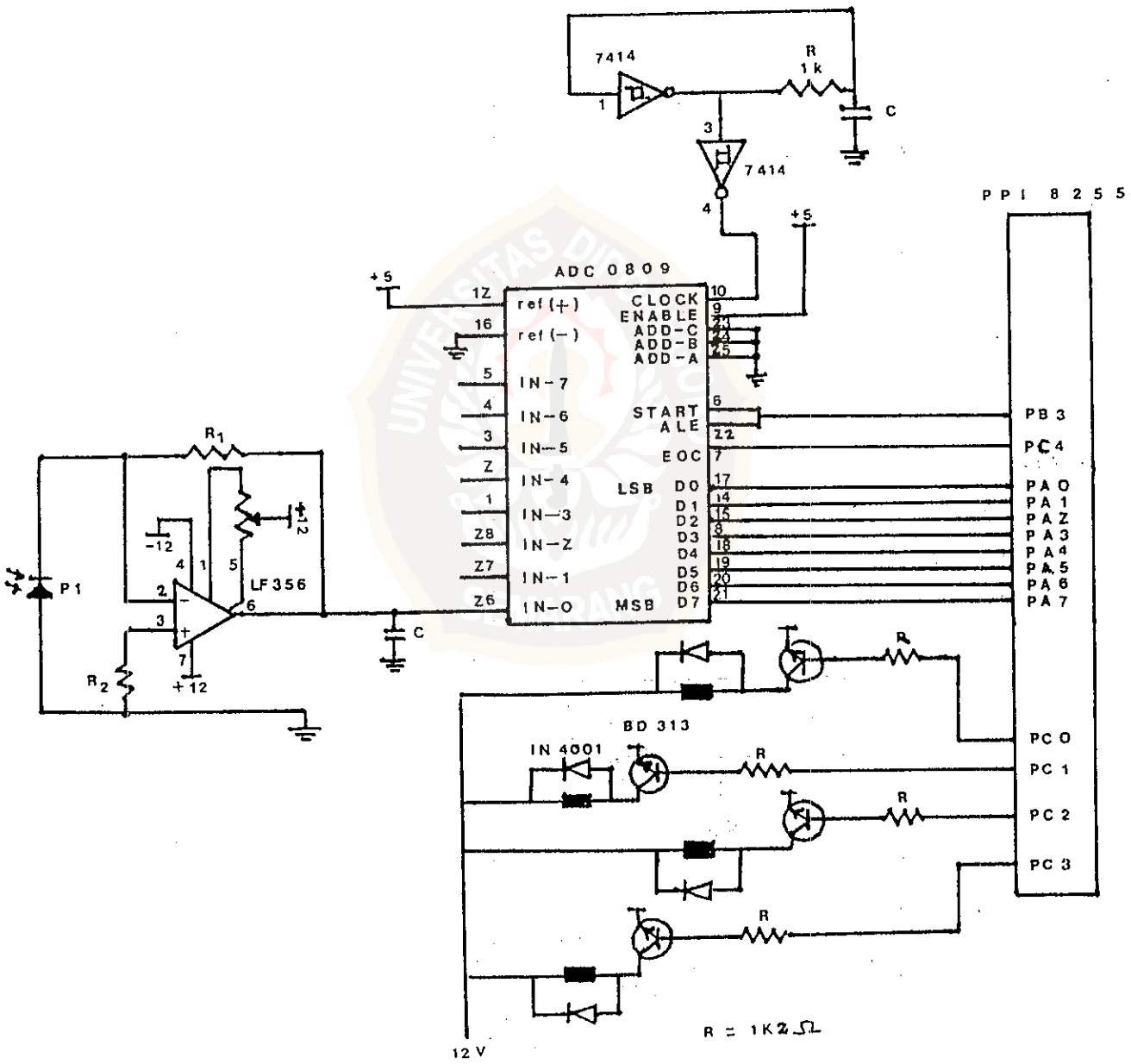
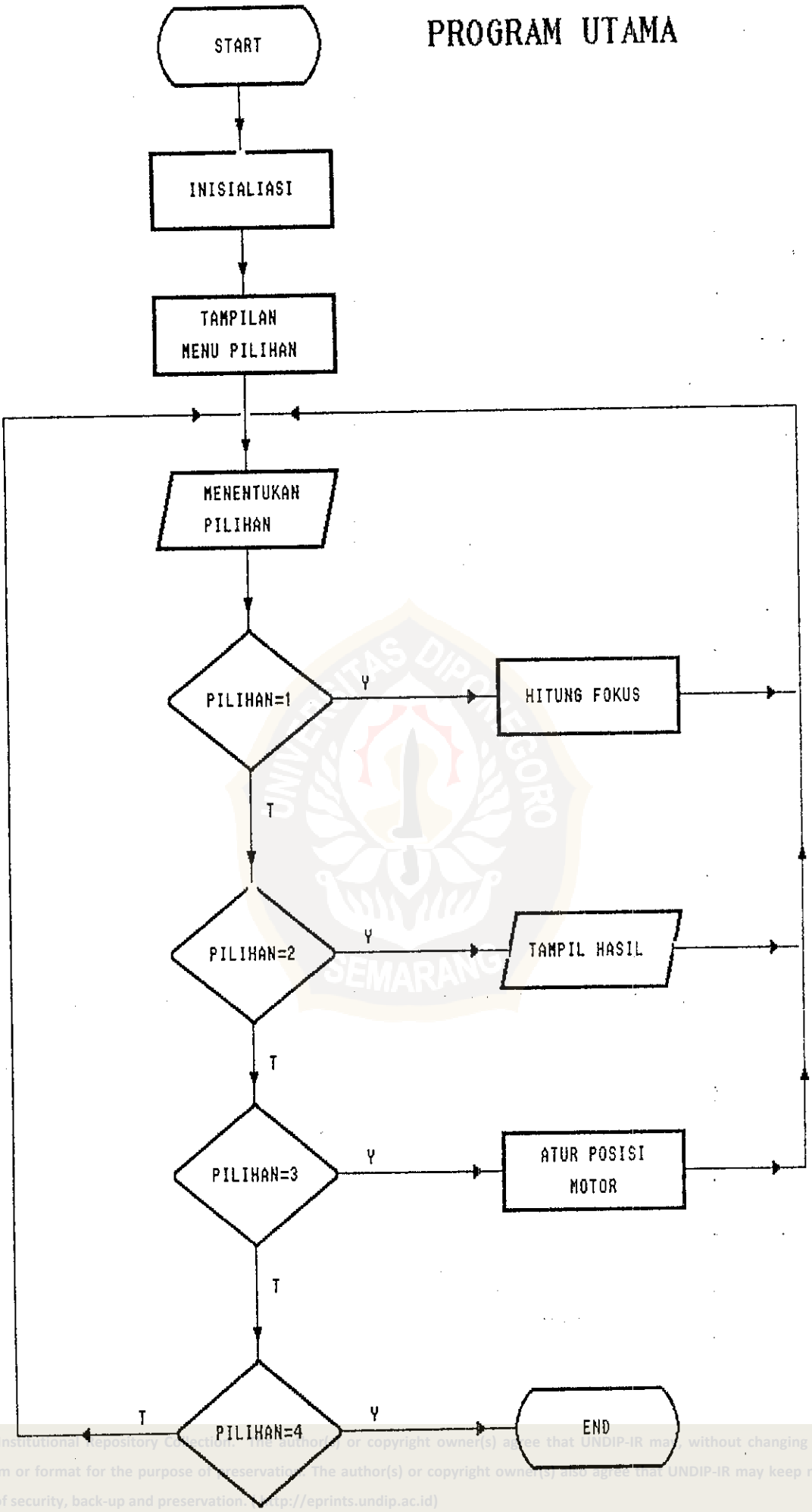
Lampiran 1

