Via Santa Maria Maddalena 12, 38100 Trento, Italy
tel. +39-0461-260 552 - fax + 39-0461-260 617
e-mail: info@neuricam.com; http: www.neuricam.com

NC1001 256 x 256 Digital Camera

APPLICATION NOTE AN001

Rel. 11/98

Introduction

The NC1001 digital optical sensor is intended to be used in conjunction with common microprocessors or microcontrollers. For a detailed description refer to the datasheet of the component.

Shown below are some examples of the application of the sensor in systems.

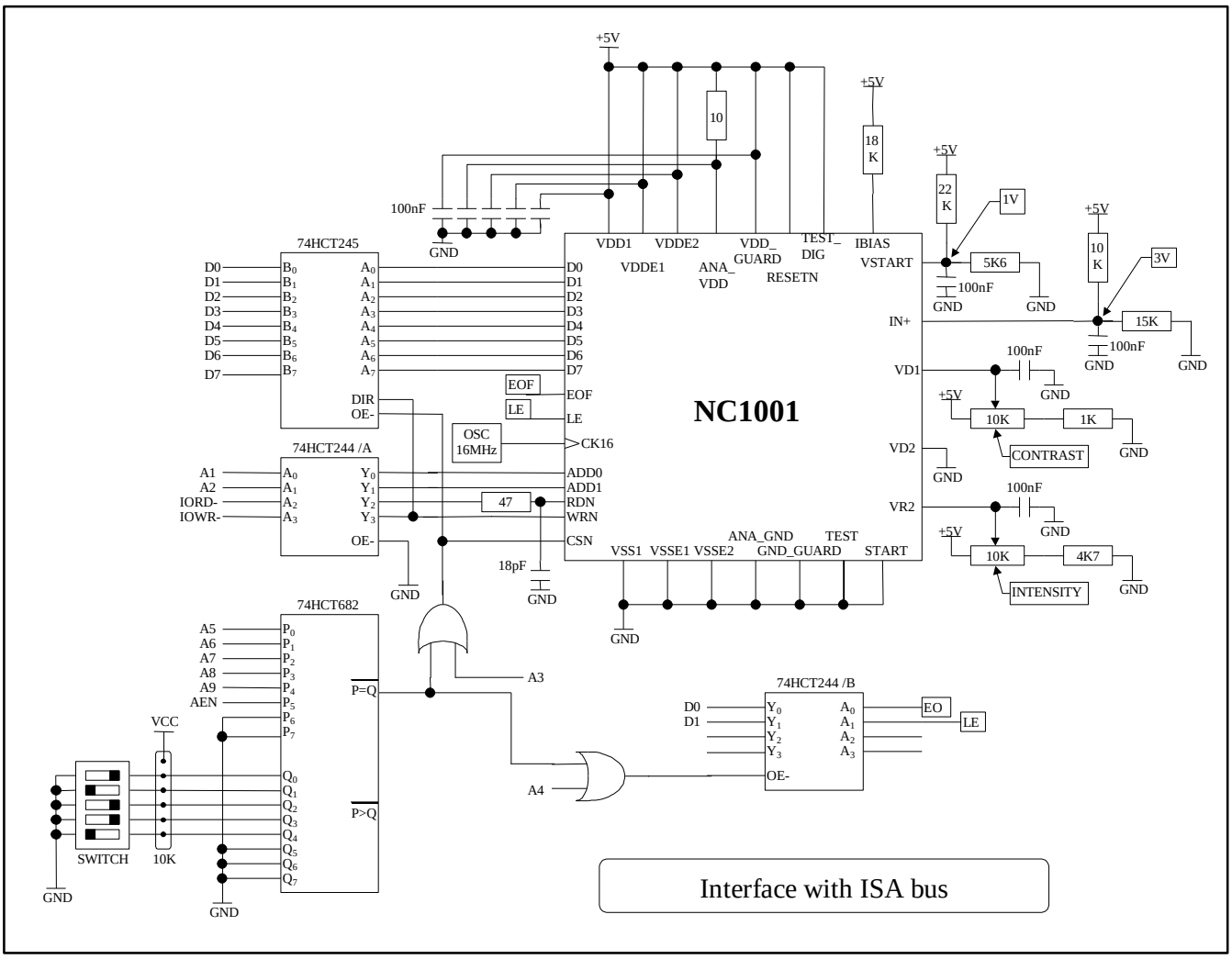
Digital interface:
The minimum configuration of the sensor does not require any digital components for interfacing with microprocessor buses. In many cases, however, depending on the nature of the application, clock generation, external buffering or address decoding may be necessary.

