Using the RealTek without a Serial EEPROM

The RealTek 8019AS Ethernet MAC is a extremely popular in embedded 8-bit devices because it is both inexpensive and has a resonable number of features. The device shares a software interface with the National Semiconductor 8390 which formed the basis for the Novell NE2000 driver specification, which was originally intended for PC ISA bus cards. Though lacking in comprehensive official documentation, there is a large amount of public information on this chip.

Comprehensive description of RealTek features here...

The RealTek chip is designed to read its initial configuration information off of a serial EEPROM which holds the RealTek CONFIG1, CONFIG2, CONFIG3, CONFIG4, the Ethernet MAC address, and a product ID. Most of the information in these registers can either be set with jumper configuration or through the parallel inteface.

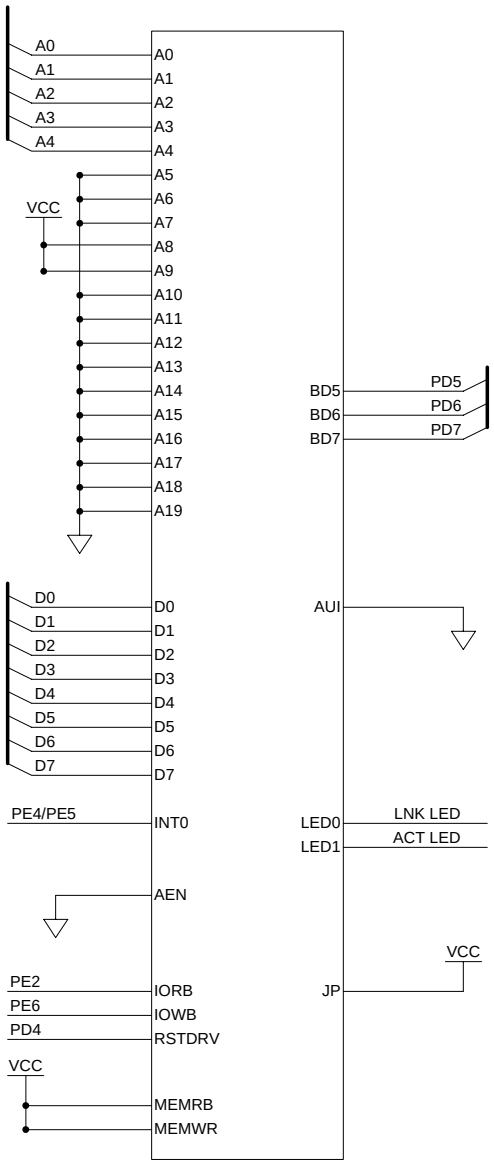
Comprehensive description of 93C46 features here...

The 93C46 has a simple interface with a clock line SK, an address line DI, and a data line DO. When the RealTek wants to read a byte from the 93C46, it starts SK clocking and clocks out an address on DI. It expects the result to be clocked out on DO.

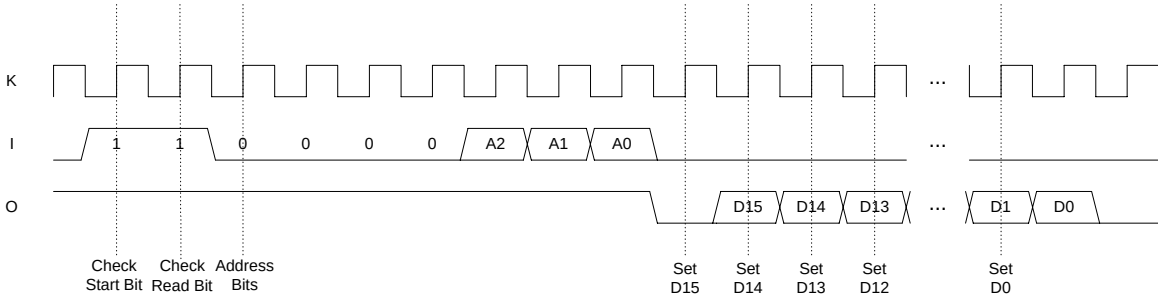
We wanted to reduce the component cost in our Ethernet designs and saw that the 93C46 is a place where we could save some money. We initially removed it from the board and after some discussion with the RealTek technical support discovered that three bits in a configuration registers can only be configured from the 9346. These bits control full duplex mode and the LED modes. We wanted to be able to support full duplex mode and wanted a different setting for the LED settings than the default. We initially tried tying DO high which would in effect set the LEDs to the right value and the full duplex mode on, but discovered that the description of one bit in the RealTek configuration register should never have a one written to it. We decided to replace the 93C46 with three I/O lines from the Rabbit.

The following diagram shows how the RealTek is wired in an upcoming product. It does not include the connection to hook up the Ethernet or the crystal. We have the RealTek wired in jumper mode. Jumper mode causes the RealTek to setup things like the I/O port, interrupt lines based on external jumpers.