GAMBAR RANGKAIAN INSTRUMEN UKUR FOKUS LENSA ELEKTRONIS

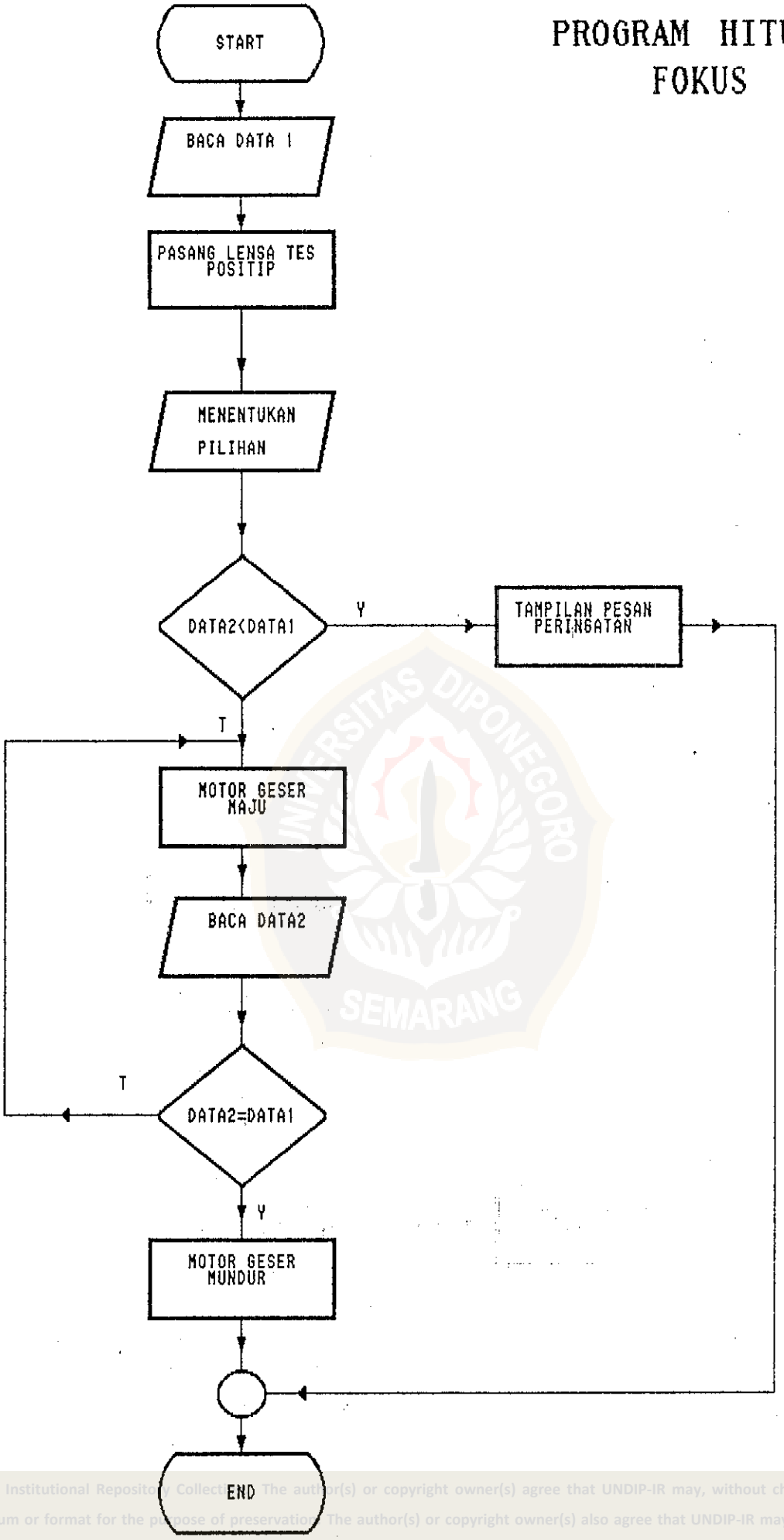


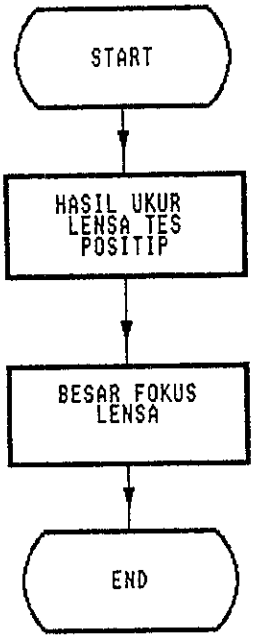
Lampiran 2

PROGRAM UTAMA



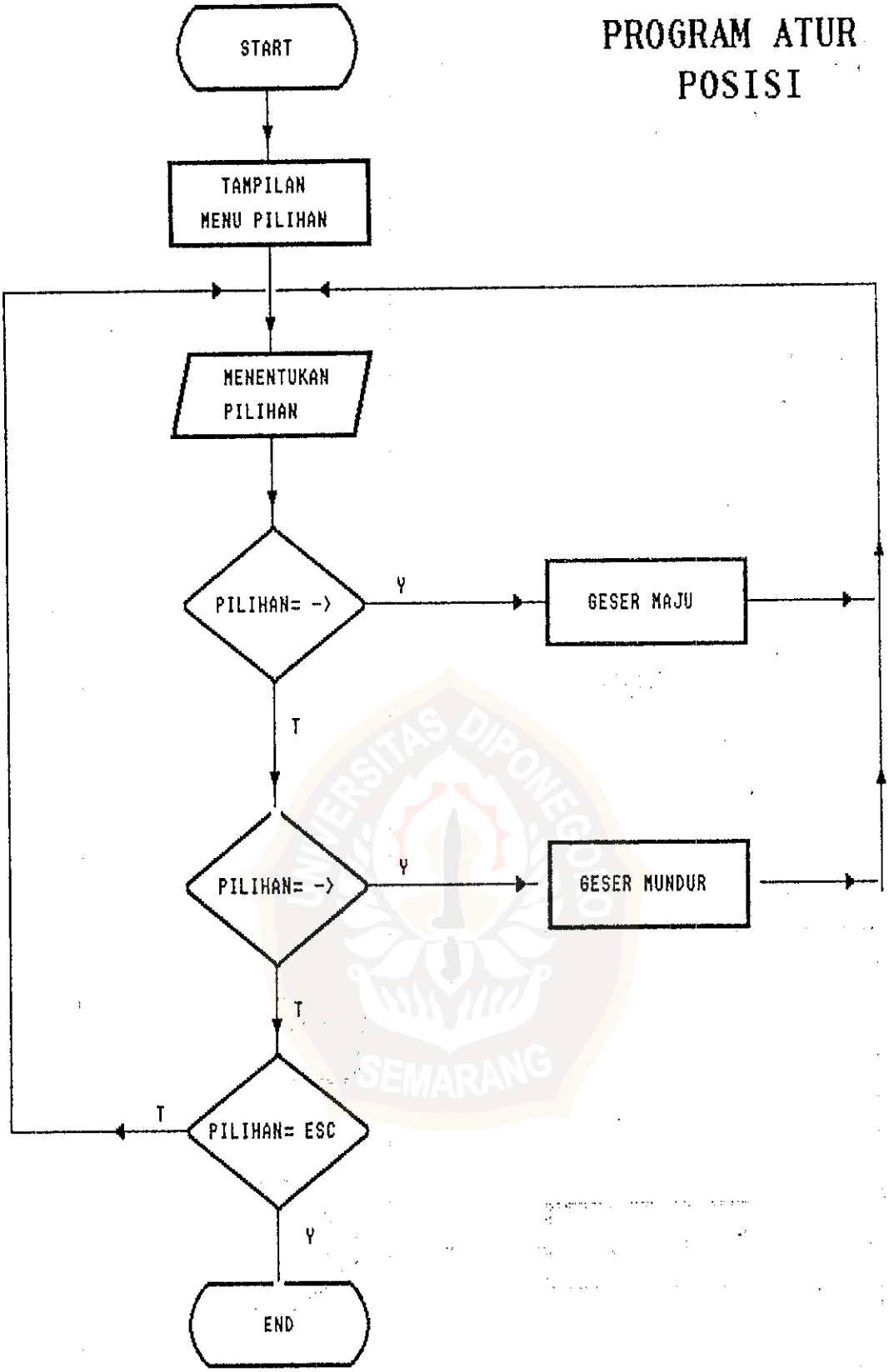
PROGRAM HITUNG FOKUS





PROGRAM TAMPIL HASIL

PROGRAM ATUR POSISI



Lampiran 3

```
#include <dos.h>
#include <stdio.h>
#include <math.h>
void fokus(void);
void tampil(void);
void menu_pilihan(void);
void posisi(void);

int c;
int AA = 50;

main()
{
     outportb (0x307,0x98);  /* Inisialisasi 8255 */
     menu_pilihan ();
}


void menu_pilihan()
     {
     char pilihan;
     clrscr();
     gotoxy(20,12);
     puts("************ MENU PILIHAN ****************");
     gotoxy(20,14);
     puts("**  [1]  Mencari jarak fokus lensa      **");
```

```
gotoxy(20,16);
puts("**  [2]  Menampilkan hasil                **");
gotoxy(20,18);
puts("**  [3]  Mengatur posisi awal             **");
gotoxy(20,20);
puts("**  [4]  Selesai                          **");
gotoxy(20,22);
puts("*****************************************");
gotoxy(20,24);
printf("   Pilihan anda  (1....4) :  ");
do
  pilihan=getch();
  while ( pilihan < '1'  ||   pilihan > '4');
  printf("%c\n\n",pilihan);
  switch(pilihan-'0')
  {
  case 1 : clrscr();
           fokus();
           break;
  case 2 : clrscr();
           tampil();
           break;
  case 3 : posisi();
           break;
  case 4 : exit(0);
  }
}
```

```
void fokus()
      {
      int i,t;
      unsigned int x[2];
      extern int AA;

      outportb(0x305,0);
      outportb(0x305,8);
      delay(100);
      outportb(0x305,0);

      do
      {
       t = inportb(0x306);
       delay(200);
      }
      while(t == 0);

      for(i=0; i<3; i++)
      {
       x[1] = inportb(0x304);
       delay(200);
      }

      gotoxy(10,5);

      printf("Nilai awal : %d " , x[1] );
```

```
gotoxy(10,10);
printf("Pasanglah lensa tes positip !.");

getch();

delay(1000);

outportb(0x305,0);
outportb(0x305,8);
delay(100);
outportb(0x305,0);

do
{
 t = inportb(0x306);
 delay(200);
}
while(t == 0);

for(i=0;i<3;i++)
  {
  x[2]=inportb(0x304);
  delay(200);
  }

if ( x[2] < x[1] )
  {
    c=0;
```

```
do
{
 outportb(0x306,3);delay(AA);
 outportb(0x306,6);delay(AA);
 outportb(0x306,12);delay(AA);
 outportb(0x306,9);delay(AA);


 outportb(0x305,0);
 outportb(0x305,8);
 delay(100);
 outportb(0x305,0);

 do
 {
  t = inportb(0x306);
  delay(200);
 }
 while(t == 0);

 for(i=0;i<3;i++)
 {
 x[2]=inportb(0x304);
 delay(200);
 }

 gotoxy(10,7);
```

```
        printf("Nilai akhir : %d ", x[2]);
        c++;
      }while(x[2] != x[1]);

        outportb(0x306,0);

        for (i= 1; i<=c; i++)
        {
        outportb(0x306,9);delay(AA);
        outportb(0x306,12);delay(AA);
        outportb(0x306,6);delay(AA);
        outportb(0x306,3);delay(AA);
        }
        outportb(0x306,0);
        menu_pilihan();

     }
    else
     {
      clrscr();
      outportb(0x306,0);
      gotoxy(10,5);
      printf("Periksa lagi , mungkin ada kesalahan !.");
      delay(3000);
      menu_pilihan();
    }
  }
```

```
void tampil()
     {
     float f1;
     extern int c,AA;

     f1 = (float)c * 4 *  0.005;

     gotoxy(10,15);

     printf("Jarak pergeseran sumber cahaya : %0.2f mm",f1);

     gotoxy(10,17);

     if ( f1 != 0)
       printf("Fokus lensa ; %d ", 2500/(int)f1 );
     else
       printf("Fokus lensa = 0 ");

     getch();
     menu_pilihan();
     }


void posisi()
     {
     extern int AA;
     char pilihan;
     clrscr();
```

```
   gotoxy(20,8);
   puts("************ MENU PILIHAN ****************");
   gotoxy(20,10);
cprintf("**  %c   Geser Maju           **\r\n",'\x1A');
   gotoxy(20,12);
cprintf("**  %c   Geser Mundur         **\r\n",'\x1B');
   gotoxy(20,14);
   puts("**  Esc  Selesai                          **");
   gotoxy(20,16);
   puts("*****************************************");
   gotoxy(20,18);

   for(;;)
   {
   outportb(0x306,0);
   pilihan=getch();
   if(!pilihan);
   pilihan=getch();
   switch(pilihan)
     {
     case 77 : outportb(0x306,3);delay(AA);
               outportb(0x306,6);delay(AA);
               outportb(0x306,12);delay(AA);
               outportb(0x306,9);delay(AA);
               gotoxy(55,2);
               printf("Maju   ");
               break;
     case 75 : outportb(0x306,9);delay(AA);
```

```
                outportb(0x306,12);delay(AA);
                outportb(0x306,6);delay(AA);
                outportb(0x306,3);delay(AA);
                gotoxy(55,2);
                printf("Mundur ");
                break;
    case 27 : clrscr();
                menu_pilihan();
    }
  }
}
```

