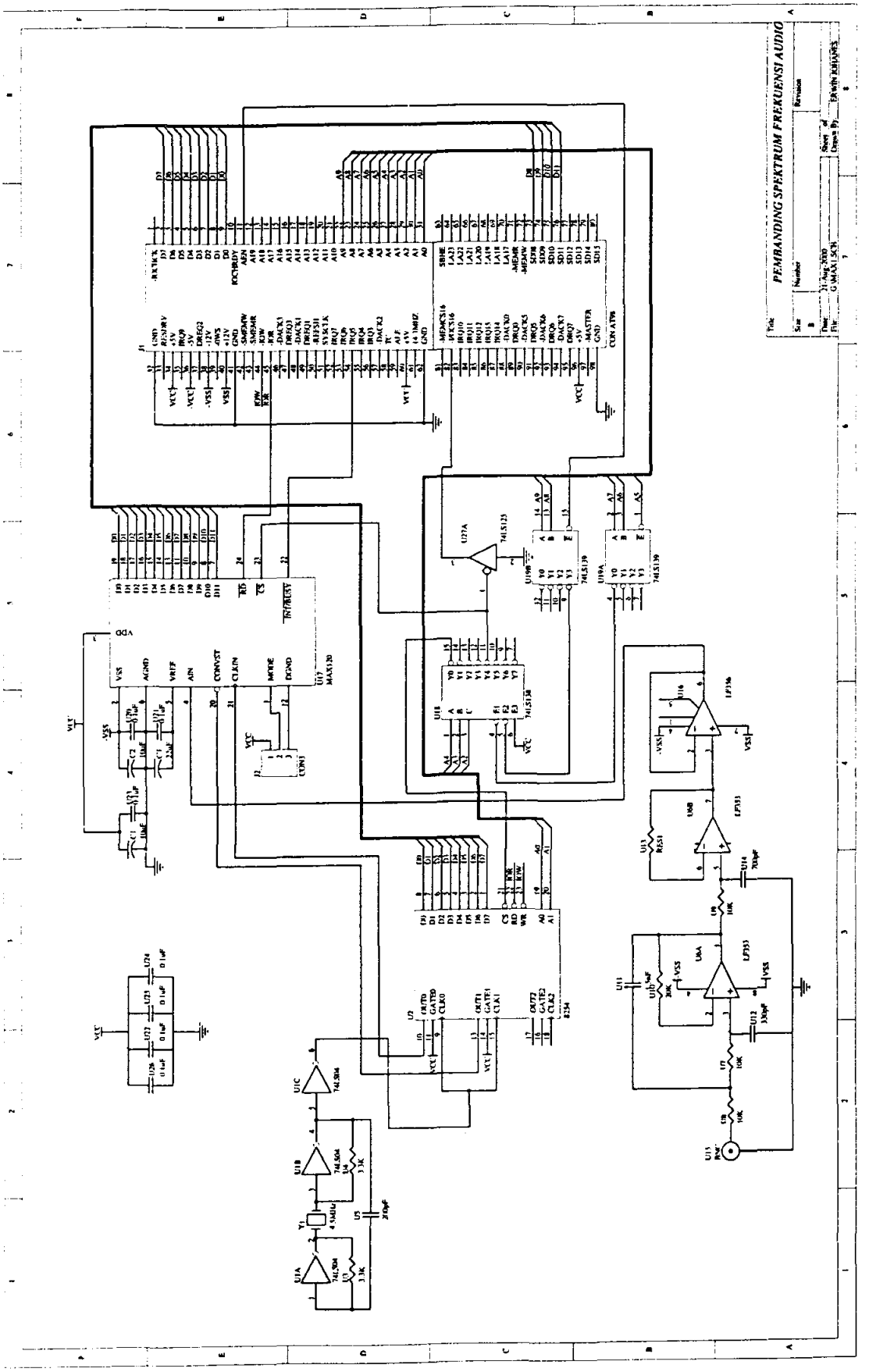
LAMPIRAN

LAMPIRAN 1



LAMPIRAN 2

```
{*************************************************************************
                PROGRAM PEMBANDING SPEKTRUM FREKUENSI AUDIO

                        NAMA   : ERWIN JOHANES BUDHIHARTONO
                        NRP    : 5103096030

        Keterangan : Untuk proses FFT radix 2 dari program ini diambil dari program FFT radix 2
                     dari Bapak Widya Andyaardja ST, MT dan  telah digunakan sebelumnya
                     oleh Tikno Rahardjo ST dalam buku skripsinya yang berjudul
                     Spektrum Analyzer

 *************************************************************************}
uses newdelay,crt,dos,graph;
type complex=record
    re : real;
    im : real;
    end;
    tsamp=array[0..1024] of longint;
    tsamp1=array[0..1024] of complex;
    str7=string[7];

const pcw=$303;         {alamat control word dari PIT}
     strmenu:array[0..4] of string[44]=('1. Lihat Spektrum','2. Rekam Hasil Spektrum','3. Lihat Hasil
     Rekaman', '4. Bandingkan Hasil Rekaman','5. Keluar');   {mencetak menu}
     adc=$304;          {alamat ADC}
     sampel=1024;       {jumlah titik sample yang diambil}
     bit=10;            {jumlah bit dari titik sample}
     Gray10:FillPatternType=($AA,$AA,$AA,$AA,$AA,$AA,$AA,$AA);

var xpos,ypos,grdriver,grmode,Errcode,p,n,d,sampling,max,min : integer;
   old    : fillpatterntype;
   rev    : pointer;
   h      : tsamp;
   ch,tombol,pil : char;
   f,f1           : file of tsamp;     {deklarasi tipe file}
   fout,fout1     : file of longint;   {deklarasi tipe file}
   data           : tsamp1;
   count          : integer;
   datastr        : str7;
   nfile1,nfile2,cth    : string;
   selesai,mulai1,mulai : boolean;
   a,b                  : longint;
   err, err1, a1, temp  : extended;
```

```
   data_max, data_min,data_total : array [0..1024] of integer;
   Vpp, Vpp2, dB, dB2, gain  : array [0..sampel] of real;

procedure Inisialisasi_PIT;
{Procedure ini digunakan untuk menginisialisasi dan mengaktifkan PIT untuk
 bekerja pada mode 3 atau Square Wave Generator demana PIT mempunyai alamat
 300H - 303H}
begin
  asm
  mov dx, 303h          {menginputkan alamat control word dari PIT}
  mov al, 9Fh          {mengaktifkan mode 3 dan alamat counter 2 PIT}
  out dx, al
  mov dx, 302h          {mengaktifkan alamat counter 2 PIT}
  mov al, 3h           {menginputkan pembagian 3 dalam decimal}
  out dx, al

  mov dx, 303h          {menginputkan alamat control word dari PIT}
  mov al, 5Fh          {mengaktifkan mode 3 dan alamat counter 1 PIT}
  out dx, al
  mov dx, 301h          {mengaktifkan alamat counter 1 PIT}
  mov al, 73h          {menginputkan pembagian 73 dalam decimal}
  out dx, al
  end;
end;

procedure writef;
{Procedure ini digunakan untuk merekam data yang telah disampling kedalam
 bentuk file}
begin
  write(f,h);          {menyimpan data dalam bentuk file}
end;

procedure readf(as:byte);
{Procedure ini digunakan untuk membaca data yang telah disimpan}
begin
  read(f,h)           {baca data pertama}
end;

procedure Baca_adc; interrupt;
{Procedure ini digunakan untuk menginterupsi setiap 1024 titik sample yang
 diambil dengan menggunakan IRQ 5}
begin
  h[n]:=portw[adc] and $FFF;       {menghilangkan bit 12 - 15}
  if h[n] and $800=$800 then h[n]:=-2048+(h[n] and $7FF);     {inverse nilai
  negatif}
  inc(n);              {menaikkan fungsi n}
```

```
 if n>=sampel then
 begin
  n:=0;
  sampling:=1;
  port[$21]:=port[$21] or $20;   {disable IRQ 5}
 end;
 port[$20]:=$20;          {akhir interrupt}
end;

procedure Inisialisasi_graph;
{Procedure ini digunakan untuk mengaktifkan mode grafik}
begin
 grDriver:=Detect;        {mendeteksi driver grafik}
 InitGraph(grDriver, grMode,'g:win\bgi');
