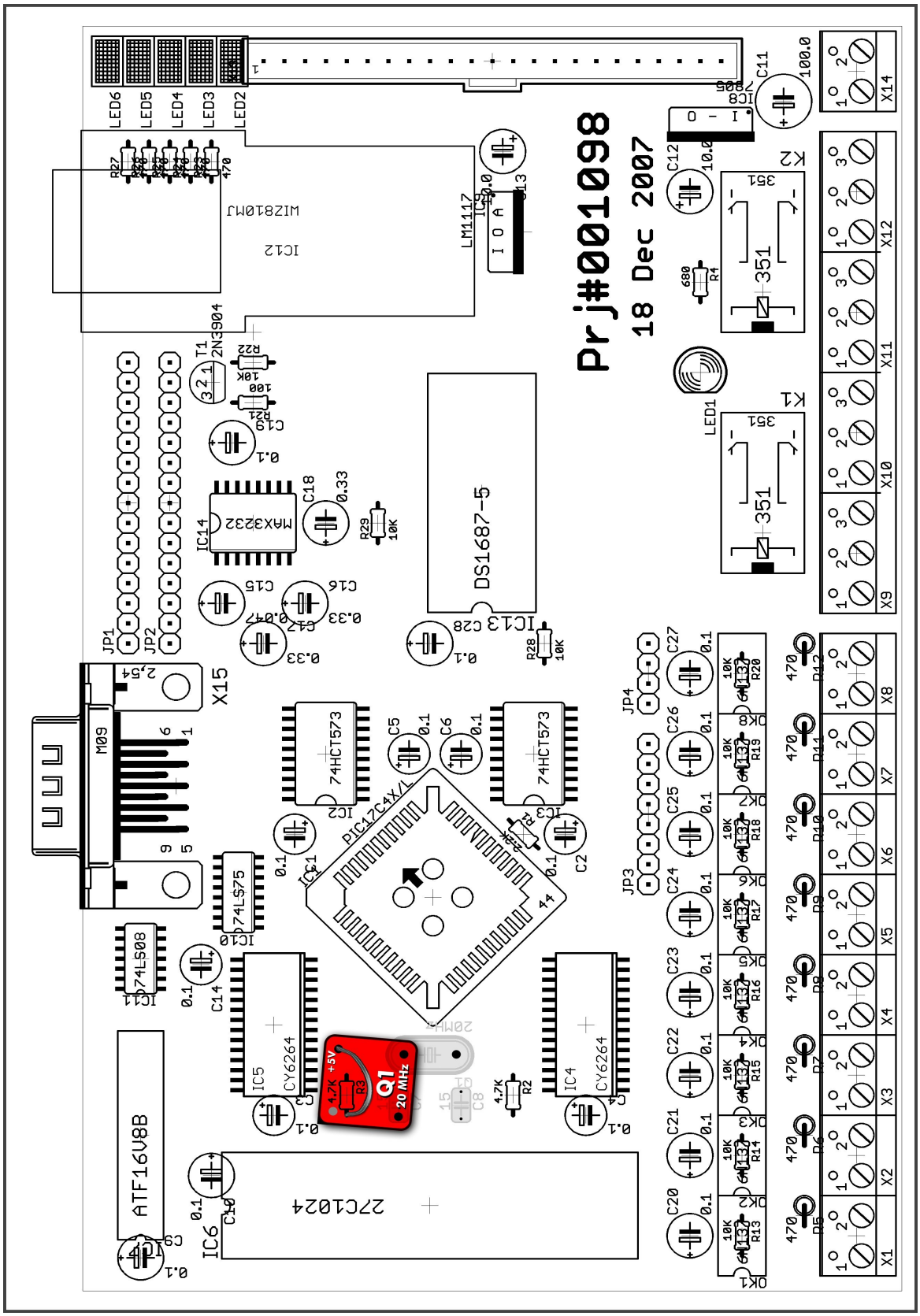

Prj#001038
18 Dec 2007
NPC17 v1.0
DALLAS
DS1687-5
REAL TIME
PHILIPPINES
SHINMEI
WIZ810MJ REV1.0
WIZnet
25.000
20.000 MHz
IOv1.1

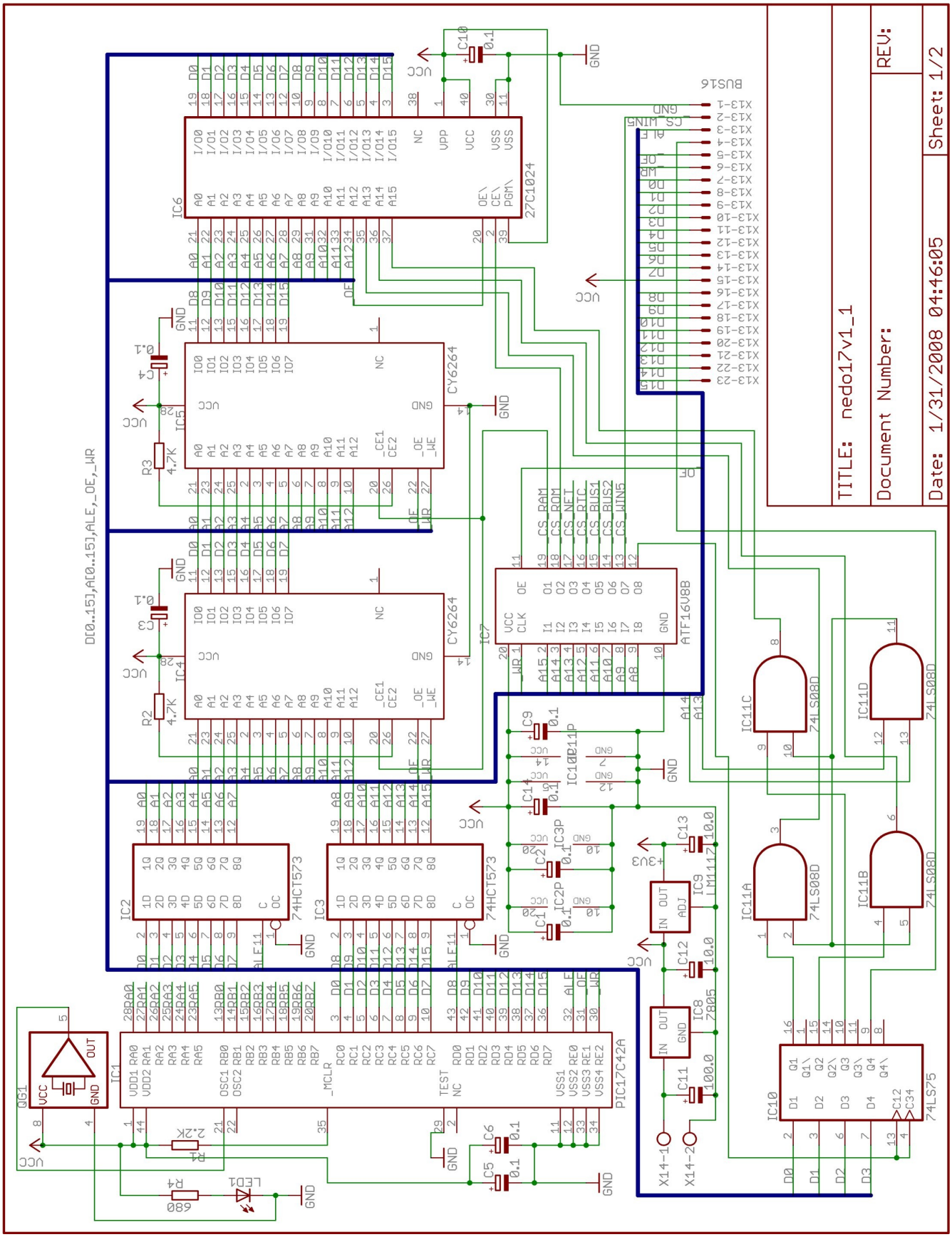

Prj#001098
18 Dec 2007
NPC17 v1.0
DS1687-5
PIC17C44
74HCT573
74LS32
74LS75
ATF16V8B
CY6264
27C1024
MAX232
LM1117
2N3904
WIZ810MJ
20MHz