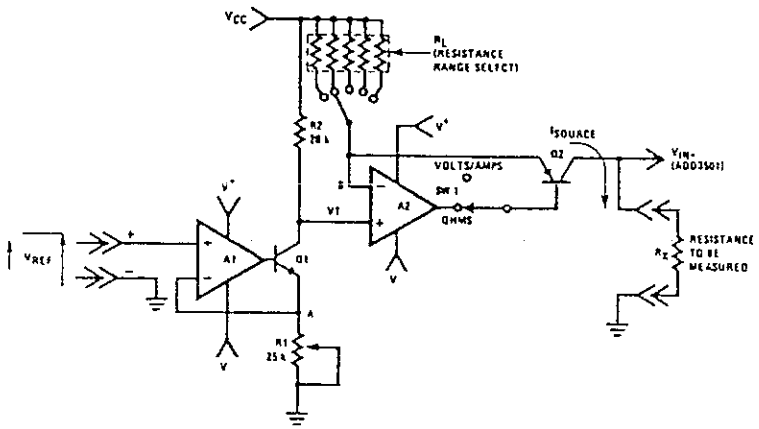


FIGURE 3. Constant-Current Source

PC BOARD LAYOUT

It is imperative to have only one, single-point, analog signal ground connection for the entire system. In a multi-ground layout, the presence of ground-loop resistances will cause the op amps' offset currents and AC response to have a devastating effect on system gain, linearity, and display LSD flicker. Similar precautions must also be taken in the layout of the analog and high-switching-current (digital) paths of the ADD3501.

A FINAL NOTE

The digital multimeter described in this note was developed with the goals of accuracy and low cost. For the high-end DMM market segments, however, improvements to the basic circuit of *Figure 1* are possible in the following areas:

1. Expand the VOLTS mode to include a 200 mV full-scale range;
2. Decrease the full-scale current-measurement loading voltage from 2V to 200 mV; and,
3. Provide a true-rms measurement capability.
4. Increase resolution by substituting the ADD3701—3 ½-digit DVM chip—which is interchangeable and provides a maximum display count of 3,999.

The first 2 improvements involve a dividing down of the ADD3501 feedback loop by a ratio of 10:1, which reduces the 2V full-scale input requirement to 200 mV. This not only allows a 200 mV signal between the ADD3501's V_{IN+} and V_{IN-} inputs to display a full-scale reading, but implies that the maximum voltage dropped across the current-measuring-mode resistance also will be 200 mV. Note, though, that the values of the current-measurement resistors must be scaled down by a factor of ten.

Additionally, a 200 mV full-scale input implies a resolution of 100 μV/LSD. At such low input levels, the DMM may require some clever circuitry to eliminate the gain and linearity distortions that can arise from the offset currents in the AC-to-DC converter.

The third possible improvement—the reading of true-rms values—can be implemented by replacing the AC-to-DC converter of *Figure 2* with National's LH0091, a true-rms-to-DC converter, and appropriate interface circuitry.

Using the ADC0808/ ADC0809 8-Bit μP Compatible A/D Converters with 8-Channel Analog Multiplexer

INTRODUCTION

The ADC0808/ADC0809 Data Acquisition Devices (DAD) implement on a single chip most the elements of the standard data acquisition system. They contain an 8-bit A/D converter, 8-channel multiplexer with an address input latch, and associated control logic. These devices provide most of the logic to interface to a variety of microprocessors with the addition of a minimum number of parts.

These circuits are implemented using a standard metal-gate CMOS process. This process is particularly suitable to applications where both analog and digital functions must be implemented on the same chip.

These two converters, the ADC0808 and ADC0809, are functionally identical except that the ADC0808 has a total unadjusted error of ±½ LSB and the ADC0809 has an unadjusted error of ±1 LSB. They are also related to their big brothers, the ADC0816 and ADC0817 expandable 16 channel converters. All four converters will typically do a conversion in ~100 μs when using a 640 kHz clock, but can convert a single input in as little as ~50 μs.

1.0 FUNCTIONAL DESCRIPTION

The ADC0808/ADC0809, shown in *Figure 1*, can be functionally divided into 2 basic subcircuits. These two subcircuits are an analog multiplexer and an A/D converter. The multiplexer uses 8 standard CMOS analog switches to provide for up to 8 analog inputs. The switches are selectively turned on, depending on the data latched into a 3-bit multiplexer address register.

The second function block, the successive approximation A/D converter, transforms the analog output of the multiplexer to an 8-bit digital word. The output of the multiplexer goes to one of two comparator inputs. The other input is derived from a 256R resistor ladder, which is tapped by a MOSFET transistor switch tree. The converter control logic controls the switch tree, funneling a particular tap voltage to the comparator. Based on the result of this comparison, the control logic and the successive approximation register (SAR) will decide whether the next tap to be selected should be higher or lower than the present tap on the resistor ladder. This algorithm is executed 8 times per conversion, once every 8 clock periods, yielding a total conversion time of 64 clock periods.

When the conversion cycle is complete the resulting data is loaded into the TRI-STATE® output latch. The data in the output latch can then be read by the host system any time before the end of the next conversion. The TRI-STATE capability of the latch allows easy interface to bus oriented systems.

The operation of these converters by a microprocessor or some control logic is very simple. The controlling device first selects the desired input channel. To do this, a 3-bit channel address is placed on the A, B, C input pins; and the ALE input is pulsed positively, clocking the address into the multiplexer address register. To begin the conversion, the START pin is pulsed. On the rising edge of this pulse the internal registers are cleared and on the falling edge the start conversion is initiated.

As mentioned earlier, there are 8 clock periods per approximation. Even though there is no conversion in progress the ADC0808/ADC0809 is still internally cycling through these 8 clock periods. A start pulse can occur any time during this cycle but the conversion will not actually begin until the converter internally cycles to the beginning of the next 8 clock period sequence. As long as the start pin is held high no conversion begins, but when the start pin is taken low the conversion will start within 8 clock periods.

The EOC output is triggered on the rising edge of the start pulse. It, too, is controlled by the 8 clock period cycle, so it will go low within 8 clock periods of the rising edge of the start pulse. One can see that it is entirely possible for EOC to go low before the conversion starts internally, but this is not important, since the positive transition of EOC, which occurs at the end of a conversion, is what the control logic is looking for.

Once EOC does go high this signals the interface logic that the data resulting from the conversion is ready to be read. The output enable (OE) is then raised high. This enables the TRI-STATE outputs, allowing the data to be read. *Figure 2* shows the timing diagram.

2.0 ANALOG INPUTS

2.1 Ratiometric Inputs

The arrangement of the REF(+) and REF(−) inputs is intended to enable easy design of ratiometric converter systems. The REF inputs are located at either end of the 256R resistor ladder and by proper choice of the input voltage several applications can be easily implemented.

Figure 3 shows a typical input connection for ratiometric transducers. A ratiometric transducer is a conversion device whose output is proportional to some arbitrary full-scale value. In other words, the transducer's absolute output value is of no particular concern but the ratio of the output to the full-scale is of great importance. For example, the potentiometric displacement transducers of *Figure 3* have this feature. When the wiper is at midscale, the output voltage $V_O = V_F \times$ (Wiper Displacement) $= V_F \times 0.5$. This enables the use of much less accurate and less expensive references. The important consideration for this reference is noise. The reference must be "glitch free" because a voltage spike during a conversion cycle could cause conversion inaccuracies.

Since highly accurate references aren't required it is possible to use the system power supply as a reference, as shown in *Figure 4*. If the power supply is to be used in this manner supply noise must be kept to a minimum to preserve conversion accuracy. If possible the supply should be well bypassed and separate reference and supply PC board traces, originating as close as possible to the power supply or regulator, should be used. This is illustrated in *Figure 4*. External accessibility of both ends of the resistor ladder enables several variations on these basic connections, as are shown in *Figures 5* and *6*. The magnitude of the reference voltage, $V_{REF} = REF(+) - REF(-)$, can be varied from about ~0.5V to V_{CC}, but the center voltage must be maintained within ±0.1V of $V_{CC}/2$. This constraint is due to the design of the transistor switch tree, which could malfunction if the offset from center scale becomes excessive. Variation of the reference voltage can sometimes eliminate the need for external gain blocks to scale the input voltage to a full-scale range of 5V.

Figure 5 shows a center referencing technique, using two equal resistors to symmetrically offset an LM336 2.5V reference, from both supplies. The offset from either supply is

$$V_{OFF} = \frac{V_{CC} - V_{REF}}{2} = 1.25V$$

These resistors should be chosen so that they limit current through the LM336 to a reasonable value, say 5 mA. The total resistor current is:

$$I_R = I_{REF} + I_{LADDER} + I_{TRAN}$$

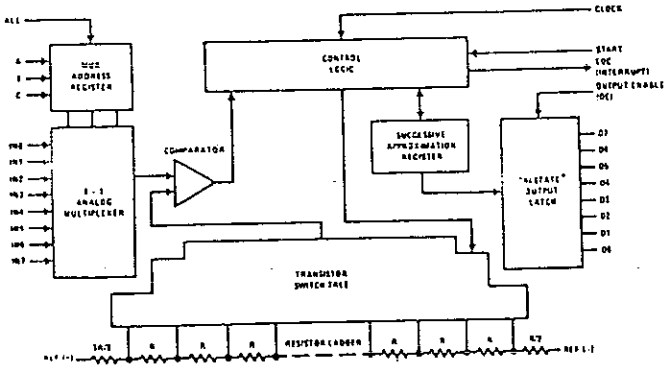


FIGURE 1. ADC0808/ADC0809 Functional Block Diagram

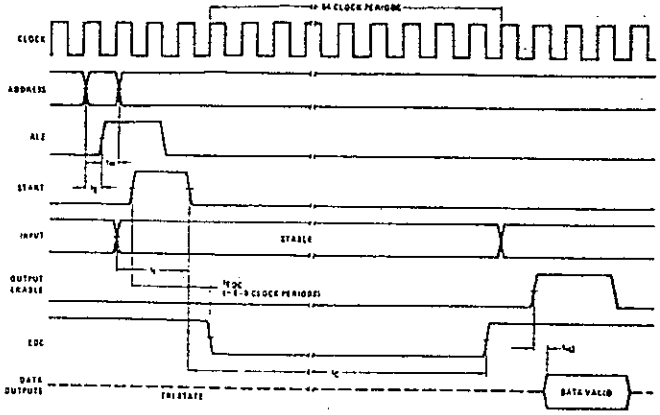


FIGURE 2. ADC0808/ADC0809 Timing Diagram

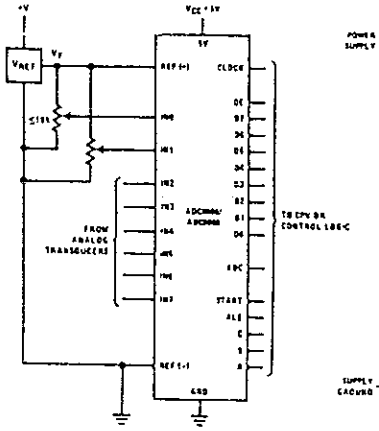


FIGURE 3. Ratiometric Converter with Separate Reference

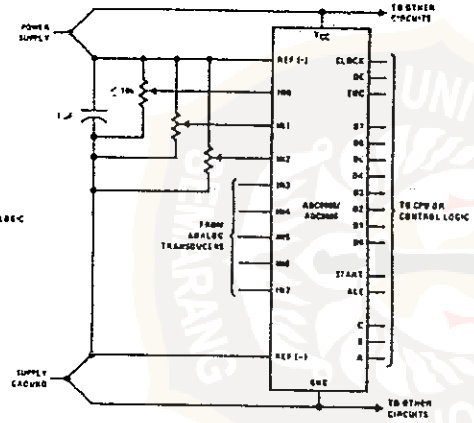


FIGURE 4. Ratiometric Converter with Power Supply Reference

where I_{LADDER} is the 256R ladder current, I_{TRAN} is the current through all the transducers, and I_{REF} is the current through the reference. R1 and R2 should be well matched and track each other over temperature.

For odd values of reference voltage, the reference could be replaced by a resistor, but due to loading and temperature problems, these resistors should be buffered to the REF(+) and REF(−) inputs, *Figure 6*. The power supply must be well bypassed as supply glitches would otherwise be passed to the reference inputs. The reference voltage magnitude is:

$$V_{REF} = V_{DD}\left(\frac{R2}{2R1 + R2}\right) \text{ For } R3 = R1$$

There are several op amps that can be used for buffering this ladder. Without adding another supply, an LM358 could be used if the REF(+) input is not to be set above 3.5V. The LM10 can swing closer to the positive supply and can be used if a higher $V_{REF(+)}$ voltage is needed.

As the REF(+) to REF(−) voltage decreases the incremental voltage step size decreases. At 5V one LSB represents ~20 mV, but at 1V, one LSB represents ~4 mV.

As the reference voltage decreases, system noise will become more significant so greater precaution should be enforced at lower voltages to compensate for system noise; i.e., adequate supply and reference bypassing, and physical as well as electrical isolation of the inputs.

2.2 Absolute Analog Inputs

The ADC0808/ADC0809 may have been designed to easily utilize ratiometric transducers, but this does not preclude the use of non-ratiometric inputs. A second type of input is the absolute input. This is one which is independent of the reference. This implies that its *absolute* numerical voltage value is very critical, and to accurately measure this voltage the accuracy of the reference voltage becomes equally critical. The previous designs can be modified to accommodate absolute input signals by using a more accurate reference. In *Figure 4* the power supply reference could be replaced by LM336-5.0 reference. R1 and R2 of *Figure 6*, and R1 and R3 of *Figure 7* may have to be made more accurately equal.

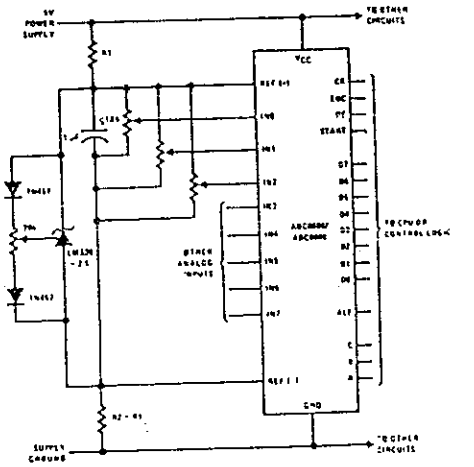


FIGURE 5. Mid-Supply Centered Reference Using LM336 2.5V Reference

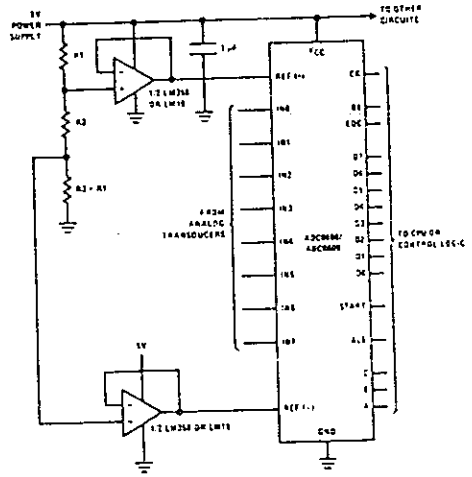


FIGURE 6. Mid-Supply Centered Reference Using Buffered Resistors

In some small systems it is possible to use the precision reference as the power supply as shown in *Figure 7*. An unregulated supply voltage > 5V is required, but the LM336-5.0 functions as both a regulator and reference. The dropping resistor R must be chosen so that, for the whole range of supply currents needed by the system, the LM336-5.0 will stay in regulation. As in *Figure 4* separate supply and reference traces should be used to maintain a noiseless supply.

If the system requires more power, an op amp can be used as shown in *Figure 8* to isolate the reference and boost the supply current capabilities. Here again, a single unregulated supply is required.

2.3 Differential Inputs

Differential measurements can be obtained by playing a tle software trick. This simply involves sequentially conv ing two channels then subtracting the two results. For ample, if the difference voltage between channel 1 and required, merely convert channel 1 and read the res Then convert channel 2, input the result, and subtrac from the first result. (See *Figure 9*.) When using this pro dure, both input signals must be stable throughout both c version times or the end result will be incorrect. One wa get around this is to use two sample/holds which are sa pled at the same time.

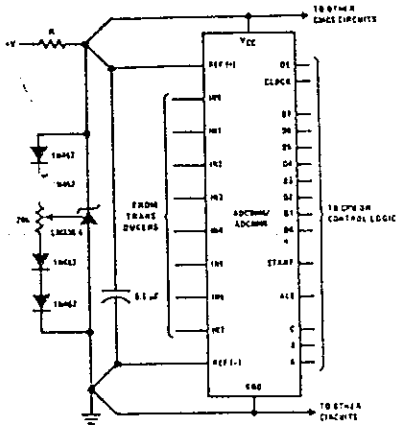


FIGURE 7. Precision Reference used as a Power Supply

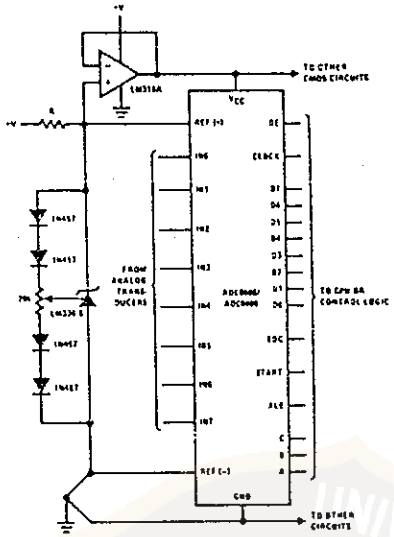


FIGURE 8. Precision Reference Buffered for Power Supply

A second method is to use two chips to convert a differential channel, *Figure 10*. Typically each channel 1 would be connected to opposite sides of the differential input. Both converters are started simultaneously. When both converters' EOC outputs go high the output of the AND gate will go high indicating that the data is ready to be read.

The circuit in *Figure 10* can be slightly modified to provide increased data throughput by using two converters in a parallel data acquisition scheme. *Figure 11* shows this circuit in which all the input channels are connected in pairs through LF398 monolithic sample/holds. Under normal operation a sample/hold is accessed through an MM74C42 which will pulse an MM74C221, generating a sample pulse. After a sample/hold is done sampling the signal, the appropriate channel is started. If this process is alternated between two converters the sample rate can be doubled.

2.4 Analog Input Considerations

Analog inputs into the ADC0808/ADC0809 can handle any input signal that is maintained within the supply limits, but some careful consideration must be given to the output impedance of the transducer or buffer. Using transducers with large source impedances can cause errors due to comparator input currents.

To understand the nature of these currents a short discussion of comparator operation is required. *Figure 12* shows a simplified model of the comparator and multiplexer. This comparator alternately samples the input voltage and the ladder voltage. As it samples the input, C_C and C_P are charged up to the input voltage. It then samples the ladder and discharges the capacitor. The net charge difference is determined by a modified inverter chain and results in a 1 or 0 state at the output.

Eight samples are made per conversion, resulting in eight spikes of varying magnitude on the input.

If the source resistance is large, it adds to the RC time constant of the switched capacitor which will inhibit the input from settling properly, causing errors. As one might expect, the maximum source resistance allowable for accurate conversions is inversely proportional to clock frequency. This resistance should be ≤1 kΩ at 1.2 MHz and ≤2 kΩ at 640 kHz. If a potentiometer-type ratiometric transducer is used it should be ≤5 kΩ at 1.2 MHz and ≤10 kΩ at 640 kHz.

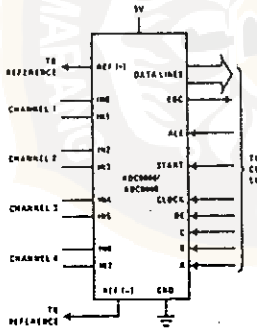


FIGURE 9. Software Controlled Differential Converter

If large source impedances are unavoidable (≥2 kΩ at 640 kHz), the transient errors can be reduced by placing a bypass capacitor ≥0.1 μF from the analog inputs to ground. This will reduce the spikes to a small average current which will cause some error as well, but this can be much less than the error otherwise incurred. The maximum voltage error for a potentiometer input with a bypass capacitor added is:

$$V_{ERR} \approx \left[\frac{R_{POT}}{5}(I_{IN})\frac{Ck}{640\ \text{kHz}}\right] V$$

where R_{POT} = total potentiometer resistance; I_{IN} = maximum input current at 640 kHz, 2 μA; and Ck = clock frequency.