 Errcode:= GraphResult;
 if graphresult<>grok then halt;
end;

procedure mul(var z:complex;x,y:complex);
{Procedure ini digunakan untuk menghitung perkalian bilangan kompleks}
begin
 z.re:=x.re*y.re-x.im*y.im;
 z.im:=x.re*y.im+x.im*y.re;
end;

procedure sub(var z:complex;x,y:complex);
{Procedure ini digunakan untuk menghitung pengurangan bilangan kompleks}
begin
 z.re:=x.re-y.re;
 z.im:=x.im-y.im;
end;

procedure add(var z:complex;x,y:complex);
{Procedure ini digunakan untuk menghitung pengurangan bilangan kompleks}
begin
 z.re:=x.re+y.re;
 z.im:=x.im+y.im;
end;

function abs1(x:complex):real;
{Function ini digunakan untuk mengitung magnitude bilangan kompleks}
begin
 abs1:=sqrt(sqr(x.re)+sqr(x.im))
end;
```

```
procedure bit_reverse;
{Procedure ini digunakan untuk megubah urutan bilangan kompleks pada file data}
var i,j,k:word;
   t:complex;
begin
 j:=1;
 for i:=1 to sampel-1 do
 begin
   if i<j then
   begin
     t:=data[j];
     data[j]:=data[i];
     data[i]:=t;
   end;
   k:=sampel shr 1;
   while k<j do
   begin
     j:=j-k;
     k:=k shr 1;
   end;
   j:=j+k;
 end;
end;


procedure fft;
{Procedure ini merupakan procedure utama yang digunakan untuk mengubah
 domain waktu ke domain frekuensi dengan menggunakan persamaan FFT pada
 data }
var u,w,t:complex;
   i,j,k,le,le1,p:word;
begin
 for k:= 1 to bit do
 begin
   le:=1 shl k;
   le1:=le shr 1;
   u.re:=1; u.im:=0;
   w.re:=cos(pi/le1);
   w.im:=-sin(pi/le1);
   for j:=1 to le1 do
   begin
     i:=j;
     repeat
       p:=i+le1;
       mul(t,data[p],u);
       sub(data[p],data[i],t);
```

```
      add(data[i],data[i],t);
      inc(i,le);
    until i>=sampel;
    mul(u,u,w);
   end;
  end;
end;


procedure warning;
{Procedure ini digunakan untuk memeunculkan tulisan pada layar komputer}
begin
 inisialisasi_graph;
 setlinestyle(0,0,ThickWidth);    {menggambar garis tepi}
 setcolor(yellow);
 getfillpattern(old);
 bar3d(80,130,580,220,2,true);    {menggambar kotak}
 setcolor(red);
 settextstyle(TriplexFont,0,3);
 outtextxy(90,132,'Tekan ENTER Untuk Memulai');    {mencetak tulisan}
 outtextxy(90,172,'& Tekan Esc Untuk Berhenti');   {mencetak tulisan}
 Readln;
end;


procedure area;
{Procedure ini digunakan untuk menggambar area grafik sinyal}
begin
 SetFillPattern(Gray10,Black);
 Bar(50,60,600,360);     {menggambar persegi}
 setcolor(11);
 line(50,60,50,360);     {menggambar garis vertikal skala}
 line(50,360,441,360);   {menggambar garis horisontal skala}
 setcolor(7);
 for p:=0 to 27 do
 line(50+(p*17),362,50+(p*17),358);     {mencetak skala horizontal}
 for p:=0 to 10 do
 line(48,60+(p*30),52,60+(p*30));
 for p:=0 to 27 do
 line(50+(p*17),358,50+(p*17),60);      {mencetak garis horizontal}
 for p:=0 to 10 do
 line(50,60+(p*30),560,60+(p*30));
 setcolor(9);
 outtextxy(140,6,'GRAFIK SPEKTRUM FREKUENSI AUDIO');
 setcolor(15);
 outtextxy(40,370,' 0'); outtextxy(73,370,' 2');    {mencetak agka skala horizontal}
```

```
  outtextxy(107,370,' 4'); outtextxy(142,370,' 6');
  outtextxy(174,370,' 8'); outtextxy(213,370,'10');
  outtextxy(248,370,'12 '); outtextxy(282,370,'14 ');
  outtextxy(316,370,'16 '); outtextxy(349,370,'18 ');
  outtextxy(383,370,'20'); outtextxy(418,370,'22');
  outtextxy(452,370,'24'); outtextxy(483,370,'KHz');
  setcolor(10);
end;

procedure rumus_gain_high; Near;
{Procedure ini digunakan untuk mencari nilai V puncak maks}
begin
  if h[p] > data_max[max] then
  begin
    data_max[max] := h[p];
  end;
  Vpp[p] := (data_max[max]+0.00001)/204.8;
  dB[p] := 20*(ln((Vpp[p])/(2*sqrt(2))/0.775)/ln(10));
end;

procedure rumus_gain_low; Near;
{Procedure ini digunakan untuk mencari nilai V puncak min}
begin
  if h[p] < data_min[min] then
  begin
    data_min[min] := h[p];
  end;
  Vpp2[p] := ((-1)*data_min[min]+0.00001)/204.8;
  dB2[p] := 20*(ln((Vpp2[p])/(2*sqrt(2))/0.775)/ln(10));