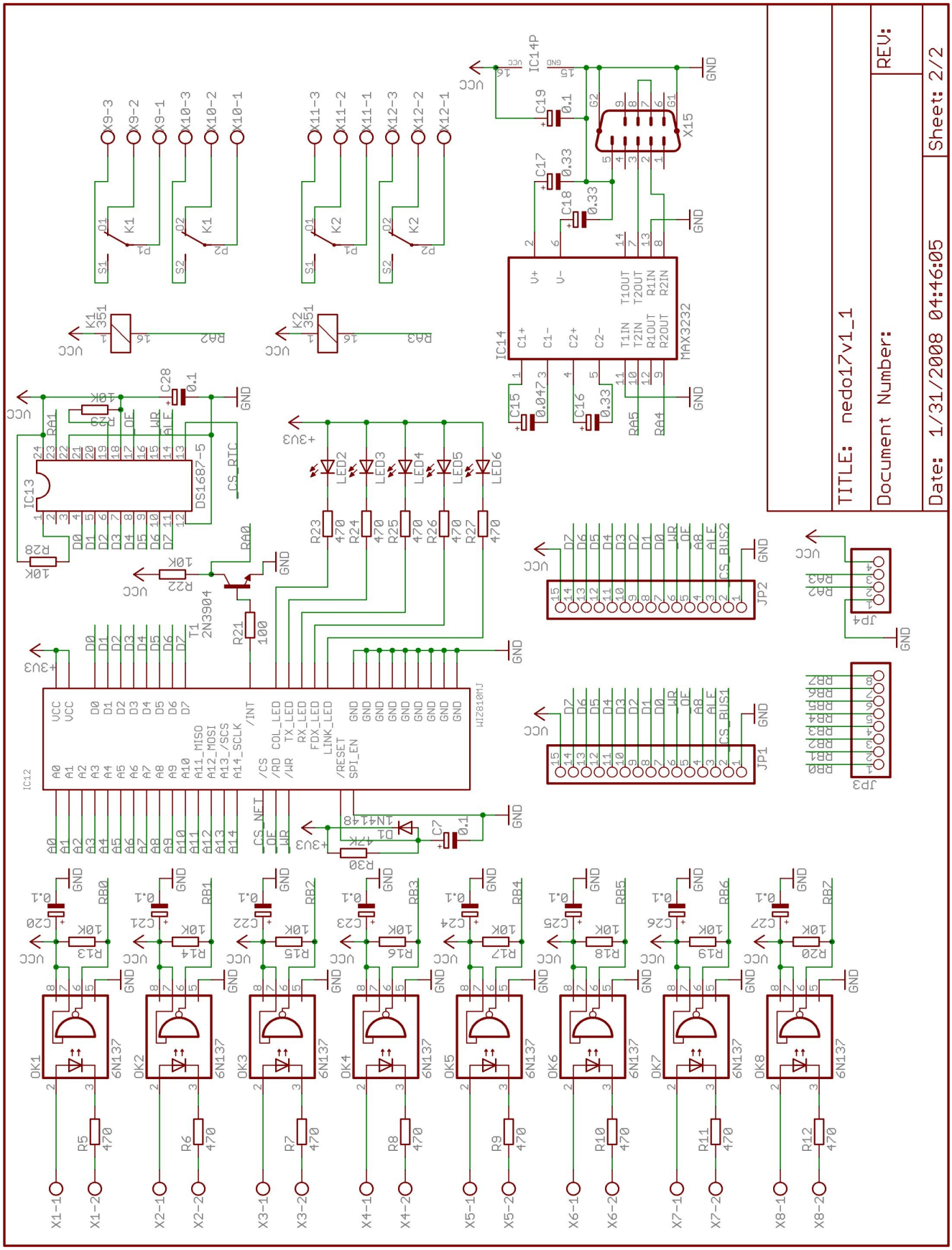
Printed Circuit Board (PCB)

PCB was designed specially for this project, but it's version 1.0 of design with couple of errors. To make it compatible with v1.1 (TTL oscillator instead of crystal and RC-reset for WIZnet) some modifications have to be done (see schematics for details):




Prj#001098
18 Dec 2007
LED6
LED5
LED4
LED3
LED2
WIZ810MJ
IC12
LM1117
IC9
2N3904
T1
MAX232
IC14
DS1687-5
IC13
74HCT573
IC2
74HCT573
IC3
PIC42C4X/
IC1
74LS75
IC10
74LS08
IC11
CY6264
IC5
CY6264
IC4
27C1024
IC6
ATF16V8B
IC7
7805
Q1
20 MHz
X15
M09
JP1
JP2
JP3
JP4
K1
K2
351
LED1
X1
X2
X3
X4
X5
X6
X7
X8
X9
X10
X11
X12
X14


TITLE: nedo17v1_1
Document Number:
REV:
Date: 1/31/2008 04:46:05
Sheet: 1/2
PIC17C42A
27C1024
CY6264
74HCT573
ATF16V8B
74LS08D
74LS75
LM1117-3.0
LM1117-10.0
7805
BUS16
D[0..15],A[0..15],ALE,_OE,_WR


TITLE: nedo17v1_1
Document Number:
REV:
Date: 1/31/2008 04:46:05
Sheet: 2/2
IC14P
IC14
MAX3232
IC13
DS1687-5
IC12
WIZ810MJ
T1
2N3904
JP1
JP2
JP3
JP4
X15
OK1
OK2
OK3
OK4
OK5
OK6
OK7
OK8
6N137
K1
K2
LED2
LED3
LED4
LED5
LED6
1N4148
CS_RTC
CS_NET
CS_BUS1
CS_BUS2
+3V3
VCC
GND

RS-232 commands for I/O board:

There are 3 commands for this board: ? - read, ! - write, $ - execute

After command you should specify hexadecimal address, then '=' and then one or more words in hexadecimal form. To set network settings in NVM you should do these:

Set default port for listening (80):

!300E=00000050

Set flag for frequency (see later):

!3010=00FF

Set gateway address (for example 10.19.73.1):

!3011=000A001300490001

Set subnet mask (for example 255.255.255.0):

!3015=00FF00FF00FF0000

Set MAC address (for exampe AA:AA:0A:13:49:0A):

!3019=00AA00AA000A00130049000A

Set IP address (for example 10.19.73.10):

!301F=000A00130049000A

Frequencies for data sampling (every 2nd SQW from RTC)

Flag 0x3010 value	Data sampling rate	Data sampling frequency
0x00	Data sampling and network disabled	Data sampling and network disabled
0x01	*7.8125 ms*	*128 Hz*
0x02	*15.625 ms*	*64 Hz*
0x03	244.141us	4096 Hz
0x04	488.281 us	2048 Hz
0x05	976.5625 us	1024 Hz
0x06	1.953125 ms	512 Hz
0x07	3.90625 ms	256 Hz
0x08	7.8125 ms	128 Hz
0x09	15.625 ms	64 Hz
0x0A	31.25 ms	32 Hz
0x0B	62.5 ms	16 Hz
0x0C	125 ms	8 Hz
0x0D	250 ms	4 Hz
0x0E	500 ms	2 Hz
0x0F	1 s	1 Hz

HTTP requests

HTTP requests for this board:

http://IP – request for all available data (2048 samples max)

http://IP/HHHH – request for specified number of records (HHHH is hexadecimal)

http://IP/s1 – switch on first relay

http://IP/c1 – switch off first relay

http://IP/s2 – switch on second relay

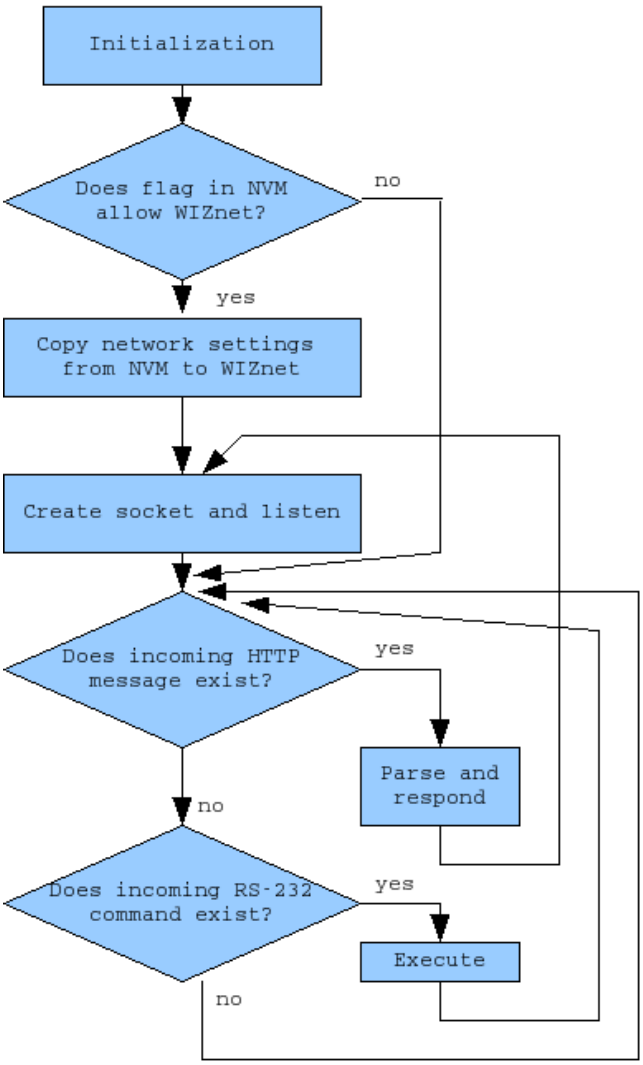
http://IP/c2 – switch off second relay

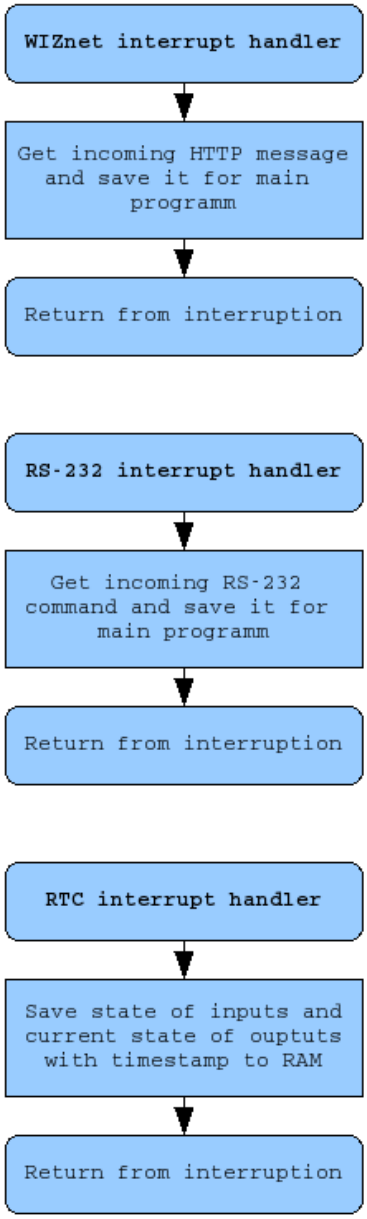
Example of data (**YYMMDDhhmmssiio**, where YY-year, MM-month, DD-date, hh-hours, mm-minutes, ss-seconds, ii-inputs, o-ouptuts):