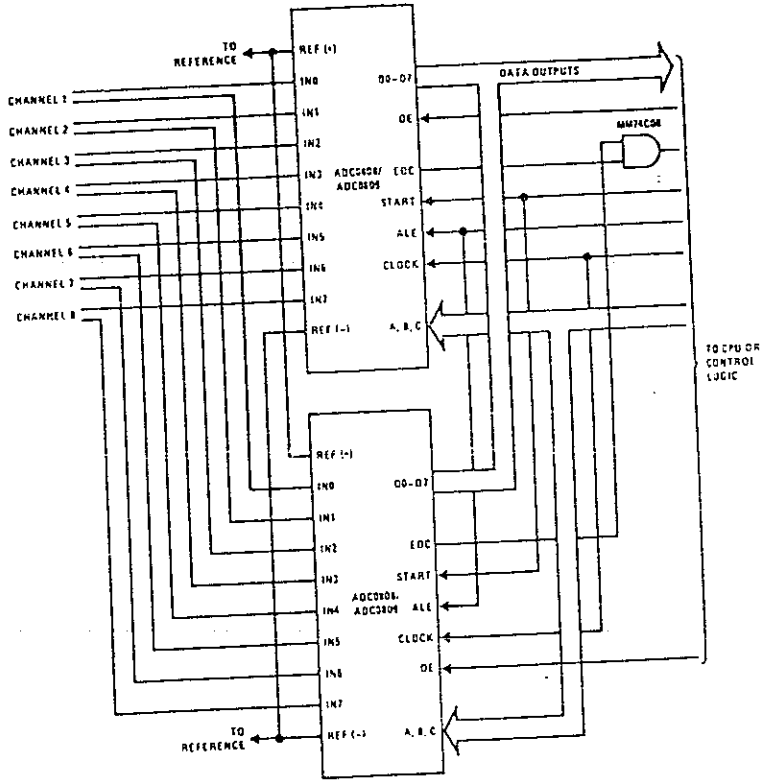


FIGURE 10. Dual Converter Differential Circuit

For standard buffer source impedance the maximum error is:

$$V_{ERR} = \left[I_{IN}R_S\left(\frac{Ck}{640\ \text{kHz}}\right)\right] V$$

where R_S = buffer source resistance; I_{IN} = the maximum input current at 640 kHz, 2 μA; and Ck = clock frequency.

3.0 MICROPROCESSOR INTERFACING

The ADC0808/ADC0809 converters were designed to interface to most standard microprocessors with very little external logic, but there are a few general requirements which must be considered to ensure proper converter operation. Most microprocessors are designed to be TTL compatible and, due to speed and drive requirements, incorporate many TTL circuits. The data outputs of the ADC0808/ADC0809 are capable of driving one standard TTL load which is adequate for most small systems, but for larger systems extra buffering may be necessary. The EOC output is not quite as powerful as the data outputs, but normally it is not bussed like the data outputs.

The converter inputs are standard CMOS compatible inputs. When TTL outputs are connected to any of the digital inputs a pull-up resistor should be tied from the TTL output to V_{CC} ~ 5 kΩ. This will ensure that the TTL will pull-up above 3.5V.

Usually the converter clock will be derived from the microprocessor system clock. Some slower microprocessor clocks can be used directly, but at worst a few div stages may be necessary to divide microprocessor frequencies above 1.2 MHz to a usable value.

The timing of the START and ALE pulses relative to chan selection and signal stability can be critical. The simpl approach to microprocessor interfaces usually ties START and ALE together. When these lines are strobed the address is strobed into the address register and the conversion is started. The propagation delay from ALE to comparator input of the selected input signal is about ~ 2 (input source resistance << 1 kΩ). If the start pulse is short the comparator can sample the analog input before it is stable. When using a slow clock ≤500 kHz the sample period of the comparator input is long enough to allow delay to settle out.

If the ADC0808/ADC0809 clock is >500 kHz, a delay between the START and ALE pulses is required. There three basic methods to accomplish this. The first possibility is to design the microprocessor interface so that the START and ALE inputs are separately accessible. This is simple some extra address decoding is available. Separate access of the START and ALE pins allows the microprocessor, via software, to set the delay time between the START and ALE pulses.

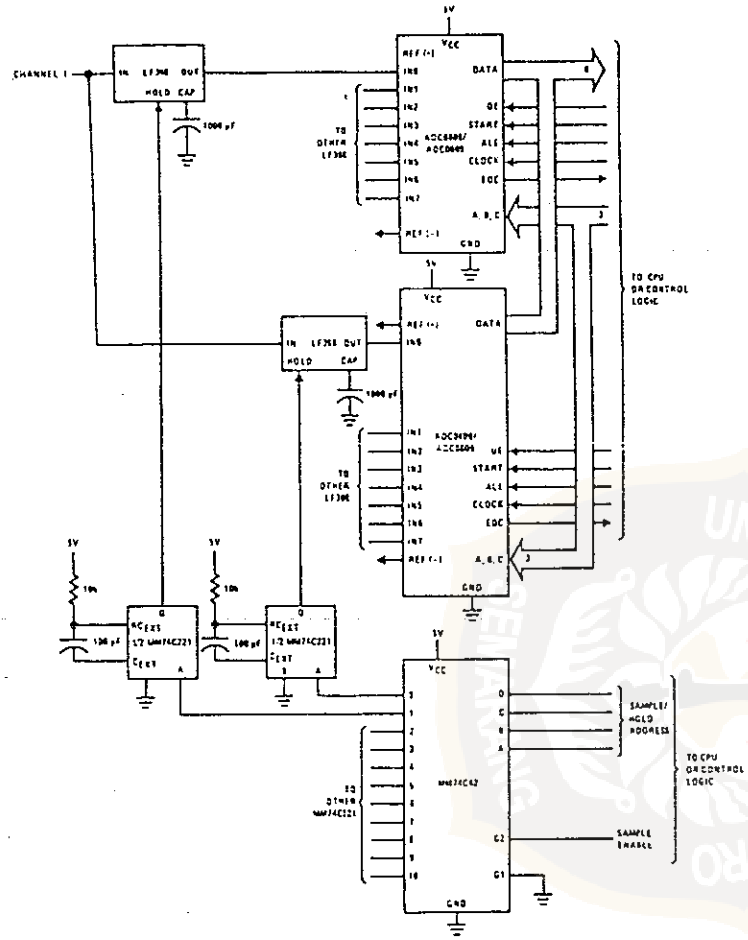


FIGURE 11. Parallel Data Acquisition with Sample/Holds

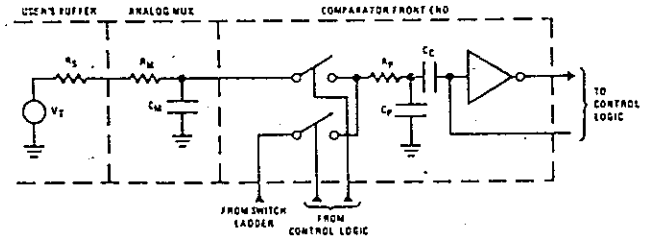


FIGURE 12. Analog Multiplexer and Comparator Input Model

If extra decoding is not available, then START and ALE could be tied together. To obtain the proper delay, the microprocessor would cause START/ALE to be strobed twice by executing the load and start instruction twice. The first time this instruction is executed, the new channel address is loaded and the conversion is started. The second execution of this instruction will reload the same channel address and restart the conversion. But since the multiplexer address register contents are unchanged the selected analog input will have already settled by the time the second instruction is issued. Actual implementations of these ideas are shown in following sections.

A third possibility when ALE and START are tied together is to stretch the microprocessor derived ALE/START pulse by inserting a one-shot at these inputs and creating a positive pulse >3 μs. Since ALE loads the multiplexer register on the positive going edge of the pulse and START begins the conversion on the falling edge, the width of the pulse sets the ALE to START delay time.

Most microprocessor interfaces would be designed such that a START pulse is issued by a memory or I/O write instruction, although a memory or I/O read can be used. The ALE strobe on the other hand, requires a write by the CPU when A, B, and C are connected to the data bus, and could use a read instruction if A, B, and C are connected to the address bus, but the software could get confusing. The logic to derive the OE strobe must be connected to the microprocessor so that a memory or I/O read instruction will cause OE to be pulsed. A read is required since the ADC0808/ADC0809 data must be read.

3.1 Interfacing to the 8080

The simplest interface would contain no address decoding which may seem unreasonable; but if the system ports are I/O mapped, up to 8 of them can be connected to the CPU with no decoding. Each of the 8 I/O address lines would serve as a simple port enable line which would be gated with read and write strobes to select a particular port. This scheme is shown in *Figure 13*. A7 is the address line used and, whenever it is zero and an I/O read or write is low, the port is accessed. This implementation shows A, B, C connected to D0, D1, D2 causing the information on the data bus to select the channel, but A, B, and C could be connected to the address bus, with a loss of only 3 ports. Address decoding schemes are tabulated in *Figure 14*. (Remember A, B, C inputs are only valid when selecting a channel to convert, and are not used to read data.)

Two LSTTL NOR gates are used to generate the ADC0808/ADC0809 read/write strobes. When the 8080 writes to the ADC0808/ADC0809 the ALE and START inputs are strobed, loading and starting the conversion. When the CPU reads the ADC0808/ADC0809 the OE input is taken high and the data outputs are enabled.