end;

procedure gain_dB; Near;
{Procedure ini digunakan untuk mencari nilai gain}
begin
  data_total[p] := (data_max[max]-data_min[min]);
  Vpp2[p] := (Vpp[p]+Vpp2[p])/2;
  gain[p] := (dB[p]+dB2[p])/2;
  if gain[p] <= -16.3943 then gain[p] := -16.3943;
end;

procedure tulis_gain; Near;
{Procedure ini digunakan untuk menulis gain}
begin
  str(Vpp2[p]:1:4,datastr);
  outtextxy(546,386,datastr);
  str(gain[p]:4:4,datastr);
```

```
 outtextxy(546,100,datastr);
end;

procedure viewrekaman;
{Procedure ini digunakan untuk menampilkan kembali data yang telah direkam}
var nfile:string;
begin
 inisialisasi_graph;   {menginisialisasi grafik}
 inisialisasi_PIT;     {menginisialisasi PIT}
 setlinestyle(0,0,ThickWidth);
 setcolor(YELLOW);
 getfillpattern(old);
 bar3d(80,130,580,220,2,true);
 setcolor(red);
 settextstyle(TriplexFont,0,3);
 outtextxy(90,152,'Masukkan nama file : ');   {menunggu input nama file}
 gotoxy(45,11);
 readln(nfile);       {membuka file dengan nama yang diinputkan}
 assign(f,nfile);
 reset(f);
 warning;
 inisialisasi_graph;  {menginisialisasi grafik}
 setcolor(14);
 setcolor(blue);
 rectangle(1,39,639,439);
 setcolor(9);
 outtextxy(533,230,'TIME DOMAIN');
 setviewport(2,40,512,420,true);     {mencetak area sinyal time domain}
 area;
 n:=0;
 sampling:=0;
 count:=0;
 repeat
  readf(1);
  for p:=0 to sampel do
    begin
      data[p].re:=h[p]/175*(1-cos(2*pi*(p shl 1)/sampel)); {menghitung data real}
      data[p].im:=0;       {menghitung data imajiner}
    end;
  bit_reverse;           {membalik bit data}
  fft;                   {menghitung nilai FFT}
  p:=0;
  repeat
   inc(p);
     begin
       begin
```

```
        setviewport(2,40,512,420,false);  {menutup area sinyal sebelumnya}
        setcolor(7);
        rectangle(517,59,631,181);
        setviewport(520,100,630,220,true); {membatasi area sinyal baru}
        setcolor(10);
        moveto(p,60+(h[p-1] div 35));
        lineto(p,60+(h[p] div 35));    {menggambar sinyal time domain}
        setviewport(520,100,630,220,false);  {menutup area sinyal sebelumnya}
        setviewport(2,40,512,420,true);  {membatasi area sinyal baru}
       end;
       if p<500 shl 1 then
       begin
        setcolor(12);
        moveto(50+p,360-round(50*abs1(data[p-1])/sampel));
        lineto(50+p,360-round(50*abs1(data[p])/sampel)); {menggambar sinyal frekuensi domain}
       end;
      end;
    if p>=500 then
      begin
       for p:=0 to sampel do
         begin
          data[p].re:=(h[p]/100)*(1-cos(2*pi*(p)/sampel));  {menghitung nilai data real}
          data[p].im:=0;   {menghitung nilai data imajiner}
         end;
       bit_reverse;    {membalik nilai bit data}
       fft;          {menghitung data dengan proses FFT}
       p:=0;
      end;
  until p=0;
  inc(count);          {menaikkan pewaktu}
  str(count,cth);
  outtextxy(1,1,'Count ke :'+cth);   {mencetak nilai pewaktu}
  clearviewport;
  setviewport(2,40,512,420,false);    {menutup area sinyal sebelumnya}
  setviewport(520,100,630,220,true);  {membatasi area sinyal baru}
  clearviewport;                  {membersihkan area}
  setviewport(520,100,630,220,false); {menutup area sinyal sebelumnya}
  setviewport(2,40,512,420,true);     {membatasi area sinyal baru}
  area;               {menggambar area sinyal }
 until eof(f);
 close(f);             {menutup file}
 closegraph;           {menutup mode grafik}
end;
```

```
procedure rekam;
{Procedure ini digunakan untuk merekam data yang telah diambil sebelumnya}
var nfile:string;
begin
 inisialisasi_graph;     {menginisialisasi grafik}
 inisialisasi_PIT;       {menginisialisasi PIT}
 setlinestyle(0,0,ThickWidth);
 setcolor(yellow);
 getfillpattern(old);
 bar3d(80,130,580,220,2,true);    {menggambar kotak}
 setcolor(red);
 settextstyle(TriplexFont,0,3);