080131223502FF0

080131223501FF0

080131223500FF0


Initialization
Does flag in NVM allow WIZnet?
no
yes
Copy network settings from NVM to WIZnet
Create socket and listen
Does incoming HTTP message exist?
yes
Parse and respond
no
Does incoming RS-232 command exist?
yes
Execute
no


WIZnet interrupt handler
Get incoming HTTP message and save it for main programm
Return from interruption
RS-232 interrupt handler
Get incoming RS-232 command and save it for main programm
Return from interruption
RTC interrupt handler
Save state of inputs and current state of ouptuts with timestamp to RAM
Return from interruption

ioboard.asm

```
; ioboard.asm for project 001098

        processor 17C42A
       radix dec

include "p17c42a.inc"
include "pixmacro.inc"
include "wiznet.inc"

RAMBAH  equ    0x40
ROMBAH  equ    0x60
REG3ADR equ    0x37F3
RAMBADR equ    RAMBAH<<8
ROMBADR equ    ROMBAH<<8
NVMPHI  equ    0x300E
NVMPLO  equ    0x300F
NVMINET equ    0x3011
WIZINET equ    0x8001
PAGSIZE equ    0x2000
BUFINIT equ    0x80
BUFSIZE equ    0x80

; RMSR = 0x03
RXSIZE  equ    8192
RXMASK  equ    0x1FFF
; TMSR = 0x03
TXSIZE  equ    8192
TXMASK  equ    0x1FFF

temp_w  equ    0x1A
temp_s  equ    0x1B
temp_b  equ    0x1C
temp_p  equ    0x1D
temp_f  equ    0x1E
iperif  equ    0x1F
count0  equ    0x20
count1  equ    0x21
count2  equ    0x22
counter equ    0x23
testnum equ    0x24
hibyte  equ    0x25
```

```
bytel  equ    0x26
byteh  equ    0x27
tempch equ    0x28

stat1  equ    0x34
stat2  equ    0x35
isqwe  equ    0x36
isqwe1 equ    0x37
isqwe2 equ    0x38
isqwe3 equ    0x39
dstahi equ    0x3A
dstalo equ    0x3B
temp3  equ    0x3C
temp4  equ    0x3D
temp5  equ    0x3E
tmphex equ    0x3F
fzero  equ    0x40
temp   equ    0x41
asrchi equ    0x42
asrclo equ    0x43
adsthi equ    0x44
adstlo equ    0x45
ncount equ    0x46
nerr   equ    0x47
rtemp  equ    0x48
htemp  equ    0x49
gethi  equ    0x4A
getlo  equ    0x4B
qtemp  equ    0x4C
counta  equ   0x4D
countb equ    0x4E
countc equ    0x4F
counth equ    0x50
countl equ    0x51
hibyte2 equ   0x52
stemp  equ    0x53
curhi  equ    0x54
curlo  equ    0x55
temp_h equ    0x56
temp_l equ    0x57
freeh  equ    0x58
freel  equ    0x59
temp1  equ    0x5A
```

```
temp2   equ    0x5B
stahi   equ    0x5C
stalo   equ    0x5D
wrhi    equ    0x5E
wrlo    equ    0x5F
itemp   equ    0x60
ireg    equ    0x61
iadrhi  equ    0x62
iadrlo  equ    0x63
isizhi  equ    0x64
isizlo  equ    0x65
ioffhi  equ    0x66
iofflo  equ    0x67
istahi  equ    0x68
istalo  equ    0x69
irdhi   equ    0x6A
irdlo   equ    0x6B
itmphi  equ    0x6C
itmplo  equ    0x6D
ilefhi  equ    0x6E
ileflo  equ    0x6F
iflag   equ    0x70
iregg   equ    0x71

ibuf    equ    0x120
itmp    equ    0x121
iwhi    equ    0x122
iwlo    equ    0x123
ichi    equ    0x124
iclo    equ    0x125

delay1s macro
        local  D1
        movlw  100
        movwf  counter
D1:     delay10ms
        decfsz counter,f
        goto   D1
        endm

delay100ms macro
        local  D1
        movlw  10
```

```
        movwf   counter
D1:     delay10ms
        decfsz  counter,f
        goto    D1
        endm

delay10ms macro
        delay2  64,count1,count2
        clrwdt
        endm

memtst  macro   A,V,L
        memr1n  A
        movwf   temp
        movlw   V
        cpfseq  temp
        goto    L
        endm

        org 0x0000
_start:
        goto main

        org 0x0008
_int:   movpf   WREG,temp_w
        movpf   PCLATH,temp_p
        call    exint
        movpf   temp_p,PCLATH
        movpf   temp_w,WREG
        retfie

        org 0x0010
        retfie

        org 0x0018
_t0cki: movpf   WREG,temp_w
        movpf   PCLATH,temp_p
        call    extoki
        movpf   temp_p,PCLATH
        movpf   temp_w,WREG
        retfie

        org 0x0020
```

```
_perif: movpf  WREG,temp_w
        movpf  ALUSTA,temp_s
        movpf  BSR,temp_b
        movpf  FSR1,temp_f
        movpf  PCLATH,temp_p
        clrf   PCLATH,f
        ; Autoincrement for FSR1
         bcf     ALUSTA,FS3
        bsf    ALUSTA,FS2
        movlr  1
        movfp  ibuf^0x100,FSR1
        clrf   WREG,f
        cpfsgt FSR1
        goto   __perif1
__perif0:
        clrwdt
        movlb  1
        btfss  PIR^0x100,RCIF
        goto   __perif1
        movlb  0
        movpf  RCREG,iperif
        movlw  0x0A
        cpfseq iperif
        goto   __perif01
        clrf   WREG,f
        movfp  WREG,INDF1
        movfp  WREG,FSR1
        goto   __perif1
__perif01:
        movlw  0x1F
        cpfsgt iperif
        goto   __perif0
        movfp  iperif,INDF1
        goto   __perif0
__perif1:
        movpf  FSR1,ibuf^0x100
        movfp  temp_p,PCLATH
        movfp  temp_f,FSR1
        movfp  temp_w,WREG
        movfp  temp_s,ALUSTA
        movfp  temp_b,BSR
        retfie
```

```
extoki: movpf   ALUSTA,temp_s
        movpf   BSR,temp_b
        movlr   0
        movpf   TBLPTRH,temp_h
        movpf   TBLPTRL,temp_l
        ; increment counter and pass every 2nd interrupt
        incf    isqwe,f
        btfsc   isqwe,0
        goto    sqwe0
        movlw   HIGH(ROMBADR)
        cpfseq  curhi
        goto    sqwe1
        movlw   LOW(ROMBADR)
        cpfseq  curlo
        goto    sqwe1
        savew   RAMBADR,curhi,curlo
sqwe1:  memr1n  0x3009 ; read year
        movwf   isqwe3
        memr1n  0x3008 ; read month
        memwf   isqwe3,curhi,curlo
        incwf   curhi,curlo
        memr1n  0x3007 ; read date
        movwf   isqwe3
        memr1n  0x3004 ; read hours
        memwf   isqwe3,curhi,curlo
        incwf   curhi,curlo
        memr1n  0x3002 ; read minutes
        movwf   isqwe3
        memr1n  0x3000 ; read seconds
        memwf   isqwe3,curhi,curlo
        incwf   curhi,curlo
        movpf   PORTB,isqwe3
        clrf    WREG,f
        tstfsz  stat1
        iorlw   0x01
        tstfsz  stat2
        iorlw   0x02
        memwf   isqwe3,curhi,curlo
        incwf   curhi,curlo
sqwe0:  movfp   temp_l,TBLPTRL
        movfp   temp_h,TBLPTRH
        movfp   temp_b,BSR
        movfp   temp_s,ALUSTA
```

```
        return ; retfie above

exint:  movpf   ALUSTA,temp_s
        movpf   BSR,temp_b
        movpf   FSR1,temp_f
        movlr   0
        movpf   TBLPTRH,temp_h
        movpf   TBLPTRL,temp_l
        ; Autoincrement for FSR1
         bcf     ALUSTA,FS3
        bsf     ALUSTA,FS2
        ; check nature of interrupt
        memr1n  IR
        movwf   iregg
        btfss   iregg,0
        goto    intend
        ; read interrupt register for socket 0
        memr1n  S0_IR
        movwf   ireg
        ; check for RECV interruption
        btfss   ireg,2
        goto    intend
        ; get out if buffer still busy
        saveb   BUFINIT,FSR1
        ; get received size, also save it as left size
        memr1n  S0_RX_RSR0
        movwf   isizhi
        movwf   ilefhi
        memr1n  S0_RX_RSR0+1
        movwf   isizlo
        movwf   ileflo
        ; calculate offset and save RD-address for future
        memr1n  S0_RX_RD0
        movwf   irdhi
        andlw   HIGH(RXMASK)
        movwf   ioffhi
        memr1n  S0_RX_RD0+1
        movwf   irdlo
        andlw   LOW(RXMASK)
        movwf   iofflo
        ; calculate start address
        savew   IINCHIP_MAP_RXBUF,istahi,istalo
        addwff  istahi,istalo,ioffhi,iofflo
```

```
        ; save threshold for buffer
        savew   IINCHIP_MAP_RXBUF+RXSIZE,itmphi,itmplo
inorm:  memr1f  istahi,istalo
        movwf   INDF1
        decwf   ilefhi,ileflo
        ; check for end of data
        movfp   ilefhi,WREG
        iorwf   ileflo,w
        btfsc   ALUSTA,Z
        goto    irdok
        incwf   istahi,istalo
        ; check for buffer overflow
        movfp   istahi,WREG
        cpfseq  itmphi
        goto    inorm2
        movfp   istalo,WREG
        cpfseq  itmplo
        goto    inorm2
        ; buffer overflow
        savew   IINCHIP_MAP_RXBUF,istahi,istalo
inorm2: ; check for end of our buffer
        tstfsz  FSR1
        goto    inorm
irdok:  ; our buffer filled
        addwff  irdhi,irdlo,isizhi,isizlo
        subwff  irdhi,irdlo,ilefhi,ileflo
        movfp   irdhi,WREG
        memw1n  S0_RX_RD0
        movfp   irdlo,WREG
        memw1n  S0_RX_RD0+1
        memw1d  Sn_CR_RECV,S0_CR ; should we do this here? we got everything already...
        ; signal for main program
        saveb   1,iflag
        ; disable socket 0 interrupts
        memw1d  0x00,IMR ; ???
intend: movfp   ireg,WREG
        andlw   0x0F
        memw1n  S0_IR
        movfp   iregg,WREG
        memw1n  IR
        movfp   temp_l,TBLPTRL
        movfp   temp_h,TBLPTRH
        movfp   temp_f,FSR1
```

```
        movfp   temp_b,BSR
        movfp   temp_s,ALUSTA
        return ; retfie see above