The main connections to the RealTek are the address lines, data lines, RSTDRV, INT0, BD5, BD6, and BD7. RSTDRV starts the chip when it is not asserted. The INT0 line is hooked to the PE4 and PE5 Rabbit interrupt pair. BD5, BD6, and BD7 would normally be connected to the 9346: BD5 is the clock line, BD6 is the address line, and BD7 is the data line.



The following diagram is based on a diagram from the RealTek 8019 documentation. It is a timing diagram for the RealTek reading the serial EEPROM. We hooked SK and DI to inputs on the Rabbit and DO to an output. After we initialize the parallel interface to the RealTek we start the RealTek and allow it to read all zeros. When the RealTek starts responding again, we tell it to reread the serial EEPROM by writing 0x40 to the 9346CR register on the RealTek.



We then start looking at the SK line sampling DI on each low to high transition of SK. The time from rising edge to rising edge on SK is about 6.4μs. When we find DI high on a rising edge, we check DI at the next low to high transition. If that is high, we know that we have read the start bit and the read operation bit. At this point we are going to play a little bit of a trick. Since we only care about the CONFIG1 register and all of the other registers on the RealTek can be set with the parallel interface, any response to a byte request will be either 0x70 for full duplex or 0x30 when we do not want full duplex. We count the eight clocks

The following code has some assumptions. The first is that you select SK and DI on the same register. The second is that you select DO on a bit addressable register. Since the timing is somewhat tight I would suggest you turn interrupts off when you call this routine.

If you select a I/O pin that is not pin seven you will need to rotate the RT_FDLED0LED1 so the most significant bit will go out first...

```
;
; Z-World, Copyright (c) 2001
;
#define RT_FDLED0LED1  0x70
#define PD_WR1         0xC000
#define EESK           5
#define EEDI           6
#define EEDO           PDB7R
#define EEPORT         PDDR

nodebug
void init_realtek()
{
#asm
;
;    Setup the I/O configuration and strobes
;

         ld       a,(PEDDRShadow)   ; PE2,PE6 io strobes
         or       a,0x44
```

```
        ld      (PEDDRShadow),a
ioi     ld      (PEDDR),a

        ld      a,(PEFRShadow)    ; PE2,PE6 io strobes
        or      a,0x44
        ld      (PEFRShadow),a
ioi     ld      (PEFR),a

        ld      a,0x90            ; set the io strobes
        ld      (IB2CRShadow),a
ioi     ld      (IB2CR),a
        ld      a,0xa8
        ld      (IB6CRShadow),a
ioi     ld      (IB6CR),a

        ld      a,0x90            ; PD7,PD4 outputs
ioi     ld      (PDDDR),a

        ld      a,0x00            ; no open drains, no special functions
ioi     ld      (PDDCR),a
ioi     ld      (PDFR),a

        ld      a,0x10
ioi     ld      (PDDR),a

        ld      hl,1000           ; wait for NIC to reset and establish link...
        call    mswait

        ld      a,0x00            ; PD7=OFF,PD4=OFF
ioi     ld      (PDDR),a

        ld      hl,1000

        call    mswait

;
;   Issue instruction for the Realtek to read 9346
;

        ld      a,0xc0
ioe     ld      (PD_WR1),a

        ld      a,0x40
ioe     ld      (PD_WR1+PD_9346CR),a

        call    emu_9346

        ipres
#endasm
}
```

```
#asm nodebug
clock_9346::
h2l:
ioi     bit     EESK,(hl)
        jr      z,h2l
l2h:
ioi     bit     EESK,(hl)
        jr      nz,l2h
        ret
#endasm

nodebug
void emu_9346()
{
#asm
;
; code to emulate the 93C46
;
        ld      c,1
        ld      hl,EEPORT

        ;
        ; start bit
        ;
emu_1:
        call    clock_9346
ioi     bit     EEDI,(hl)
        jr      z,emu_1

        ;
        ; read 10 - read command
        ;

emu_2:
        call    clock_9346
ioi     bit     EEDI,(hl)
        jr      z,emu_1

emu_3:
        call    clock_9346
ioi     bit     EEDI,(hl)
        jr      nz,emu_1

        ;
        ; six clocks for address
        ;

        ld      b,6
emu_4:
```

```
        call    clock_9346
        djnz    emu_4

        ;
        ; send 0
        ;

        xor     a
ioi     ld      (EEDO),a

        ;
        ; send word
        ;

        ld      a,RT_FDLED0LED1
        ld      b,16

emu_5:
        call    clock_9346
ioi     ld      (EEDO),a
        rlca
        djnz    emu_5

        dec     c
        jp      p,emu_1

        xor     a
ioi     ld      (EEDO),a

#endasm
}
```

Getting a range of Ethernet addresses.

References

8019 Data Sheet
8019AS Data Sheet
National 8390 Data Sheets.
Atmel 93C46 Data Sheet.