 outtextxy(90,152,'Masukkan nama file : ');     {menunggu input nama file}
 gotoxy(45,11);
 readln(nfile);                  {membuka file dengan nama yang telah diinputkan}
 assign(f,nfile);
 rewrite(f);                    {menuliskan data kedalam file}
 warning;
 inisialisasi_graph;             {menginisialisasikan grafik}
 setcolor(14);
 setcolor(blue);
 rectangle(1,39,639,439);        {menggambar kotak}
 setcolor(9);
 outtextxy(533,230,'TIME DOMAIN');
 setviewport(2,40,512,420,true);
 area;           {menggambar area sinyal}
 tombol:=#26;  n:=0;
 sampling:=0;
 count:=0;
 getintvec($0D,rev);          {simpan alamat IRQ5}
 setintvec($0D,@Baca_adc);    {set alamat IRQ5}
 port[$21]:=port[$21]and $DF;
 repeat
 until sampling=1;
 writef;
 for p:=0 to sampel do
   begin
    data[p].re:=h[p]/175*(1-cos(2*pi*(p shl 1)/sampel));  {menghitung data real}
    data[p].im:=0;      {menghitung data imajiner}
   end;
 bit_reverse;          {membalik nilai bit data}
 fft;                {menghitung nilai FFT}
 p:=0;
 repeat
  repeat
   inc(p);
```

```
  setviewport(2,40,512,420,false);    {menutup area sinyal sebelumnya}
  setcolor(7);
  rectangle(517,59,631,181);
  setviewport(520,100,630,220,true);  {membuka area sinyal baru}
  setcolor(10);
  moveto(p,60+(h[p-1] div 35));
  lineto(p,60+(h[p] div 35));         {menggambar data time domain}
  setviewport(520,100,630,220,false); {menutup area sinyal sebelumnya}
  setviewport(2,40,512,420,true);     {membuka area sinyal baru}
  if sampling=1 then
    begin
     if p<500 shl 1 then
       begin
        setcolor(12);
        moveto(50+p,360-round(50*abs1(data[p-1])/sampel));
        lineto(50+p,360-round(50*abs1(data[p])/sampel));  {menggambar sinyal domain
frekuensi}
       end;
    end;
  if p>=500 then
    begin
     port[$21]:=port[$21] and $DF;    {enable IRQ5}
       repeat
       until sampling=1;
     for p:=0 to sampel do
       begin
        data[p].re:=(h[p]/256)*(1-cos(2*pi*(p)/sampel)); {menghitung data real}
        data[p].im:=0;  {menghitung data imajiner}
       end;
     bit_reverse;  {membalik nilai bit data}
     fft;          {menghitung nilai FFT}
     clearviewport;   {membersihkan area}
     setviewport(2,40,512,420,false);   {menutup area sinyal sebelumnya}
     setviewport(520,100,630,220,true); {membatasi area sinyal baru}
     clearviewport;   {membersihkan area}
     setviewport(520,100,630,220,false); {menutup area sinyal sebelumnya}
     setviewport(2,40,512,420,true);   {membatasi area sinyal baru}
     area;  {menggambar area sinyal}
     p:=0;
    end;
  if keypressed then tombol:=readkey; {menunggu input tombol}
until (tombol=#27) or (p=0);
writef;       {menulis file}
inc(count);
str(count,cth);
outtextxy(1,1,'Count : '+cth);  {mencetak nilai pewaktu}
```

```
 until tombol=#27;
 setintvec($0D,rev);      {set alamat IRQ5}
 port[$21]:=port[$21] or $20;  {enable IRQ5}
 close(f);           {menutup file}
end;

procedure drawmenu;
{Procedure ini digunakan untuk menggambar menu}
var i:byte;
begin
 inisialisasi_graph;    {menginisialisasi grafik}
 setlinestyle(0,0,ThickWidth);
 setcolor(yellow);
 getfillpattern(old);
 bar3d(80,30,580,80,2,true);  {menggambar kotak}
 setcolor(red);
 settextstyle(TriplexFont,0,3);
 outtextxy(90,32,'PEMBANDING SPEKTRUM FREKUENSI AUDIO');
 xpos:=140;ypos:=100;        {menset posisi}
 for i:=0 to 4 do
 begin
  setfillpattern(old,9);setcolor(Green);
  Bar3d(xpos,ypos+i*40,xpos+370,ypos+(i+1)*40,3,TRUE);
  setfillpattern(old,8);setcolor(white);
  outtextxy(xpos+20,ypos+5+i*40,strmenu[i]);
 end;
 settextstyle(TriplexFont,0,3);
 outtextxy(140,322,'Tekan Nomor Sesuai Menu : (_)');
end;

Procedure bandingkan;
{Procedure ini digunakan untuk membandingkan dua buah sample yang telah disimpan
 dalam dua file yang berbeda dan kemudian diubah dalam nbentuk FFT untuk