; send buffer through WizNET (interrupts must be disabled!)
sendbuf:
        movlw   BUFINIT
        subwf   FSR0,w
        movwf   countl
        movwf   countc
        clrf    counth,f
sendbuf0:
        ; get free size
        memr1n  S0_TX_FSR0
        movwf   freeh
        memr1n  S0_TX_FSR0+1
        movwf   freel
        saveb   BUFINIT,FSR0
        clrwdt
        movff   freeh,temp1
        movff   freel,temp2
        subwff  temp1,temp2,counth,countl
        btfsc   temp1,7
        goto    sendbuf0
        ; calculate offset and save WR-address for future
        memr1n  S0_TX_WR0
        movwf   wrhi
        andlw   HIGH(TXMASK)
        movwf   temp1
        memr1n  S0_TX_WR0+1
        movwf   wrlo
        andlw   LOW(TXMASK)
        movwf   temp2
        ; calculate start address
        savew   IINCHIP_MAP_TXBUF,stahi,stalo
        addwff  stahi,stalo,temp1,temp2
        ; save threshold for buffer
        savew   IINCHIP_MAP_TXBUF+TXSIZE,counta,countb
sendbuf1:
        movpf   INDF0,WREG
        memw1f  stahi,stalo
        incwf   stahi,stalo
        ; check for buffer overflow
```

```
        movfp   stahi,WREG
        cpfseq  counta
        goto    sendbuf2
        movfp   stalo,WREG
        cpfseq  countb
        goto    sendbuf2
        ; buffer overflow
        savew   IINCHIP_MAP_TXBUF,stahi,stalo
sendbuf2:
        ; check for end of our buffer
        decfsz  countc,f
        goto    sendbuf1
        ; our buffer copied
        addwff  wrhi,wrlo,counth,countl
        movfp   wrhi,WREG
        memw1n  S0_TX_WR0
        movfp   wrlo,WREG
        memw1n  S0_TX_WR0+1
        memw1d  Sn_CR_SEND,S0_CR
        saveb   BUFINIT,FSR0
        return

; copy HTTP header to our buffer (interrupts must be disabled!)
sendhdr:
        saveb   BUFINIT,FSR0
        savew   header,TBLPTRH,TBLPTRL
sendhdr1:
        memr    hibyte2 ; read word from program memory
        movwf   stemp ; save low byte
        ; process high byte of the word
        movfp   hibyte2,WREG
        cpfslt  fzero
        return ; return from subroutine if 0
        movwf   INDF0
        ; process low byte of the word
        movfp   stemp,WREG
        cpfslt  fzero
        return ; return from subroutine if 0
        movwf   INDF0
        goto    sendhdr1

; header must be <=112 characters and be multiple of 16
header: data "HTTP/1.1 200 OK\nContent-type: text/plain      \n\n",0
```

```
;             0123456789ABCDE F0123456789ABCDEF0123456789ABCD E F

; send HTTP reply with DATA (interrupts must be disabled!)
getdata:
        savew   4,temp3,temp4
         bsf     CPUSTA,GLINTD ; disable interrupts
        movff   curhi,dstahi
        movff   curlo,dstalo
         bcf     CPUSTA,GLINTD ; enable interrupts
getdata1:
        clrwdt
        movlw   HIGH(RAMBADR)
        cpfseq  dstahi
        goto    getdata2
        movlw   LOW(RAMBADR)
        cpfseq  dstalo
        goto    getdata2
        savew   ROMBADR,dstahi,dstalo
getdata2:
        subwff  dstahi,dstalo,temp3,temp4
        movfp   dstahi,TBLPTRH
        movfp   dstalo,TBLPTRL
        memr    hibyte2
        movwf   stemp
        tstfsz  hibyte2
        goto    getdata3
        goto    getdata5
getdata3:
        swapf   hibyte2,w
        call    conv_hex
        movwf   INDF0
        movfp   hibyte2,WREG
        call    conv_hex
        movwf   INDF0
        swapf   stemp,w
        call    conv_hex
        movwf   INDF0
        movfp   stemp,WREG
        call    conv_hex
        movwf   INDF0
        memr    hibyte2
        movwf   stemp
        swapf   hibyte2,w
```

```
        call    conv_hex
        movwf   INDF0
        movfp   hibyte2,WREG
        call    conv_hex
        movwf   INDF0
        swapf   stemp,w
        call    conv_hex
        movwf   INDF0
        movfp   stemp,WREG
        call    conv_hex
        movwf   INDF0
        memr    hibyte2
        movwf   stemp
        swapf   hibyte2,w
        call    conv_hex
        movwf   INDF0
        movfp   hibyte2,WREG
        call    conv_hex
        movwf   INDF0
        swapf   stemp,w
        call    conv_hex
        movwf   INDF0
        movfp   stemp,WREG
        call    conv_hex
        movwf   INDF0
        memr    hibyte2
        movwf   stemp
        swapf   hibyte2,w
        call    conv_hex
        movwf   INDF0
        movfp   hibyte2,WREG
        call    conv_hex
        movwf   INDF0
        movfp   stemp,WREG
        call    conv_hex
        movwf   INDF0
        movlw   '\n'
        movwf   INDF0
        tstfsz  FSR0
        goto    getdata4
        call    sendbuf
getdata4:
        decwf   gethi,getlo
```

```
        movfp   gethi,WREG
        iorwf   getlo,w
        btfss   ALUSTA,Z
        goto    getdata1
getdata5:
        tstfsz  FSR0
        call    sendbuf
        return

; send HTTP reply with OK (interrupts must be disabled!)
getok:  saveb   'O',INDF0
        saveb   'K',INDF0
        saveb   '\n',INDF0
        call    sendbuf
        return

; send character WREG to COM-port
com_send:
        clrwdt
        movlb   1
        btfss   PIR^0x100,TXIF
        goto    com_send
        movlb   0
        movwf   TXREG
        return

; read 1 hexadecimal digit using INDF0 and save to WREG
readh:  movlw   0x30
        subwf   INDF0,w
        movpf   WREG,itmp^0x100
        movlw   0x09
        cpfsgt  itmp^0x100
        goto    readh1
        movlw   0x07
        subwf   itmp^0x100,f
readh1: movfp   itmp^0x100,WREG
        return

; read 4 bytes using INDF0 and save 2-byte word (iwhi, iwlo)
readw:  call    readh
        movwf   iwhi^0x100
        swapf   iwhi^0x100,f
        call    readh
```

```
        addwf   iwhi^0x100,f
        call    readh
        movwf   iwlo^0x100
        swapf   iwlo^0x100,f
        call    readh
        addwf   iwlo^0x100,f
        return

; send half byte (WREG) in hexadecimal form
sendh:  andlw   0x0F
        addlw   0x30
        movwf   itmp^0x100
        movlw   0x39
        cpfsgt  itmp^0x100
        goto    sendh_
        movfp   itmp^0x100,WREG
        addlw   0x07
        goto    sendh__
sendh_:
        movfp   itmp^0x100,WREG
sendh__:
        call    com_send
        return

; send 2-byte word (iwhi, iwlo) in hexadecimal form
sendw:  swapf   iwhi^0x100,w
        call    sendh
        movfp   iwhi^0x100,WREG
        call    sendh
        swapf   iwlo^0x100,w
        call    sendh
        movfp   iwlo^0x100,WREG
        call    sendh
        return

; send new line
sendn:  callw   0x0D,com_send
        callw   0x0A,com_send
        return

; send ERR and new line
sende:  callw   'E',com_send
        callw   'R',com_send
```

```
        callw   'R',com_send
        callw   0x0D,com_send
        callw   0x0A,com_send
        return

; call far subroutine using iwhi and iwlo
fakecall:
        movfp   iwhi^0x100,PCLATH
        movfp   iwlo^0x100,PCL

; read 1 hexadecimal digit using INDF0 and save to WREG
readh0: movlw   0x30
        subwf   INDF0,w
        movpf   WREG,rtemp
        movlw   0x09
        cpfsgt  rtemp
        goto    readh2
        movlw   0x07
        subwf   rtemp,f
readh2: movfp   rtemp,WREG
        return

; read 4 hex digits using INDF0 and save 2-byte word (gethi, getlo)
readw0: call    readh0
        movwf   gethi
        swapf   gethi,f
        call    readh0
        addwf   gethi,f
        call    readh0
        movwf   getlo
        swapf   getlo,f
        call    readh0
        addwf   getlo,f
        return

; convert lower half byte of W to hexadecimal digit
conv_hex:
        andlw   0x0F
        addlw   0x30
        movwf   tmphex
        movlw   0x39
        cpfsgt  tmphex
        goto    conv_hex_
```

```
        movfp   tmphex,WREG
        addlw   0x07
        return
conv_hex_:
        movfp   tmphex,WREG
        return

; subprogram to delay 1 second
one_delay:
        movlw   100
        movwf   counta
del1s:  clrwdt
        ; 10ms for 20MHz
        delay2  64,countb,countc
        decfsz  counta,f
        goto    del1s
        return

; parse HTTP request and execute commands
qparse: saveb   BUFINIT,FSR0
        saveb   BUFSIZE,qtemp
        ; looking for path
qloop:  movfp   INDF0,WREG
        sublw   '/'
        btfsc   ALUSTA,Z
        goto    qfound1
        decfsz  qtemp,f
        goto    qloop
        return
qfound1: ; path found
        movfp   INDF0,WREG
        movwf   qtemp
        movlw   'c'
        cpfseq  qtemp
        goto    qfound2
        ; it's request to clear output
        movfp   INDF0,WREG
        movwf   qtemp
        movlw   '1'
        cpfseq  qtemp
        goto    qf1a
        ; clear 1st output (RA2)
        bsf     PORTA,2
```

```
        clrf    stat1,f
        goto    qfound0
qf1a:   movlw   '2'
        cpfseq  qtemp