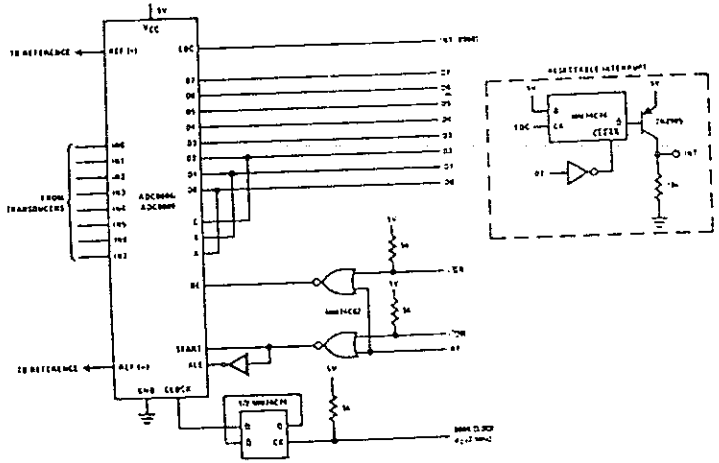


FIGURE 13. Minimum 8080/8224/8228 Interface

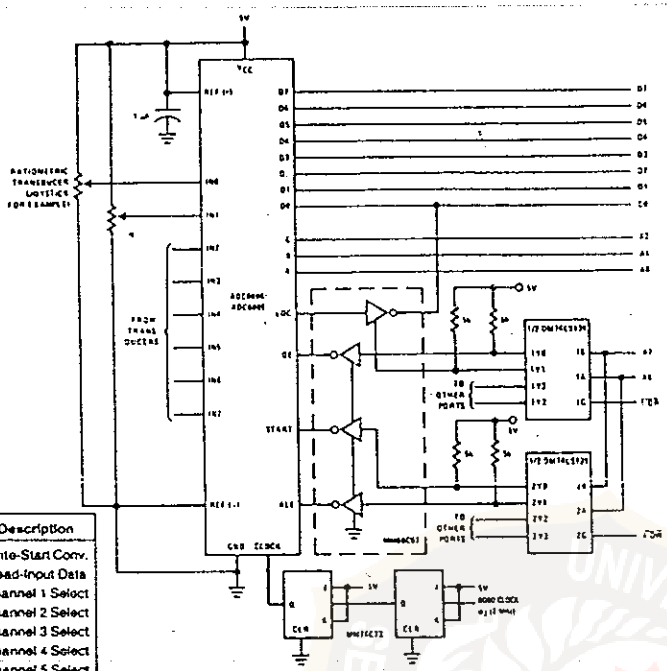
A7	A6	A5	A4	A3	A2	A1	A0	D2	D1	D0	Output Port Description
1	1	1	1	1	1	1	0	X	X	X	Spare Port
1	1	1	1	1	1	0	1	X	X	X	Spare Port
1	1	1	1	1	0	1	1	X	X	X	Spare Port
1	1	1	1	0	1	1	1	X	X	X	Spare Port
1	1	1	0	1	1	1	1	X	X	X	Spare Port
1	1	0	1	1	1	1	1	X	X	X	Spare Port
1	0	1	1	1	1	1	1	X	X	X	Spare Port
0	1	1	1	1	1	1	1	0	0	0	Channel 0 Port
0	1	1	1	1	1	1	1	0	0	1	Channel 1 Port
0	1	1	1	1	1	1	1	0	1	0	Channel 2 Port
0	1	1	1	1	1	1	1	0	1	1	Channel 3 Port
0	1	1	1	1	1	1	1	1	0	0	Channel 4 Port
0	1	1	1	1	1	1	1	1	0	1	Channel 5 Port
0	1	1	1	1	1	1	1	1	1	0	Channel 6 Port
0	1	1	1	1	1	1	1	1	1	1	Channel 7 Port

FIGURE 14a. Write Address Decoding for 8080 Output Ports (A, B, C Connected to D0, D1, D2)

A7	A6	A5	A4	A3	A2	A1	A0	Output Port Description
0	1	1	1	1	0	0	0	Channel 0 Port
0	1	1	1	1	0	0	1	Channel 1 Port
0	1	1	1	1	0	1	0	Channel 2 Port
0	1	1	1	1	0	1	1	Channel 3 Port
0	1	1	1	1	1	0	0	Channel 4 Port
0	1	1	1	1	1	0	1	Channel 5 Port
0	1	1	1	1	1	1	0	Channel 6 Port
0	1	1	1	1	1	1	1	Channel 7 Port
1	1	1	1	0	X	X	X	Spare Port
1	1	1	0	1	X	X	X	Spare Port
1	1	0	1	1	X	X	X	Spare Port
1	0	1	1	1	X	X	X	Spare Port

X = "don't care"

FIGURE 14b. Modified Write Address Decoding for 8080 Output Ports (A, B, C Connected to A0, A1, A2)



Address A7–A0	Description
0 0 X X X X X X	Write-Start Conv.
0 0 X X X X X X	Read-Input Data
0 1 X X X 0 0 0	Channel 1 Select
0 1 X X X 0 0 1	Channel 2 Select
0 1 X X X 0 1 0	Channel 3 Select
0 1 X X X 0 1 1	Channel 4 Select
0 1 X X X 1 0 0	Channel 5 Select
0 1 X X X 1 0 1	Channel 6 Select
0 1 X X X 1 1 0	Channel 7 Select
0 1 X X X 1 1 1	Channel 8 Select
0 1 X X X X X X	Read-Input EOC

FIGURE 15. 8080/8224/8228 Interface Using Partial Decoding

Figure 13 implements a simple interrupt concept where EOC is tied directly to the 8080 interrupt input. When the INS8228 is used and the INTA pin is tied high through a 1 kΩ resistor, the interrupt will cause a restart, RST, instruction to be executed, which will then cause a jump to a restart vector and execution of the interrupt routine. If a very simple multi-interrupt system is desired, a wire OR'ed configuration employing resettable latches as shown in *Figure 13's* inset can be used. In this simple design the MM74C74 is reset when the ADC0808/ADC0809 data is read. If more complicated interrupt structures are required, then an interrupt controller is usually the best solution.

The I/O port address structure for *Figure 13's* implementation is shown in *Figure 14a*. If the A, B, C inputs are tied to A0, A1, A2 inputs the port structure is as shown in *Figure 14b*. The later method makes each channel look like a separate port address, whereas if A, B, C are tied to the data bus the ADC0808/ADC0809 looks like one start conversion port address whose channel is selected by the 3-bit status word written to it on the data bus.

Figure 15 shows a slightly more complex interface, where the address is partially decoded by a DM74LS139, dual 2-4 line decoder which creates the read and write strobes to operate the converter. This design interfaces to the processor in a polled type of interface. An MM80C97 TRI-STATE buffer is used to buffer the EOC line to the data bus, as well as provide the correct level for the START, ALE, and OE pulses. The converter clock is a divided 8080 system clock.

Typically, the software to use *Figure 15* would first select the desired channel by writing the channel address to the ALE port address, 01XXXCBA, where X = don't care, and CBA is the channel address. Next the conversion is started by writing to the START address, 00XXXXXX. Now the processor must wait a few instruction cycles to allow EOC to fall. Once EOC falls, its status can be checked by reading the EOC line, address 01XXXXXX. When the EOC line is detected high again (a low on D0), the data can be read by accessing the OE port, address 00XXXXXX. As in the previous example the A, B, C inputs can be tied to D0, D1, D2 rather than A0, A1, A2, so that the information on the data bus selects the channel to be converted. *Figure 15* can be connected in an interrupt mode by incorporating the interrupt flip-flop of *Figure 13*.

A few typical utility routines to operate the ADC0808/ADC0809 application in *Figure 13* are shown in *Figure 16a*. These routines assume that the resettable interrupt flip-flop is used. *Figure 16b* illustrates some typical polled I/O routines for *Figure 15*. Notice that in *Figure 16a* the CUT START1 instruction is executed twice to allow the analog input signal to settle as discussed earlier.

The application in *Figure 17* uses a 8-bit bus comparator and a few gates to decode a read and write strobe. Viewed from the CPU this interface looks like a bidirectional data port whose address is set by the logic levels on the T_n inputs of the DM8131 comparator. When data is written to the ADC0808/ADC0809 the 3 least significant bits on the address bus define the channel to be converted. The rest of the bits are decoded to provide the START and ALE strobes. When the conversion is completed EOC sets the interrupt flip-flop, and when the data is read the interrupt is reset.


```
;
;
; START CONVERSION (A, B, C CONNECTED TO D0, D1, D2)
;
CHANN1      EQU   7
START1      EQU   7FH
DATA        EQU   7FH
;
START:      LDA   CHANN1   ; LOAD CHANNEL ADDRESS INTO ACE
            OUT   START1   ; STORE IT TO ADC0808/ADC0809 AND START
            OUT   START1   ; RESTART ADC0808/ADC0809 TO ACCOUNT FOR
                           ; MULTIPLEXER DELAY
;           EI             ; ENABLE INTERRUPTS IF NOT ALREADY
            —     —        ; PROCESS PROGRAM
;
; INTERRUPT HANDLER ROUTINE
;
INTRP:      IN    DATA     ; READ DATA AND RESET INTERRUPT
            —     —        ; PROCESS DATA
            EI             ; ENABLE INTERRUPTS IF DESIRED
            RET            ; RETURN TO MAIN PROGRAM
```

FIGURE 16a. Typical 8080 Resettable Interrupt I/O Routines

```
;
; START CONVERSION (A, B, C CONNECTED TO A0, A2, A3) AND POLL EOC
; (FIGURE 15)
SELECT      EQU   40H      ; SELECT CHANNEL 0
START       EQU   00H      ; START CONVERTER
EOCIN       EQU   40H      ; READ EOC
DATA        EQU   00H      ; READ DATA
START:      OUT   SELECT   ; SELECT CHANNEL
            OUT   START    ; START CONVERSION
            NOP            ; INSERT INSTRUCTIONS TO WAIT 0-8
            NOP            ; CLOCK PERIODS OF ADC0808/ADC0809 CLOCK
            NOP            ; FOR EOC TO DROP (8NOPs MINIMUM)
            NOP
            NOP
;
; READ AND TEST EOC
;
STATUS:     IN    EOCIN    ; INPUT EOC BIT
            ANI   01H      ; MASK OUT OTHER BITS
            JZ    READY    ; IF INPUT BIT IS ZERO JUMP READY
            —     —        ; ELSE CONTINUE EXECUTING PROGRAM
; OR
; CONTINUOUS POLLING ROUTINE
;
STAT 2:     IN    EOCIN    ; INPUT EOC STATUS BIT
            ANI   01H      ; MASK OUT ALL BITS BUT D0
            JNZ   STAT 2   ; JUMP TO TRY AGAIN IF NOT READY
READY:      IN    DATA     ; IF READY INPUT DATA
            —     —        ; CONTINUE EXECUTING PROGRAM
```

FIGURE 16b. Typical Polled I/O Routines for ADC0808/ADC0809

Both the decoder and the bus comparator methods of address decoding have their own advantages. Bus comparators will more completely decode addresses but are capable of only a limited number of port strobes. Decoders, on the other hand, provide less decoding but more port strobes. There is a trade off for minimum parts systems as far as which route to go, and it will depend on the CPU and type of system.