 diperbandingkan satu persatu setelah nilai nol yang ada di permulaan sinyal
 dieliminasi terlebih dahulu}
var i,j : integer;
   nfile:string;
   nfile1:string;
begin
 clrscr;
 clearviewport;
 inisialisasi_graph;    {menginisialisasi grafik}
 mulai:=false;
 setcolor(yellow);
 getfillpattern(old);
 bar3d(80,130,580,220,2,true);
```

```
setcolor(red);
settextstyle(TriplexFont,0,3);
outtextxy(90,152,'Masukkan nama file : '); {menunggu input nama file pertama}
gotoxy(45,11);
readln(nfile);
setcolor(yellow);
getfillpattern(old);
bar3d(80,130,580,220,2,true);
setcolor(red);
settextstyle(TriplexFont,0,3);
outtextxy(90,152,'Masukkan nama file : '); {menunggu input nama file kedua}
gotoxy(45,11);
readln(nfile1);
warning;
inisialisasi_graph;
setcolor(blue);
rectangle(1,39,639,439);
setcolor(9);
outtextxy(533,230,'TIME DOMAIN');
setviewport(2,40,512,420,true); {membatasi area sinyal}
area;                    {mencetak area sinyal}
for i:=1 to 2 do              {membuka file secara berurutan}
  begin
    if i=1 then
      begin
        assign(f,nfile);       {membuka file}
        reset(f);
        assign(fout,'fft.dat'); {menyimpannya dalam bentuk FFT}
        rewrite(fout);
      end
    else
      begin
        assign(f,nfile1);      {membuka file kedua}
        reset(f);
        assign(fout,'fft1.dat'); {menyimpannya dalam bentuk FFT}
        rewrite(fout);
      end;
    repeat
      read(f, h);
      for p:=0 to sampel do
        begin
          data[p].re:=h[p]/175*(1-cos(2*pi*(p shl 1)/sampel)); {menghitung nilai real data}
          data[p].im:=0; {menghitung nilai imajiner data}
        end;
      bit_reverse; {membalik nilai bit data}
      fft;       {memproses data dengan FFT}
```

```
    for j:=0 to sampel do
      begin
       a:=round(abs1(data[j]));
       write(fout, a);       {menyimpan bentuk absolut data}
      end;
    area;
    p:=0;
    setviewport(2,40,512,420,false); {menutup area sinyal sebelumnya}
    setcolor(7);
    rectangle(517,59,631,181);
    setviewport(520,100,630,220,true); {membatasi area sinyal baru}
     for p:=1 to 500 do
      begin
       if i=1 then setcolor(10) else setcolor(red);
       if p>1 then
       lineto(p,60+(h[p] div 35));    {menggambar sinyal time domain pertama}
       moveto(p,60+(h[p-1] div 35));
      end;
    delay(50);
    clearviewport;
    setcolor(3);
    setviewport(520,100,630,220,false);   {menutup area sinyal sebelumnya}
    setviewport(2,40,512,420,true);       {membatasi area sinyal baru}
    p:=0;
    for p:=1 to 500 do
      begin
       if i=1 then setcolor(10) else setcolor(red);
         if p>1 then
           lineto(50+p,360-round(50*abs1(data[p])/sampel));    {menggambar sinyal frekuensi
domain}
           moveto(50+p,360-round(50*abs1(data[p-1])/sampel));
      end;
    delay(50);
    clearviewport;
  until eof(f);
  close(fout);    {menutup file}
  close(f);
end;
assign(fout, 'fft.dat');
reset(fout);
assign(fout1, 'fft1.dat');
reset(fout1);
j:=0;
repeat
  read(fout, a);
   if a>4 then mulai:=true;    {logika untuk mengeliminasi nilai 0 data}
```

```
    until mulai;
    mulai:=false;
    repeat
      read(fout1, b);
        if b>4 then mulai:=true;
    until mulai;
    err:=0;
    while not eof(fout) do
      begin
        inc(j);
        read(fout, a);
        read(fout1, b);
        err:=err+(a-b);   {proses menghitung besar simpangan}
        a1:=a1+a;
      end;
  close(fout1);
  close(fout);
  err:=(err/j);
  a1:=a1/j;
  err:=abs((err/a1))*100;  {proses menghitung besar simpangan}
  if err >= 100 then       {membatasi simpangan sampai 100 %}
  err:=100;
  setlinestyle(0,0,ThickWidth);
  inisialisasi_graph;
   setcolor(yellow);
   getfillpattern(old);
   bar3d(80,100,500,200,2,true);
   setcolor(red);
   settextstyle(TriplexFont,0,3);
   gotoxy(45,11);