        return ; do nothing if it's not 1/2
        ; clear 2nd output (RA3)
        bsf     PORTA,3
        clrf    stat2,f
        goto    qfound0
qfound2:
        movlw   's'
        cpfseq  qtemp
        goto    qfound3
        ; it's request to set output
        movfp   INDF0,WREG
        movwf   qtemp
        movlw   '1'
        cpfseq  qtemp
        goto    qf2a
        ; set 1st output (RA2)
        bcf     PORTA,2
        saveb   1,stat1
        goto    qfound0
qf2a:   movlw   '2'
        cpfseq  qtemp
        return ; do nothing if it's not 1/2
        ; set 2nd output (RA3)
        bcf     PORTA,3
        saveb   1,stat2
        goto    qfound0
qfound3:
        movlw   ' '
        cpfseq  qtemp
        goto    qfound4
        ; it's request for all data
        clrf    getlo,f
        movlw   0x08
        movwf   gethi
        call    sendhdr ; HTTP header
        call    getdata ; get 0x0800 records
        return ; return from qparse subroutine
qfound4: ; probably it's request for specified number of records
        decf    FSR0,f ; step back
```

```
        call    readw0 ; save number to gethi/getlo
        call    sendhdr ; HTTP header
        call    getdata ; get specified number of records
        return ; return from qparse subroutine
qfound0: ; send back "OK" mesage
        call    sendhdr ; HTTP header
        call    getok
        return ; return from qparse subroutine

main:
        clrf    ALUSTA,f      ; clear ALUSTA
        bsf     ALUSTA,FS3      ; no auto increment FSR1
        bcf     ALUSTA,FS1      ; auto increment FSR0
        bsf     ALUSTA,FS0
        bsf     CPUSTA,GLINTD   ; disable interrupts
        movlb   0             ; bank 0
        clrf    PORTA,f       ; clear port A
        bsf     PORTA,2       ; set "1" to RA2
        bsf     PORTA,3       ; set "1" to RA3
        bcf     PORTA,NOT_RBPU; enable weak pull ups for port B
        saveb   0xFF,DDRB     ; port B all inputs
        saveb   0x08,INTSTA   ; enable peripheral interrupts
        movlr   1             ; register bank 1
        saveb   BUFINIT,ibuf^0x100 ; setup buffer for RS-232
        movlr   0             ; register bank 0
        movlw   32            ; RS-232 : 9600 on 20 MHz
        movwf   SPBRG
        movlw   0x20
        movwf   TXSTA
        movlw   0x90
        movwf   RCSTA
        movlb   1
        movlw   0x01
        movwf   PIE^0x100
        movlb   0
        memw1d  0x02,0x300B ; RTC: disable SQWE, enable 24
        memw1d  0x30,0x300A ; RTC: enable extended regs, enable oscillator
        memw1d  0x00,0x304B ; RTC: disable E32K
        memw1d  0x20,0x300A ; RTC: disable extended regs, enable oscillator
        memr1n  0x3010 ; RTC: read NVM value for SQWE frequency
        tstfsz  WREG ; test for 0 that means network and external interrupts disabled
        goto    hinit1
        bcf     CPUSTA,GLINTD ; enable interrupts
```

```
        goto    mainloop
hinit1: andlw   0x0F ; use only lower half of byte
        iorlw   0x20 ; add 0x20 to enable oscillator
        memw1n  0x300A ; RTC: set SQWE frequency
        memw1d  0x0A,0x300B ; RTC: enable SQWE, enable 24
        saveb   0x80,T0STA ; external interrupt by raising edge (inverted /INT from WizNET)
        saveb   0x0D,INTSTA ; allow peripheral and 2 external interrupts (RA0 and RA1)
        movlr   0 ; 1st bank for file of registers
        clrf    iflag,f ; clear signal register
        clrf    fzero,f ; clear zero register
        clrf    isqwe,f ; clear sqwe counter
        bcf      CPUSTA,GLINTD ; enable interrupts
        savew   RAMBADR,curhi,curlo
        savew   RAMBADR,TBLPTRH,TBLPTRL
        savew   PAGSIZE,counth,countl
clram:  clrwdt
        clrf    hibyte2,w
        memw    hibyte2 ; clear external word
        decwf   counth,countl
        movfp   counth,WREG
        iorwf   countl,w
        btfss   ALUSTA,Z
        goto    clram

; initialize WizNET

        memw1d  0x00,MR
        memw1d  IR_SOCK0,IMR ; enable socket 0 interrupt
        memw1d  0x03,RMSR ; socket 0 - 8K
        memw1d  0x03,TMSR ; socket 0 - 8K

; copy values from NVM to WizNET registers

        saveb   18,ncount ; 18 bytes: 4-Gateway, 4-SubnetMask, 6-MAC, 4-IP
        savew   NVMINET,asrchi,asrclo
        savew   WIZINET,adsthi,adstlo
ncopy:  memr1f  asrchi,asrclo
        memw1f  adsthi,adstlo
        incwf   asrchi,asrclo
        incwf   adsthi,adstlo
        decfsz  ncount,f
        goto    ncopy
```

```
create: clrwdt ; clear watchdog timer
        clrf   nerr,f ; clear error number

; create socket

        memw1d Sn_MR_TCP|Sn_MR_ND,S0_MR
        memr1n NVMPHI ; high byte of default port from NVM
        memw1n S0_PORT0
        memr1n NVMPLO ; low byte of default port from NVM
        memw1n S0_PORT0+1
        memw1d Sn_CR_OPEN,S0_CR
        memtst S0_SR,SOCK_INIT,error1

; start listening

        memw1d Sn_CR_LISTEN,S0_CR
        memtst S0_SR,SOCK_LISTEN,error2

mainloop:
        delay10ms
        ; check for request
        tstfsz iflag
        goto   request
        call   checkcom
        goto   mainloop
request: ; we have request
        call   qparse
        clrf   iflag,f ; clear signal register
        memw1d Sn_CR_DISCON,S0_CR
        delay100ms
        memw1d Sn_CR_CLOSE,S0_CR
        saveb  SOCK_CLOSED,temp
closed: clrwdt
        ; check socket interrupts are presented
        memr1n S0_IR
        tstfsz WREG
        call   clearir
        ; check socket is closed
        memr1n S0_SR
        cpfseq temp
        goto   closed
        ; socket 0 closed and cleared
        memw1d IR_SOCK0,IMR ; enable socket 0 interrupt again
```

```
        goto   create ; go to create socket 0
clearir: ; clear S0_IR
        memw1n S0_IR
        return

error5: saveb  '5',nerr
        goto   error0
error4: saveb  '4',nerr
        goto   error0
error3: saveb  '3',nerr
        goto   error0
error2: saveb  '2',nerr
        goto   error0
error1: saveb  '1',nerr
error0: memw1d Sn_CR_CLOSE,S0_CR
        callw  'E',com_send
        callw  'R',com_send
        callw  'R',com_send
        movfp  nerr,WREG
        call   com_send
        callw  '\r',com_send
        callw  '\n',com_send
        memw1d IR_SOCK0,IMR ; enable socket 0 interrupt
        goto   create

checkcom:
        clrf   WREG,f
        movlr  1
        cpfseq ibuf^0x100
        goto   loop1
        ; command
        movlw  BUFINIT
        movwf  FSR0
        movpf  INDF0,itmp^0x100
        movlw  '?'
        cpfseq itmp^0x100
        goto   loop01
        ; command READ
        call   readw
        movfp  iwhi^0x100,WREG
        movwf  TBLPTRH
        movfp  iwlo^0x100,WREG
        movwf  TBLPTRL
```

```
        movpf   INDF0,itmp^0x100
        movlw   '='
        cpfseq  itmp^0x100
        goto    loop0r
        call    readw
        movfp   iwhi^0x100,WREG
        movwf   ichi^0x100
        movfp   iwlo^0x100,WREG
        movwf   iclo^0x100
        goto    loop0rr
loop0r: clrf    ichi^0x100,f
        movlw   0x01
        movwf   iclo^0x100
loop0rr:
        memr    iwhi^0x100
        movwf   iwlo^0x100
        call    sendw
        callw   ' ',com_send
        tstfsz  iclo^0x100
        goto    loop0rrr
        decf    ichi^0x100,f
loop0rrr:
        decf    iclo^0x100,f
        movfp   iclo^0x100,WREG
        iorwf   ichi^0x100,w
        btfss   ALUSTA,Z
        goto    loop0rr
        call    sendn
        goto    loop00
loop01: movlw   '!'
        cpfseq  itmp^0x100
        goto    loop02
        ; command WRITE
        call    readw
        movfp   iwhi^0x100,WREG
        movwf   TBLPTRH
        movfp   iwlo^0x100,WREG
        movwf   TBLPTRL
        movpf   INDF0,itmp^0x100
        movlw   '='
        cpfseq  itmp^0x100
        goto    loop0w
loop0www:
```

```
        call    readw
        movfp   iwhi^0x100,WREG
        movwf   ichi^0x100
        movfp   iwlo^0x100,WREG
        movwf   iclo^0x100
        goto    loop0ww
loop0w: decf    FSR0,f
        clrf    INDF0,f
        decf    FSR0,f
        clrf    ichi^0x100,f
        clrf    iclo^0x100,w
loop0ww:
        memw    ichi^0x100
        movpf   INDF0,WREG
        btfsc   ALUSTA,Z
        goto    loop00
        decf    FSR0,f
        goto    loop0www
loop02: movlw   '$'
        cpfseq  itmp^0x100
        goto    loop00
        ; command GOTO
        call    readw
        call    fakecall
        goto    loop00
loop0e: call    sende
loop00: movlr   0
        callw   '>',com_send
        movlr   1
        movlw   BUFINIT
        movwf   ibuf^0x100
        return
loop1:  movlw   BUFINIT
        movpf   WREG,FSR0
loop2:  movfp   ibuf^0x100,WREG
        cpfslt  FSR0
        return
        movpf   INDF0,WREG
        movlr   0
        movlr   1
        goto    loop2

        END
```