Power supply:
Note that as with all mixed analogue-digital components, the quality of the power supply is key to the achievement of high-quality images. The use of a regulated 5V power supply is advised. Proper grounding techniques should be used, such as star connection to a common ground point, filtering of the analogue supplies and separation between the analogue and the noisy digital grounds. Four-layer PCBs are strongly advised, even if good results can be obtained with well-designed two-layer boards.

Analogue references:
Simple resistor chains can be used to derive the desired analogue reference voltages and bias current from the supply voltage.

Example 1: camera with ISA bus interface

This example shows a direct connection with an ISA bus which employs TTL components.



Procedure to start camera and read one frame

1. Software reset
2. Program Line_Time Register and Frame_Time Register
3. Software start
4. Wait for falling edge of EOF
5. For j:=1 to 256
    6. Wait for rising edge of LE
        7. For i:=1 to 256
            8. Read Data_Register
        9. Next i
10. Next j

Register Addressing

		A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
Base Address	BADDR	A9	A8	A7	A6	A5	0	0	0	0	0
Control Register	CTRL_REG						1	0	0	0	0
Line Time Register	LTIME_REG						1	0	0	1	0
Frame Time Register	FTIME_REG						1	0	0	1	0
Data Register	DATA_REG						1	0	1	1	0
Input Address	INP_ADDR						0	1	0	0	0

Software Reset

```
outportb(BADDR+CRTL_REG,0x01); //Software Reset
delay(100);
outpotb(BADDR+CTRL_REG,0x00); //Release Software Reset
delay(100);
```

Program Line_Time Register

```
outportb(BADDR+LTIME_REG,value);
```

Program Frame_Time Register

```
outportb(BADDR+FTIME_REG,value);
```

Software Start

```
outportb(BADDR+CTRL_REG,0x02);
```

Wait for falling edge of EOF

```
While( (inportb(BADDR+INP_ADDR)  & 0x01   ) == 0);
While( (inportb(BADDR+INP_ADDR)  & 0x01   ) !=  0);
```

Wait for rising edge of LE

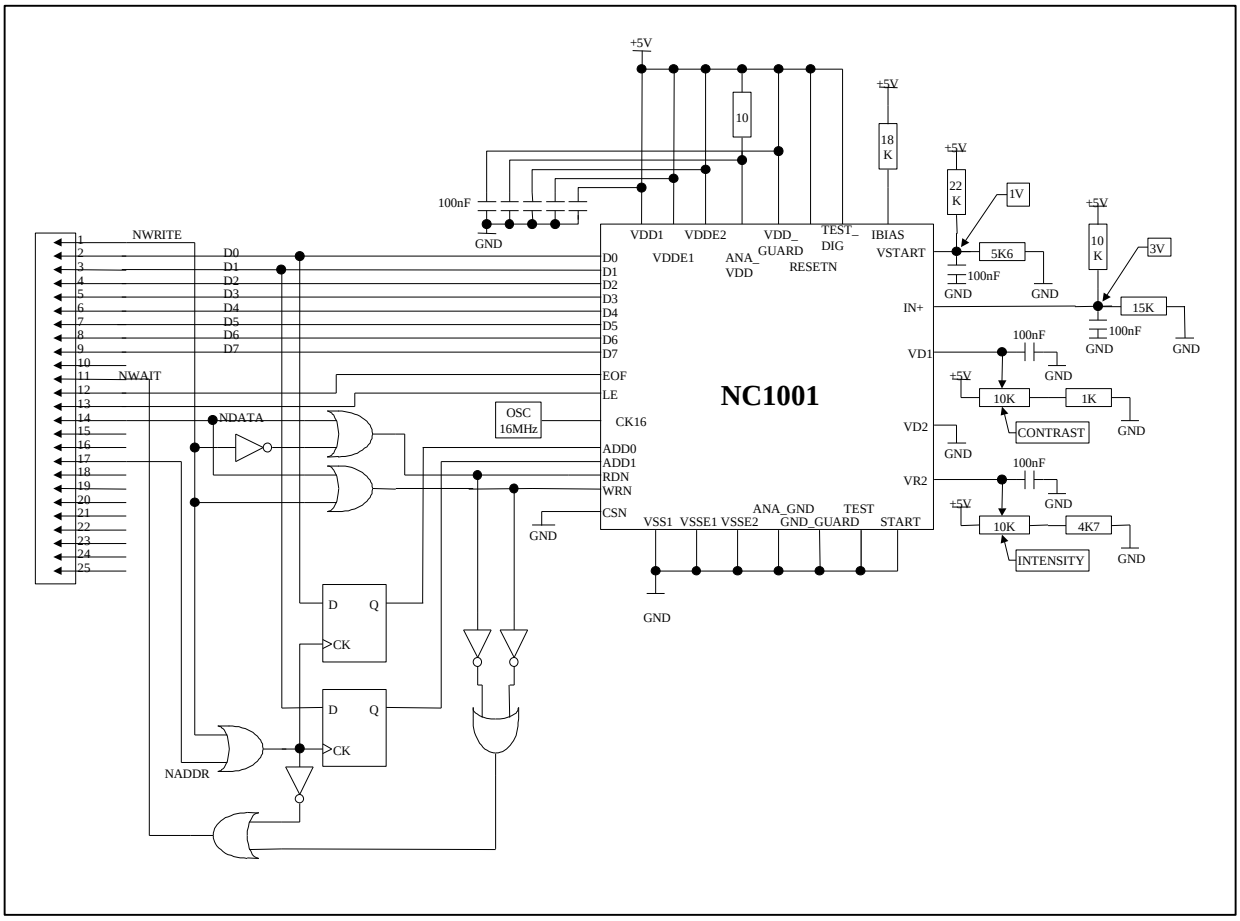
```
While( (inportb(BADDR+INP_ADDR)  & 0x02   ) != 0);
While( (inportb(BADDR+INP_ADDR)  & 0x02   ) ==  0);
```