3.2 Interfacing to the 6800

The ADC0808/ADC0809 easily interface to more than one microprocessor. The 6800 can also be used to control the converter. The 6800 has no separate I/O address space so all I/O transfers must be memory mapped. In general more address decoding logic is required to ensure that the I/O ports don't overlap existing memory. For small systems a partial address decoding scheme is shown in *Figure 18*. Generally, if several ports are desired, a small block of memory would be set aside, as is accomplished by the DM8131. *Figure 18* also illustrates a typical 6800 interrupt scheme using a flip-flop and open collector transistor. The interrupt is reset when the data is read. If more ports are needed, a decoder could be added as shown in *Figure 19*. *Figure 19* also illustrates a polled I/O mode using TRI-STATE buffer to gate EOC onto the data bus. As with the INS8080 the A, B, C inputs of the ADC0808/ADC0809 can be connected to the address bus or the data bus.

The 6800 differs from the INS8080 in that the 6800 has a single read/write (R/$\overline{W}$) strobe and a valid memory address (VMA), whereas the INS8080 has separate read and write strobes ($\overline{I/OR}$ and $\overline{I/OW}$). Normally, to obtain a read pulse, VMA, R/$\overline{W}$ and ϕ_2 are gated together and, for a write R/$\overline{W}$ is inverted. ϕ_2 is the 6800 phase 2 system clock. Also notice that the 6800 $\overline{INT}$ interrupt input is active low. This enables a standard wired-OR open collector design to be implemented.

Figure 20 illustrates some typical 6800 software utility routines for either polled or interrupt interfaces. Again notice double start instructions.

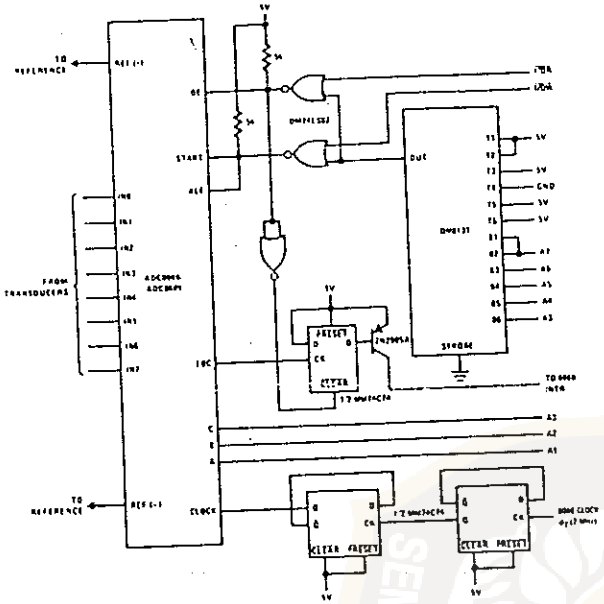


FIGURE 17. Interrupt-Type 8080/8224/8228 Interface Using 6-Bit Bus Comparator

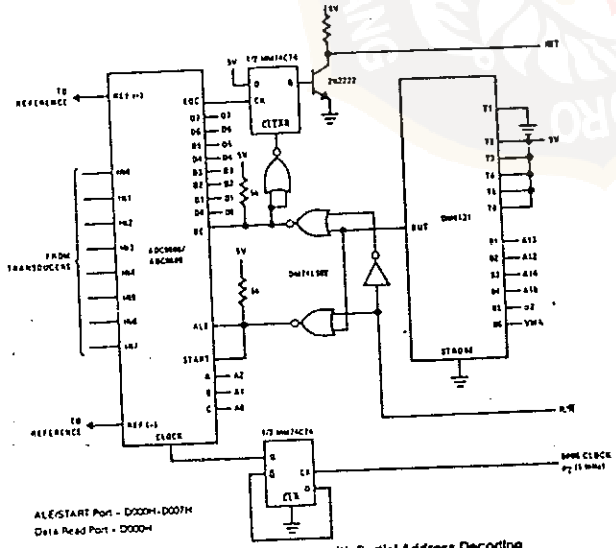


FIGURE 18. Typical 6800 Interface with Partial Address Decoding

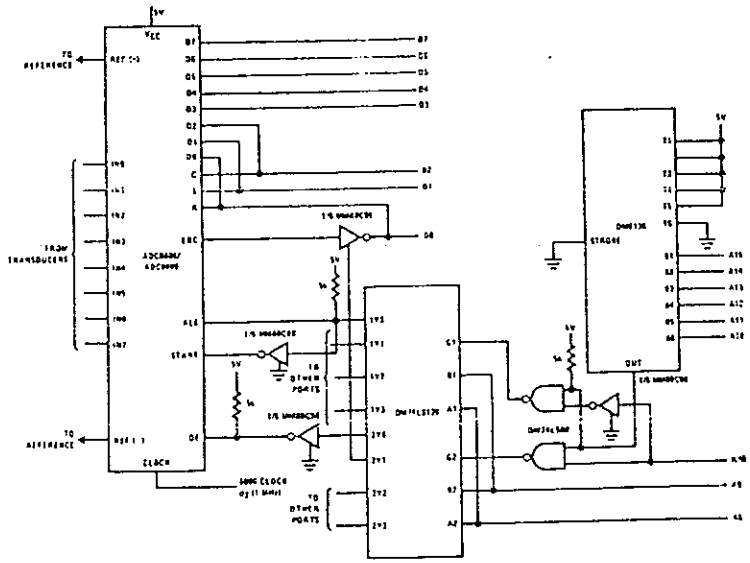


FIGURE 19. Full Decoded 6800 Interface Address

3.3 Z80 Interface

Interfacing the Z80 to the ADC0808 is much the same as interfacing to an 8080/8224/8228 CPU group. CPU instruction timing is very similar, except the read/write control signals are slightly different. Instead of the $\overline{I/OW}$ write strobe there is the $\overline{IOREQ}$ and $\overline{WR}$ and instead of $\overline{I/OR}$, $\overline{IOREQ}$ and $\overline{RD}$ are supplied.

Figure 21 shows a very simple Z80 interface, which is similar to the INS8080 interface of *Figure 13*, except that the interrupt flip-flop design is closer to the 6800 designs. This is because the Z80 $\overline{INT}$ is active low as is the 6800, but the INS8080 INT is active high.

Figure 22 shows a fully decoded bus comparator design where the DM8131 decodes 5 address bits and the $\overline{IOREQ}$ I/O request strobe. Two NOR gates gate the $\overline{RD}$ and $\overline{WR}$ strobes for ALE, START and OE inputs.

```
*
*UTILITY ROUTINES FOR ADC0808/ADC0809 INTERFACE
*
*
*LOAD AND START CONVERSION (FIGURE 18)
*
STATUS    EQU    $D600    START ADDRESS FOR CHANNEL 0
DATA      EQU    $D800    CONVERTER DATA ADDRESS
*
*
*
START     STA    STATUS   SELECT CHANNEL 0 AND START
          STA    STATUS   DO AGAIN TO LET INPUTS SETTLE
          LDX    #VECTOR  LOAD INTERRUPT VECTOR ADDRESS
          STX    $FFF8    STORE IT
          ——               EXECUTE MISC PROGRAM
          ——
          ——
          CLI              ENABLE INTERRUPT IF NOT ALREADY
          ——               EXECUTE MISC PROGRAM
          WAI              WAIT FOR INTERRUPT
*
*INTERRUPT HANDLER (FIGURE 18)
*
VECTOR    LDAA   DATA     LOAD DATA RESET INTERRUPT
          CLI              ENABLE INTERRUPTS (OPTION)
          ——               EXECUTE PROGRAM
          RTI              RETURN TO MAIN PROGRAM
```

*START AND TEST CONVERSION POLLED MODE (FIGURE 19)

```
*
DATA2     EQU   $F600    CONVERTER DATA ADDRESS
CHANN2    EQU   02       CHANNEL 2 ADDRESS
EOCIN     EQU   $F900    EOC INPUT PORT
START2    LDAA  CHANN2   LOAD A ACCUMULATOR
          STAA  STATUS   LOAD ADDRESS AND START
          NOP            WAIT
          STAA  STATUS   RESTART TO LET MUX SETTLE
          NOP            8 NOPS TO WAIT FOR EOC
          ——             TO GO LOW
          LDAA  EOCIN    LOAD EOC STATUS BIT
          ANDA  01       MASK BITS 1-7
          BEQ   READY    IF A = 0 THEN CONVERTER DONE
*
*
          ——             EXECUTE MISC PROGRAM
*
```

*CONTINUOUS POLLING OF EOC (FIGURE 19)

```
*
*
POLLIT    LDAA  EOCIN    LOAD EOC STATUS
          ANDA  CHANN2   MASK MSBs
          BNE   POLL IT  IT ACC ≠ 0 NOT READY, LOOP
READY LDAA *             ELSE READ DATA
          ——             CONTINUE PROGRAM
          ——
```