pixmacro.inc

```
; pixmacro.inc - useful macros for PIC17C4X
; ==========================================
; 10-Jan-2008 : dealy1,delay2
; 13-Jan-2008 : callw
; 19-Jan-2008 : memr,memrn,memrf,memw,memwn,memwf
; 20-Jan-2008 : memwd,fcall,fcallw
; 26-Jan-2008 : memr1,memr1n,memw1,memw1n,memw1d,fcallf
; 29-Jan-2008 : memr1f,memw1f,incwf,decwf,saveb,savew
; 30-Jan-2008 : addwfn,addwff,subwff,movff

; delay 3*N+3 cycles using one register (N=0 means 256)
; delay1 byte-value,reg-name
delay1  macro  N,R1
        local  D1
        movlw  N
        movwf  R1
D1:     decfsz R1,1
        goto   D1
        endm

; delay 774*N+3 cycles using two registers (N=0 means 256)
; delay2 byte-value,reg-name-1,reg-name-2
delay2  macro  N,R1,R2
        local  D2
        movlw  N
        movwf  R1
D2:     delay1 0,R2
        decfsz R1,1
        goto   D2
        endm

; set W and call subroutine
; callw byte-value,address
callw   macro  B,A ; 3
        movlw  B
        call   A
        endm

; call far subroutine using numeric address
; fcall word-address
fcall   macro A ; 4
```

```
        movlw   HIGH(A)
        movwf   PCLATH
        lcall   LOW(A)
        endm

; set W and call far subroutine
; fcallw byte-value,word-address
fcallw  macro   B,A ; 5
        movlw   HIGH(A)
        movwf   PCLATH
        movlw   B
        lcall   LOW(A)
        endm

; set W from R and call far subroutine
; fcallf reg-input,word-address
fcallf  macro   R,A ; 5
        movlw   HIGH(A)
        movwf   PCLATH
        movfp   R,WREG
        lcall   LOW(A)
        endm

; read memory using TBLPTRH and TBLPTRL as address
; save lower byte to W and higher byte to register
; memr reg-hibyte
memr    macro   R ; 4
        tablrd  0,1,WREG
        tlrd    1,R
        tlrd    0,WREG
        endm

; read memory using numeric address
; save lower byte to W and higher byte to register
; memrn reg-hibyte,word-address
memrn   macro   R,A ; 8
        movlw   HIGH(A)
        movwf   TBLPTRH
        movlw   LOW(A)
        movwf   TBLPTRL
        memr    R
        endm
```

```
; read memory using address saved in two registers
; save lower byte to W and higher byte to register
; memrf reg-hibyte,reg-hiadr,reg-loadr
memrf   macro  R,RH,RL ; 8
        movfp  RH,WREG
        movwf  TBLPTRH
        movfp  RL,WREG
        movwf  TBLPTRL
        memr   R
        endm

; write memory using TBLPTRH and TBLPTRL as address
; get lower byte from W and higher byte from register
; memw reg-hibyte
memw    macro  R ; 3
        tlwt   1,R
        tablwt 0,1,WREG
        endm

; write memory using numeric address
; get lower byte from W and higher byte from register
; memwn reg-hibyte,word-address
memwn   macro  R,A ; 7
        tlwt   0,WREG
        movlw  HIGH(A)
        movwf  TBLPTRH
        movlw  LOW(A)
        movwf  TBLPTRL
        tablwt 1,1,R
        endm

; write memory using address saved in two registers
; get lower byte from W and higher byte from register
; memwf reg-hibyte,reg-hiadr,reg-loadr
memwf   macro  R,RH,RL ; 7
        tlwt   0,WREG
        movfp  RH,WREG
        movwf  TBLPTRH
        movfp  RL,WREG
        movwf  TBLPTRL
        tablwt 1,1,R
        endm
```

```
; write memory by numeric word using numeric address
; memwd word-data,word-address
memwd   macro  D,A ; 9
        movlw  HIGH(A)
        movwf  TBLPTRH
        movlw  LOW(A)
        movwf  TBLPTRL
        movlw  HIGH(D)
        tlwt   1,WREG
        movlw  LOW(D)
        tablwt 0,1,WREG
        endm

; read 1 byte from memory to W using TBLPTRH and TBLPTRL as address
; memr1
memr1   macro ; 3
        tablrd 0,1,WREG
        tlrd   0,WREG
        endm

; read 1 byte from memory to W using address saved in two registers
; memr1f reg-hiadr,reg-loadr
memr1f  macro  RH,RL ; 8
        movfp  RH,WREG
        movwf  TBLPTRH
        movfp  RL,WREG
        movwf  TBLPTRL
        memr1
        endm

; read 1 byte from memory to W using numeric address
; memr1n word-address
memr1n  macro  A ; 7
        movlw  HIGH(A)
        movwf  TBLPTRH
        movlw  LOW(A)
        movwf  TBLPTRL
        memr1
        endm

; write W to memory using TBLPTRH and TBLPTRL as address
; memw1
memw1   macro ; 2
```

```
        tablwt  0,1,WREG
        endm

; write W to memory using numeric address
; memw1n word-address
memw1n  macro   A ; 7
        tlwt    0,WREG
        movlw   HIGH(A)
        movwf   TBLPTRH
        movlw   LOW(A)
        movwf   TBLPTRL
        tablwt  1,1,WREG
        endm

; write W to memory using address saved in two registers
; memw1f reg-hiadr,reg-loadr
memw1f  macro   RH,RL ; 7
        tlwt    0,WREG
        movfp   RH,WREG
        movwf   TBLPTRH
        movfp   RL,WREG
        movwf   TBLPTRL
        tablwt  1,1,WREG
        endm

; write 1 byte to memory using numeric address
; memw1d byte-data,word-address
memw1d  macro   B,A ; 8
        movlw   B
        memw1n  A
        endm

; increment word saved in 2 registers
; incwf reg-hibyte,reg-lobyte
incwf   macro   RH,RL ; 2
        infsnz  RL,f
        incf    RH,f
        endm

; decrement word saved in 2 registers
; decwf reg-hibyte,reg-lobyte
decwf   macro   RH,RL ; 4
        local   L1
```

```
        tstfsz  RL
        goto    L1
        decf    RH,f
L1:     decf    RL,f
        endm

; save byte to register
; saveb byte-data,reg-data
saveb   macro   B,R ; 2
        movlw   B
        movwf   R
        endm

; save word to 2 registers
; savew word-data,reg-hibyte,reg-lobyte
savew   macro   W,RH,RL ; 4
        movlw   HIGH(W)
        movwf   RH
        movlw   LOW(W)
        movwf   RL
        endm

; add constant to word saved in 2 registers
; addwn reg-hibyte,reg-lobyte,word-data ; reg=reg+word
addwfn  macro   RH,RL,W ; 4
        movlw   LOW(W)
        addwf   RL,f
        movlw   HIGH(W)
        addwfc  RH,f
        endm

; add word saved in 2 registers to another word saved in other 2 registers
; addwf reg-hibyte,reg-lobyte,reg2-hibyte,reg2-lobyte ; reg=reg+reg2
addwff  macro   RH,RL,R2H,R2L ; 6
        movfp   R2H,WREG
        addwf   RH,f
        movfp   R2L,WREG
        addwf   RL,f
        btfsc   ALUSTA,C
        incf    RH,f
        endm

; subtruct word saved in 2 registers from another word saved in other 2 registers
```

```
; subwf reg-hibyte,reg-lobyte,reg2-hibyte,reg2-lobyte ; reg=reg-reg2
subwff  macro   RH,RL,R2H,R2L ; 6
        movfp   R2H,WREG
        subwf   RH,f
        movfp   R2L,WREG
        subwf   RL,f
        btfss   ALUSTA,C
        decf    RH,f
        endm

; move value of one register to another
; movff reg-src,reg-dst
movff   macro   R1,R2 ; 2
        movfp   R1,WREG
        movwf   R2
        endm
```