   str(err:15,cth);
   outtextxy(120,110,'Hasil Standard Perbandingan : '); {mencetak tulisan}
   outtextxy(120,130,cth+' %');   {mencetak hasil perbandingan}
   readln;
end;

procedure Lihat_Spektrum;
{Procedure ini digunakan untuk melihat spektrum frekuensi dari input analog}
begin
  inisialisasi_graph;   {menginisialisasi grafik}
  inisialisasi_PIT;     {menginisialisasi PIT}
  warning;
  inisialisasi_graph;
  setcolor(blue);
  rectangle(1,39,639,439);
  setcolor(9);
```

```
outtextxy(533,230,'TIME DOMAIN');
setcolor(3);
outtextxy(516,300,'Gain =      dB');
setviewport(2,40,512,420,true);  {membatasi area sinyal}
area;              {mencetak area sinyal}
tombol:=#26;
n:=0;
sampling:=0;
count:=0;
getintvec($0D,rev);   {simpan alamat IRQ5}
setintvec($0D,@Baca_adc);  {set alamat IRQ5}
port[$21]:=port[$21]and $DF;   {enable IRQ5}
repeat
until sampling=1;
for p:=0 to sampel do
  begin
    data[p].re:=h[p]/158*(1-cos(2*pi*(p shl 1)/sampel)); {menghitung nilai real data}
    data[p].im:=0;  {menghitung nilai imajiner data}
  end;
bit_reverse;  {membalik nilai bit data}
fft;         {menghitung nilai FFT data}
p:=0;
repeat
 repeat
   rumus_gain_high;
   rumus_gain_low;
   gain_dB;
   inc(p);
   setviewport(2,40,512,420,false); {menghapus area sinyal sebelumnya}
   setcolor(7);
   rectangle(517,59,631,181);
   setviewport(520,100,630,220,true); {membatasi area sinyal baru}
   setcolor(10);
   moveto(p,60+(h[p-1] div 35));
   lineto(p,60+(h[p] div 35));    {mencetak sinyal time domain}
   setviewport(520,100,630,220,false); {menghapus area sebelumnya}
   setviewport(2,40,512,420,true);  {membatasi area sinyal baru}
   if sampling=1 then
     begin


      if p<500 shl 1 then
        begin
          setcolor(12);
          moveto(50+p,360-round(50*abs1(data[p-1])/sampel));
          lineto(50+p,360-round(50*abs1(data[p])/sampel)); {mencetak sinyal frekuensi domain }
```

```
          end;
        end;
      if p>=500 then
        begin
         port[$21]:=port[$21] and $DF;  {enable IRQ5}
         repeat
          rumus_gain_high;
          rumus_gain_low;
          gain_dB;
          setcolor(10);
          tulis_gain;
         until sampling = 1;
         data_max[max] := 0;
         data_min[min] := 0;

         for p:=0 to sampel do
           begin
            data[p].re:=(h[p]/158)*(1-cos(2*pi*(p)/sampel)); {menghitung nilai real data}
            data[p].im:=0;   {menghitung nilai imajiner data}
           end;
         bit_reverse;  {membalik nilai bit data}
         fft;          {menghitung nilai FFT data}
         clearviewport;   {membersihkan area sinyal}
         setviewport(2,40,512,420,false);    {menghilangkan area sebelumnya}
         setviewport(520,100,630,220,true);  {membatasi area sinyal baru}
         clearviewport;   {menghapus area sinyal}
         setviewport(520,100,630,220,false); {menghilangkan area sinyal sebelumnya}
         setviewport(2,40,512,420,true);  {membatasi area sinyal baru}
         area;   {menggambar area sinyal}
         p:=0;
        end;
      if keypressed then tombol:=readkey;
    until (tombol=#27) or (p=0);
    inc(count);
    str(count,cth);
  until tombol=#27;
  setintvec($0D,rev);        {set alamat IRQ5}
  port[$21]:=port[$21] or $20;  {enable IRQ5}
end;

procedure menu;
begin
  inisialisasi_graph;        {menginisialisasi grafik}
  repeat
    Drawmenu;                {menggambar menu}
    pil:=readkey;
```

```
        case pil of
      '1':Lihat_Spektrum;     {melihat spektrum}
        '2':Rekam;            {merekam spektrum}
        '3':ViewRekaman;        {melihat rekaman}
        '4':Bandingkan;         {membandingkan rekaman}
        end;
 until pil='5';
end;

begin
 inisialisasi_graph;   {menginisialisasikan grafik}
 menu;                 {memanggil menu}
end.
```