Read Data Register

```
unsigned char pixel = inportb(BADDR+DATA_REG);
```

Example 2: Camera with EPP interface

The example is similar to the above with synchronisation hardware with a bidirectional parallel EPP interface.



Procedure to start camera and read a frame

1. Place parallel port in EPP mode
2. Software Reset camera
3. Program Line Time Register and Frame Time Register
4. Software Start Camera
5. Read Enable
6. Wait for falling edge of EOF
7. For j:=1 to 256
    8. Wait for rising edge of LE
    9. For i:=1 to 256
        10. Read Data Register
    11. Next i
12. Next j

Parallel Port Registers

```
#define BASE    0x378;          //Parallel BASE Address
#define ECR     BASE+0x402;     //Extended Control Register
#define DCR     BASE+0x02;      //Control Register
#define DSR     BASE+0x01;      //Status Register
#define EPPAddr BASE+0x03;      //Read and write Address
#define EPPData BASE+0x04;      //Read and write Data
```

Camera Registers

```
#define CTRL_REG   0x00;      //Control Register
#define TLINE_REG  0x01;      //Line Time Register
#define TFRAME_REG 0x02;      //Frame Time Register
#define DATA_REG   0x03;      //Data Register
```

Placing Parallel Port in EPP mode

```
void Set_EPP( void )
{
      int statECR;
      int mode=0x04;
      statECR=inportb(ECR);
      statECR=(statECR & 0x1F) + (mode * 0x20);
      outport(ECR,statECR);
}
```

Write Line Time Register

```
void set_Tline(int val)
{
      outportb(EPPAddr,WAIT_REG);
      delay(1);
      outportb(EPPData, val);
}
```

Write Frame Time Register

```
void set_Tframe(int val)
{
      outportb(EPPAddr,SCAL_REG);
      delay(1);
      outportb(EPPData, val);
}
```

Software Reset Camera

```
void reset()
{
       outportb(EPPAddr,CTRL_REG);
       outportb(EPPData,1);   //Reset
       delay(1);
       outportb(EPPData,0);   //Release Reset
}
```

Software Start Camera

```
void start()
{
      outportb(EPPAddr,CTRL_REG);
```

```
      delay(1);
      outportb(EPPData,2);
}
```

Wait End Of Frame Camera

```
void wait_frame()
{
      while(  !(inportb(DSR) & 0x20) );
      while( (inportb(DSR) & 0x20));
}
```

Wait Line End Camera

```
void wait_line()
{
      while(  inportb(DSR) & 0x10 );
      while(!(inportb(DSR) & 0x10));
}
```

Read Enable

```
void Read_EN()
{
      outportb(EPPAddr,DATA_REG);
}
```

Read Data Register

```
unsigned char Read_Data()
{
      return inportb(EPPData);
}
```

Example 3: Camera with frame buffer and EPP interface

A byte-wide static memory is used to store a frame, which ensures that complete frames are transferred even on low-speed EPP interfaces. Control of the memory and interface is effected with an FPGA.