wiznet.inc

```
; WizNET definitions
COMMON_BASE    equ 0x8000 ; out base address
IINCHIP_MAP_TXBUF equ COMMON_BASE + 0x4000
IINCHIP_MAP_RXBUF equ COMMON_BASE + 0x6000
MAX_SOCK_NUM   equ 4
SOCK0          equ 0
SOCK1          equ 1
SOCK2          equ 2
SOCK3          equ 3
MR             equ COMMON_BASE
; Gateway IP Register address
GAR0           equ (COMMON_BASE + 0x0001)
; Subnet mask Register address
SUBR0          equ (COMMON_BASE + 0x0005)
; Source MAC Register address
SHAR0          equ (COMMON_BASE + 0x0009)
; Source IP Register address
SIPR0          equ (COMMON_BASE + 0x000F)
; Interrupt Register
IR             equ (COMMON_BASE + 0x0015)
; Interrupt mask register
IMR            equ (COMMON_BASE + 0x0016)
; Timeout register address( 1 is 100us )
RTR0           equ (COMMON_BASE + 0x0017)
; Retry count reigster
RCR            equ (COMMON_BASE + 0x0019)
; Receive memory size reigster
RMSR           equ (COMMON_BASE + 0x001A)
; Transmit memory size reigster
TMSR           equ (COMMON_BASE + 0x001B)
; Unreachable IP register address in UDP mode
UIPR0          equ (COMMON_BASE + 0x002A)
; Unreachable Port register address in UDP mode
UPORT0         equ (COMMON_BASE + 0x002E)
; socket register
CH_BASE        equ (COMMON_BASE + 0x0400)
; size of each channel register map
CH_SIZE        equ 0x0100
; socket Mode registers
S0_MR          equ (CH_BASE + 0 * CH_SIZE + 0x0000)
S1_MR          equ (CH_BASE + 1 * CH_SIZE + 0x0000)
```

```
S2_MR          equ (CH_BASE + 2 * CH_SIZE + 0x0000)
S3_MR          equ (CH_BASE + 3 * CH_SIZE + 0x0000)
; channel Sn_CR registers
S0_CR          equ (CH_BASE + 0 * CH_SIZE + 0x0001)
S1_CR          equ (CH_BASE + 1 * CH_SIZE + 0x0001)
S2_CR          equ (CH_BASE + 2 * CH_SIZE + 0x0001)
S3_CR          equ (CH_BASE + 3 * CH_SIZE + 0x0001)
; channel interrupt registers
S0_IR          equ (CH_BASE + 0 * CH_SIZE + 0x0002)
S1_IR          equ (CH_BASE + 1 * CH_SIZE + 0x0002)
S2_IR          equ (CH_BASE + 2 * CH_SIZE + 0x0002)
S3_IR          equ (CH_BASE + 3 * CH_SIZE + 0x0002)
; channel status registers
S0_SR          equ (CH_BASE + 0 * CH_SIZE + 0x0003)
S1_SR          equ (CH_BASE + 1 * CH_SIZE + 0x0003)
S2_SR          equ (CH_BASE + 2 * CH_SIZE + 0x0003)
S3_SR          equ (CH_BASE + 3 * CH_SIZE + 0x0003)
; source port registers
S0_PORT0       equ (CH_BASE + 0 * CH_SIZE + 0x0004)
S1_PORT0       equ (CH_BASE + 1 * CH_SIZE + 0x0004)
S2_PORT0       equ (CH_BASE + 2 * CH_SIZE + 0x0004)
S3_PORT0       equ (CH_BASE + 3 * CH_SIZE + 0x0004)
; Peer MAC registers
S0_DHAR0       equ (CH_BASE + 0 * CH_SIZE + 0x0006)
S1_DHAR0       equ (CH_BASE + 1 * CH_SIZE + 0x0006)
S2_DHAR0       equ (CH_BASE + 2 * CH_SIZE + 0x0006)
S3_DHAR0       equ (CH_BASE + 3 * CH_SIZE + 0x0006)
; Peer IP registers
S0_DIPR0       equ (CH_BASE + 0 * CH_SIZE + 0x000C)
S1_DIPR0       equ (CH_BASE + 1 * CH_SIZE + 0x000C)
S2_DIPR0       equ (CH_BASE + 2 * CH_SIZE + 0x000C)
S3_DIPR0       equ (CH_BASE + 3 * CH_SIZE + 0x000C)
; Peer port registers
S0_DPORT0      equ (CH_BASE + 0 * CH_SIZE + 0x0010)
S1_DPORT0      equ (CH_BASE + 1 * CH_SIZE + 0x0010)
S2_DPORT0      equ (CH_BASE + 2 * CH_SIZE + 0x0010)
S3_DPORT0      equ (CH_BASE + 3 * CH_SIZE + 0x0010)
; Maximum Segment Size(Sn_MSSR0) registers
S0_MSSR0       equ (CH_BASE + 0 * CH_SIZE + 0x0012)
S1_MSSR1       equ (CH_BASE + 1 * CH_SIZE + 0x0012)
S2_MSSR2       equ (CH_BASE + 2 * CH_SIZE + 0x0012)
S3_MSSR3       equ (CH_BASE + 3 * CH_SIZE + 0x0012)
; Protocol of IP Header field registers in IP raw mode
```

```
S0_PROTO       equ (CH_BASE + 0 * CH_SIZE + 0x0014)
S1_PROTO       equ (CH_BASE + 1 * CH_SIZE + 0x0014)
S2_PROTO       equ (CH_BASE + 2 * CH_SIZE + 0x0014)
S3_PROTO       equ (CH_BASE + 3 * CH_SIZE + 0x0014)
; IP Type of Service(TOS) Registers
S0_TOS         equ (CH_BASE + 0 * CH_SIZE + 0x0015)
S1_TOS         equ (CH_BASE + 1 * CH_SIZE + 0x0015)
S2_TOS         equ (CH_BASE + 2 * CH_SIZE + 0x0015)
S3_TOS         equ (CH_BASE + 3 * CH_SIZE + 0x0015)
; IP Time to live(TTL) Registers
S0_TTL         equ (CH_BASE + 0 * CH_SIZE + 0x0016)
S1_TTL         equ (CH_BASE + 1 * CH_SIZE + 0x0016)
S2_TTL         equ (CH_BASE + 2 * CH_SIZE + 0x0016)
S3_TTL         equ (CH_BASE + 3 * CH_SIZE + 0x0016)
; Transmit free memory size registers
S0_TX_FSR0     equ (CH_BASE + 0 * CH_SIZE + 0x0020)
S1_TX_FSR0     equ (CH_BASE + 1 * CH_SIZE + 0x0020)
S2_TX_FSR0     equ (CH_BASE + 2 * CH_SIZE + 0x0020)
S3_TX_FSR0     equ (CH_BASE + 3 * CH_SIZE + 0x0020)
; Transmit memory read pointer registers
S0_TX_RD0      equ (CH_BASE + 0 * CH_SIZE + 0x0022)
S1_TX_RD0      equ (CH_BASE + 1 * CH_SIZE + 0x0022)
S2_TX_RD0      equ (CH_BASE + 2 * CH_SIZE + 0x0022)
S3_TX_RD0      equ (CH_BASE + 3 * CH_SIZE + 0x0022)
; Transmit memory write pointer registers
S0_TX_WR0      equ (CH_BASE + 0 * CH_SIZE + 0x0024)
S1_TX_WR0      equ (CH_BASE + 1 * CH_SIZE + 0x0024)
S2_TX_WR0      equ (CH_BASE + 2 * CH_SIZE + 0x0024)
S3_TX_WR0      equ (CH_BASE + 3 * CH_SIZE + 0x0024)
; Received data size registers
S0_RX_RSR0     equ (CH_BASE + 0 * CH_SIZE + 0x0026)
S1_RX_RSR0     equ (CH_BASE + 1 * CH_SIZE + 0x0026)
S2_RX_RSR0     equ (CH_BASE + 2 * CH_SIZE + 0x0026)
S3_RX_RSR0     equ (CH_BASE + 3 * CH_SIZE + 0x0026)
; Read point of Receive memory
S0_RX_RD0      equ (CH_BASE + 0 * CH_SIZE + 0x0028)
S1_RX_RD0      equ (CH_BASE + 1 * CH_SIZE + 0x0028)
S2_RX_RD0      equ (CH_BASE + 2 * CH_SIZE + 0x0028)
S3_RX_RD0      equ (CH_BASE + 3 * CH_SIZE + 0x0028)
; Write point of Receive memory
S0_RX_WR0      equ (CH_BASE + 0 * CH_SIZE + 0x002A)
S1_RX_WR0      equ (CH_BASE + 1 * CH_SIZE + 0x002A)
S2_RX_WR0      equ (CH_BASE + 2 * CH_SIZE + 0x002A)
```

```
S3_RX_WR0      equ (CH_BASE + 3 * CH_SIZE + 0x002A)

; MODE register values
MR_RST         equ 0x80 ; reset
MR_PB          equ 0x10 ; ping block
MR_PPPOE       equ 0x08 ; enable pppoe
MR_LB          equ 0x04 ; little or big endian selector in indirect mode
MR_AI          equ 0x02 ; auto-increment in indirect mode
MR_IND         equ 0x01 ; enable indirect mode

; IR register values
IR_CONFLICT    equ 0x80 ; check ip confict
IR_UNREACH     equ 0x40 ; get the destination unreachable message in UDP sending
IR_PPPoE       equ 0x20 ; get the PPPoE close message
IR_SOCK0       equ 0x01 ; check socket 0 interrupt
IR_SOCK1       equ 0x02 ; check socket 1 interrupt
IR_SOCK2       equ 0x04 ; check socket 2 interrupt
IR_SOCK3       equ 0x08 ; check socket 3 interrupt

; Sn_MR values
Sn_MR_CLOSE    equ 0x00 ; unused socket
Sn_MR_TCP      equ 0x01 ; TCP
Sn_MR_UDP      equ 0x02 ; UDP
Sn_MR_IPRAW    equ 0x03 ; IP LAYER RAW SOCK
Sn_MR_MACRAW   equ 0x04 ; MAC LAYER RAW SOCK
Sn_MR_PPPOE    equ 0x05 ; PPPoE
Sn_MR_ND       equ 0x20 ; No Delayed Ack(TCP) flag
Sn_MR_MULTI    equ 0x80 ; support multicating

; Sn_CR values
Sn_CR_OPEN     equ 0x01 ; initialize or open socket
Sn_CR_LISTEN   equ 0x02 ; wait connection request in tcp mode(Server mode)
Sn_CR_CONNECT  equ 0x04 ; send connection request in tcp mode(Client mode)
Sn_CR_DISCON   equ 0x08 ; send closing reqeuset in tcp mode
Sn_CR_CLOSE    equ 0x10 ; close socket
Sn_CR_SEND     equ 0x20 ; updata txbuf pointer, send data
Sn_CR_SEND_MAC equ 0x21 ; send data with MAC address, so without ARP process
Sn_CR_SEND_KEEP       equ 0x22 ; send keep alive message
Sn_CR_RECV     equ 0x40 ; update rxbuf pointer, recv data

; Sn_IR values
Sn_IR_SEND_OK  equ 0x10 ; complete sending
Sn_IR_TIMEOUT  equ 0x08 ; assert timeout
```

```
Sn_IR_RECV     equ 0x04 ; receiving data
Sn_IR_DISCON   equ 0x02 ; closed socket
Sn_IR_CON      equ 0x01 ; established connection

; Sn_SR values
SOCK_CLOSED    equ 0x00 ; closed
SOCK_INIT      equ 0x13 ; init state
SOCK_LISTEN    equ 0x14 ; listen state
SOCK_SYNSENT   equ 0x15 ; connection state
SOCK_SYNRECV   equ 0x16 ; connection state
SOCK_ESTABLISHED equ 0x17 ; success to connect
SOCK_FIN_WAIT  equ 0x18 ; closing state
SOCK_CLOSING   equ 0x1A ; closing state
SOCK_TIME_WAIT equ 0x1B ; closing state
SOCK_CLOSE_WAIT       equ 0x1C ; closing state
SOCK_LAST_ACK  equ 0x1D ; closing state
SOCK_UDP       equ 0x22 ; udp socket
SOCK_IPRAW     equ 0x32 ; ip raw mode socket
SOCK_MACRAW    equ 0x42 ; mac raw mode socket
SOCK_PPPOE     equ 0x5F ; pppoe socket

; IP PROTOCOL
IPPROTO_IP     equ 0    ; Dummy for IP
IPPROTO_ICMP    equ 1    ; Control message protocol
IPPROTO_IGMP    equ 2    ; Internet group management protocol
IPPROTO_GGP     equ 3    ; Gateway^2 (deprecated)
IPPROTO_TCP     equ 6    ; TCP
IPPROTO_PUP     equ 12   ; PUP
IPPROTO_UDP     equ 17   ; UDP
IPPROTO_IDP     equ 22   ; XNS idp
IPPROTO_ND      equ 77   ; UNOFFICIAL net disk protocol
IPPROTO_RAW     equ 255  ; Raw IP packet
```