FIGURE 20. Typical I/O Routines for ADC0808/ADC0809 and 6800 Interface

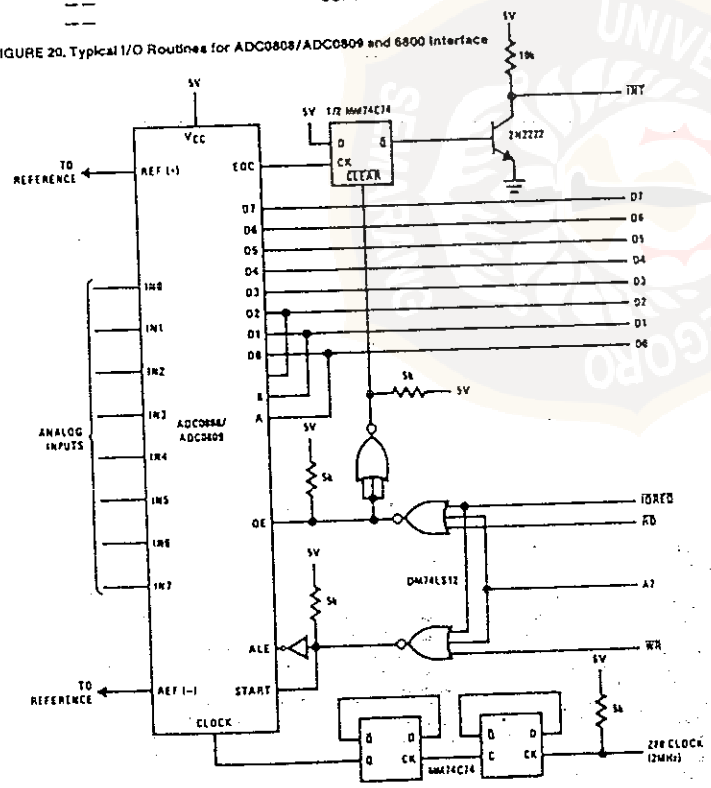


FIGURE 21. Simple Z80 Interface

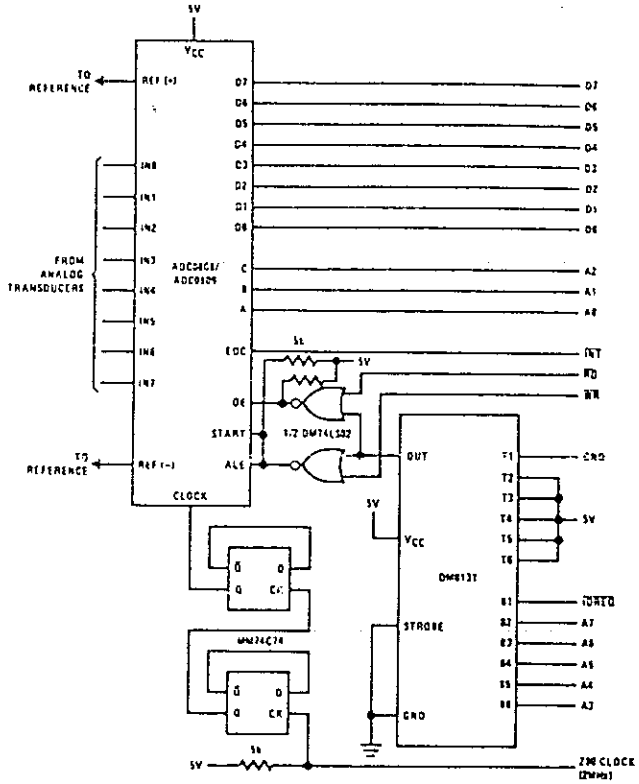


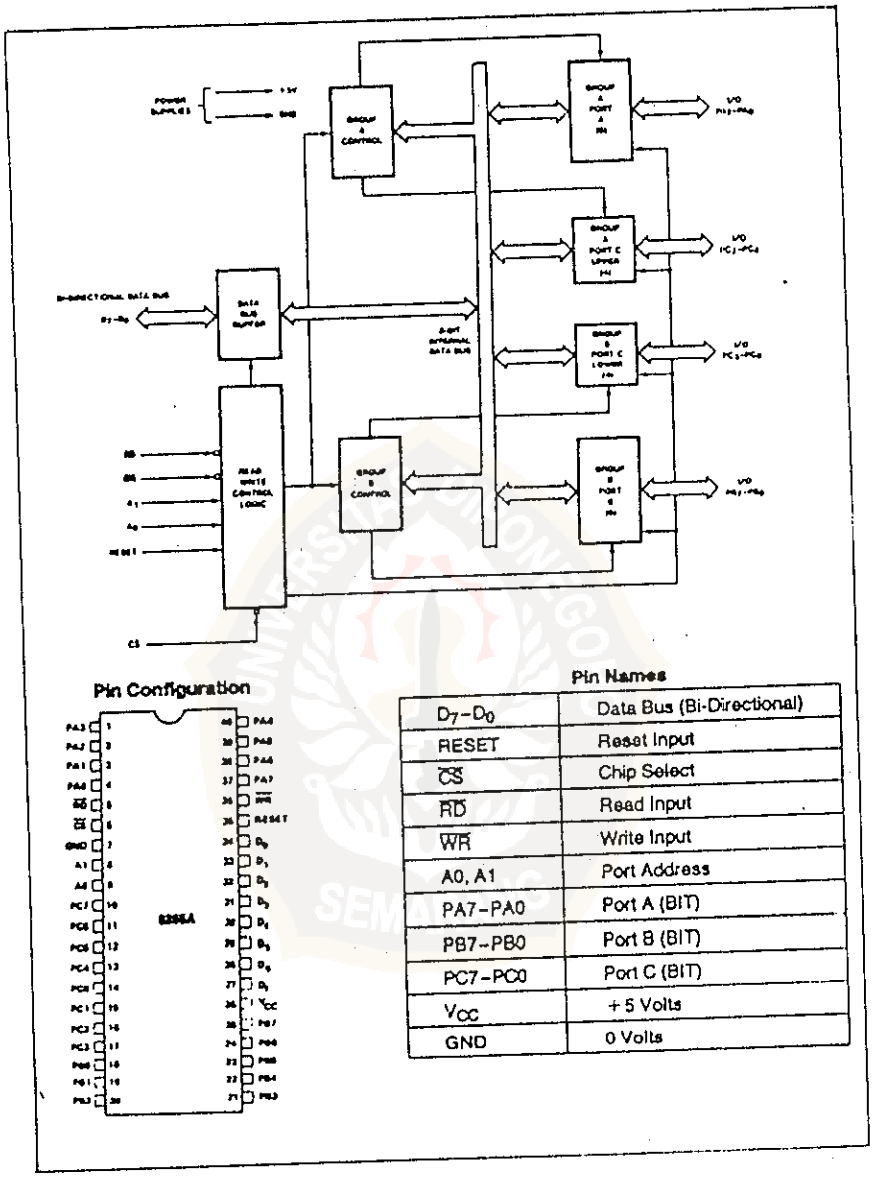
FIGURE 22. Z80 Partial Decoding Interface

An Electronic Watt-Watt-Hour Meter

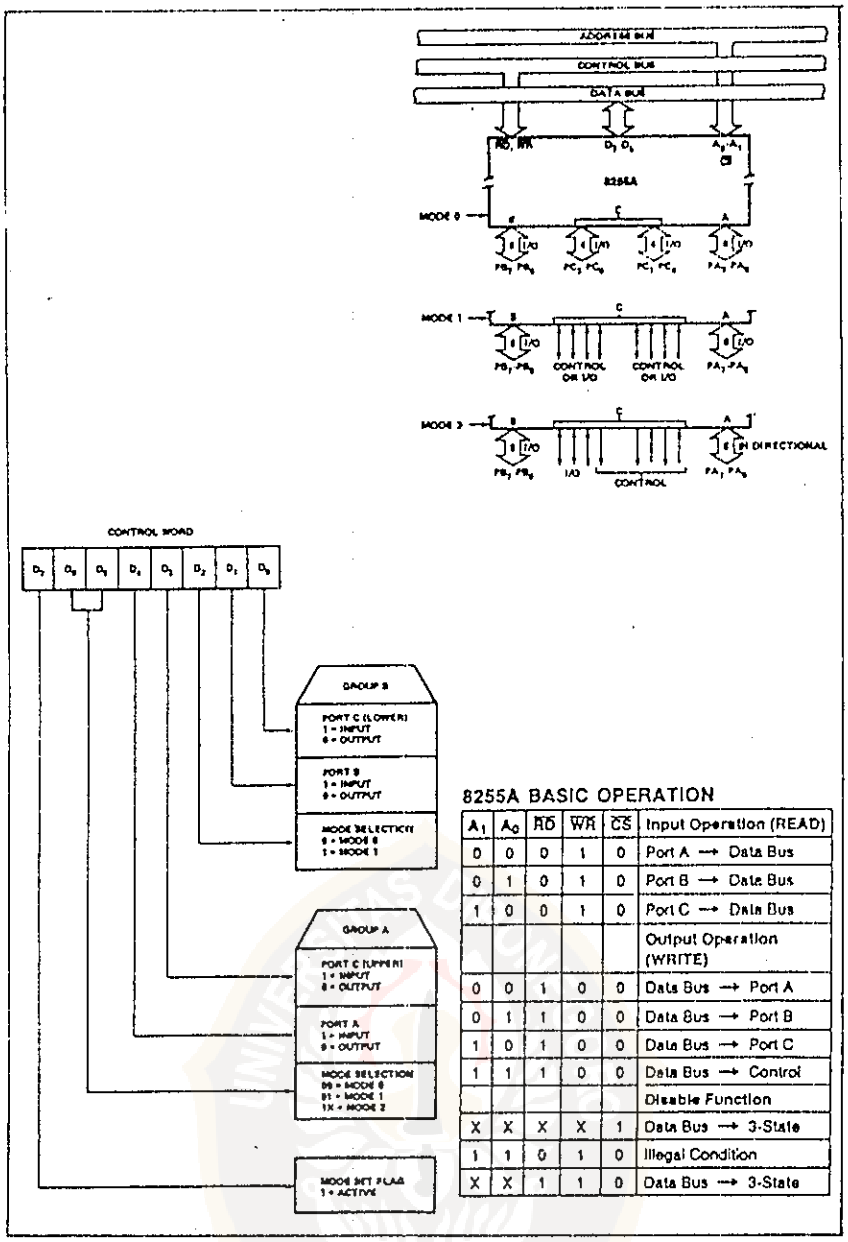
The continued emphasis on energy conservation has forced designers to consider the power consumption and efficiency of their products. While equipment for the industrial market must be designed with attention towards these factors, the consumer area is even more critical. The high cost of electricity has promoted a great deal of interest in the expense of powering various appliances. The watt-watt-hour meter outlined in *Figure 1* allows the designer to easily determine power consumption of any 115V AC powered device. The extremely wide dynamic range of the design allows measurement of loads ranging from 0.1W to 2000W.

Conceptually, the instrument is quite straightforward (*Figure 1*). The device to be monitored is plugged into a standard 110V AC outlet which is mounted on the front panel of the instrument. The AC line voltage across the monitored load is divided down and fed via an op amp to one input of a 4-quadrant analog multiplier. The current through the load is determined by the voltage across a low resistance shunt. Even at 20A the shunt "steals" only 133 mV, eliminating the inaccuracies a high resistance current shunt would contribute. This single shunt is used for all ranges, eliminating the need to switch in high impedance shunts to obtain adequate signal levels on the high sensitivity scales.

This provision is made possible by low uncertainty in the current amplifier, whose output feeds the other multiplier



Gambar 11.2 Blok diagram 8255.



8255A BASIC OPERATION

A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	Input Operation (READ)
0	0	0	1	0	Port A → Data Bus
0	1	0	1	0	Port B → Data Bus
1	0	0	1	0	Port C → Data Bus
					Output Operation (WRITE)
0	0	1	0	0	Data Bus → Port A
0	1	1	0	0	Data Bus → Port B
1	0	1	0	0	Data Bus → Port C
1	1	1	0	0	Data Bus → Control
					Disable Function
X	X	X	X	1	Data Bus → 3-State
1	1	0	1	0	Illegal Condition
X	X	1	1	0	Data Bus → 3-State

Gambar 11.3 Control word.