19-0030; Rev 0; 9/92

EVALUATION KIT AVAILABLE

MAXIM

500ksps, 12-Bit ADCs with Track/Hold and Reference

MAX120/MAX122

General Description

The MAX120 and MAX122 complete, BiCMOS, sampling 12-bit analog-to-digital converters (ADCs) combine an on-chip track/hold (T/H) and a low-drift voltage reference with fast conversion speeds and low power consumption. The T/H's 350ns acquisition time combined with the MAX120's 1.6µs conversion time results in throughput rates as high as 500k samples per second (ksps). Throughput rates of 333ksps are possible with the 2.6µs conversion time of the MAX122.

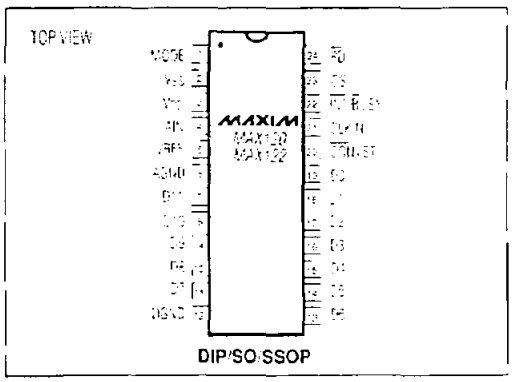
The MAX120/MAX122 accept analog input voltages from -5V to +5V. The only external components needed are decoupling capacitors for the power-supply and reference voltages. The MAX120 operates with clocks in the 0.1MHz to 8MHz frequency range. The MAX122 accepts 0.1MHz to 5MHz clock frequencies.

The MAX120/MAX122 employ a standard microprocessor (µP) interface. Three-state data outputs are configured to operate with 12-bit data buses. Data-access and bus-release timing specifications are compatible with most popular µPs without resorting to wait states. In addition, the MAX120/MAX122 can interface directly to a first in, first out (FIFO) buffer, virtually eliminating µP interrupt overhead. All logic inputs and outputs are TTL/CMOS compatible. For applications requiring a serial interface, refer to the new MAX121.

Applications

Digital-Signal Processing
Audio and Telecom Processing
Speech Recognition and Synthesis
High-Speed Data Acquisition
Spectrum Analysis
Data Logging Systems

Pin Configuration



Features

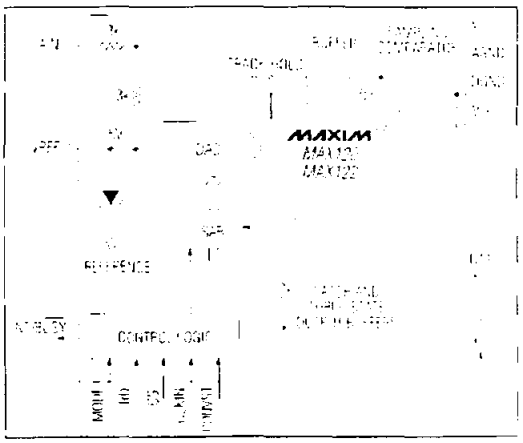
- ♦ 12-Bit Resolution
- ♦ No Missing Codes Over Temperature
- ♦ 20ppm/°C -5V Internal Reference
- ♦ 1.6µs Conversion Time/500ksps Throughput (MAX120)
- ♦ 2.6µs Conversion Time/333ksps Throughput (MAX122)
- ♦ Low Noise and Distortion:
  70 dB Min SINAD;
  -77 dB Max THD (MAX122)
- ♦ Low Power Dissipation: 210mW
- ♦ Separate Track/Hold Control Input
- ♦ Continuous-Conversion Mode Available
- ♦ ±5V Input Range, Overvoltage Tolerant to ±15V
- ♦ 24-Pin Narrow DIP, Wide SO and SSOP Packages