npc17v10.jed

```
NPC17V10.JED


Address decoder for NPC-17 v1.0
December 2007


PROJECT NUMBER 001098


GAL16V8*QP20*QF2194*G0*F0*
NOTE I-PINS: 1:_WR 2:A15 3:A14 4:A13 5:A12 6:A11 7:A10 8:A9 9:A8 11:_RD *
NOTE O-PINS: 12:WR_REG3 13:_CS_WIN5 14:_CS_BUS2 15:_CS_BUS1 16:_CS_RTC 17:_CS_NET 18:_CS_ROM 19:_CS_RAM *
NOTE-A15,W-A14--A13--A12--A11--A10--A9---A8,R*
NOTE--0000-0000-0000-0000-0000-0000-0000-0000*
NOTE PIN 19: _CS_RAM = ! (!A15 A14 !A13) *
L0000 1011 0111 1011 1111 1111 1111 1111 1111*
NOTE PIN 18: _CS_ROM = ! (!A15 !A14 !A13 + !A15 A14 A13) *
L0256 1011 1011 1011 1111 1111 1111 1111 1111*
L0288 1011 0111 0111 1111 1111 1111 1111 1111*
NOTE PIN 17: _CS_NET = ! (A15 !A14 !A13 !_WR + A15 !A14 !A13 !_RD + A15 A14 !_WR + A15 A14 !_RD) *
L0512 0110 1011 1011 1111 1111 1111 1111 1111*
L0544 0111 1011 1011 1111 1111 1111 1111 1110*
L0576 0110 0111 1111 1111 1111 1111 1111 1111*
L0608 0111 0111 1111 1111 1111 1111 1111 1110*
NOTE PIN 16: _CS_RTC = ! (!A15 !A14 A13 A12 !A11 !A10 !A9 !A8) *
L0768 1011 1011 0111 0111 1011 1011 1011 1011*
NOTE PIN 15: _CS_BUS1 = ! (!A15 !A14 A13 A12 A11 !A10 !A9) *
L1024 1011 1011 0111 0111 0111 1011 1011 1111*
NOTE PIN 14: _CS_BUS2 = ! (!A15 !A14 A13 A12 A11 !A10 A9) *
L1280 1011 1011 0111 0111 0111 1011 0111 1111*
NOTE PIN 13: _CS_WIN5 = ! (A15 !A14 A13) *
L1536 0111 1011 0111 1111 1111 1111 1111 1111*
NOTE PIN 12: WR_REG3 = (!A15 !A14 A13 A12 !A11 A10 A9 A8 !_WR) *
L1792 1010 1011 0111 0111 1011 0111 0111 0111*
NOTE XOR CONFIGURATION: *
L2048 00000001*
NOTE SIGNATURE: *
L2056 01001110 01010000 01000011 00110001*
L2088 00110111 01110110 00110001 00110000*
NOTE AC1 CONFIGURATION: *
L2120 00000000*
NOTE PRODUCT TERM DISABLE: *
L2128 10000000 11000000 11110000 10000000 10000000 10000000 10000000 10000000*
NOTE GLOBAL CINFIGURATION (SYN,AC0): *
L2192 10*
C30C0*
0000
```