Ordering Information

PART	TEMP. RANGE	PACKAGE	INL (LSBs)
MAX120CNG	0°C to +70°C	24 Narrow Plastic DIP	±1
MAX120CWG	0°C to +70°C	24 Wide SO	±1
MAX120CAG	0°C to +70°C	24 SSOP	±1
MAX120C/D	0°C to +70°C	Dice*	±1
MAX120ENG	-40°C to +85°C	24 Narrow Plastic DIP	±1
MAX120EWG	-40°C to +85°C	24 Wide SO	±1

Ordering Information continued on last page.
**Contact factory for dice specifications.*

Functional Diagram



500ksps, 12-Bit ADCs with Track/Hold and Reference

ABSOLUTE MAXIMUM RATINGS

VDD to DGND ..... -0.3V to +6V
VSS to DGND ..... +0.3V to -7V
AIN to AGND ..... ±15V
AGND to DGND ..... ±0.3V
Digital Inputs/Outputs to DGND ..... -0.3V to (VDD + 0.3V)
Continuous Power Dissipation (TA = +70°C)
Narrow Plastic DIP (derate 13.33mW/°C above +70°C) 1067mW
SO (derate 11.76mW/°C above +70°C) 941mW
SSOP (derate 8.00mW/°C above +70°C) 640mW
Narrow CERDIP (derate 12.50W/°C above +70°C) .. 1000mW

Operating Temperature Ranges:
MAX12_C_ ..... 0°C to +70°C
MAX12_E_ ..... -40°C to +85°C
MAX12_MRG ..... -55°C to +125°C
Storage Temperature Range ..... -65°C to +160°C
Lead Temperature (soldering, 10 sec) ..... +300°C

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS

(VDD = +4.75V to +5.25V, VSS = -10.8V to -15.75V, fCLK = 8MHz for MAX120 and 5MHz for MAX122, TA = TMIN to TMAX, unless otherwise noted.)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
ACCURACY							
Resolution	RES			12			Bits
Differential Nonlinearity (Note 1)	DNL	12-bit no missing codes over temp. range	MAX122AC/AE			+3/4	LSB
			MAX120C/E, MAX122BC/BE/BM			±1	
		11-bit no missing codes over temp. range	MAX120M			±2	
Integral Nonlinearity (Note 1)	INL		MAX122AC/AE			±3/4	LSB
			MAX120C/E, MAX122BC/BE/BM			±1	
			MAX120M			±2	
Bipolar Zero Error (Note 1)		Code 00...00 to 00...01 transition, near AIN = 0V				±3	LSB
		Temperature drift			±0.005		LSB/°C
Full-Scale Error (Notes 1, 2)		Including reference, adjusted for bipolar zero error, TA = +25°C				±8	LSB
Full-Scale Temperature Drift		Excluding reference			±1		ppm/°C
Power-Supply Rejection Ratio (Change in FS, Note 3)	PSRR	VDD only, 5V ±5%			±1/4	±3/4	LSB
		VSS only, -12V ±10%			±1/4	±1	
		VSS only, -15V ±5%			±1/4	±1	
ANALOG INPUT							
Input Range				-5		5	V
Input Current		AIN = +5V (approximately 6kΩ to REF)				2.5	mA
Input Capacitance (Note 4)						10	pF
Full-Power Input Bandwidth					1.5		MHz
REFERENCE							
Output Voltage		No external load, AIN = 5V, TA = +25°C		4.92		4.98	V
External Load Regulation		0mA < ISINK < 5mA, AIN = 0V				5	mV
Temperature Drift (Note 5)		MAX12_C/E				±25	ppm/°C
		MAX12_M				±30	

ELECTRICAL CHARACTERISTICS (continued)

(VDD = +4.75V to +5.25V, VSS = -10.8V to -15.75V, fCLK = 8MHz for MAX120 and 5MHz for MAX122, TA = TMIN to TMAX, unless otherwise noted.)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
DYNAMIC PERFORMANCE (MAX120: fS = 500kHz, AIN = ±5Vp-p, 100kHz; MAX122: fS = 333kHz, AIN = ±5Vp-p, 50kHz							
Signal-to-Noise Plus Distortion	SINAD	TA = +25°C	MAX120, MAX122	70	72		dB
			MAX122AC/AE	70			
			MAX122BC/BE/BM	69			
Total Harmonic Distortion (First Five Harmonics)	THD	TA = +25°C	MAX120		-82	-77	dB
			MAX122		-85	-78	
			MAX122AC/AE			-77	
			MAX122BC/BE/BM			-75	
Spurious-Free Dynamic Range	SFDR	TA = +25°C	MAX120	77	82		dB
			MAX122	78	85		
			MAX122AC/AE	77			
			MAX122BC/BE/BM	75			
CONVERSION TIME							
Synchronous	tCONV	13tCLK	MAX120			1.63	µs
			MAX122			2.60	
Clock Frequency	fCLK		MAX120	0.1		8	MHz
			MAX122	0.1		5	
DIGITAL INPUTS (CLKIN, CONVST, RD, CS)							
Input High Voltage	VIH			2.4			V
Input Low Voltage	VIL					0.8	V
Input Capacitance (Note 4)						10	pF
Input Current		VIN = 0V or VDD				±5	µA
DIGITAL OUTPUTS (INT/BUSY, D11-D0)							
Output Low Voltage	VOL	ISINK = 1.6mA				0.4	V
Output High Voltage	VOH	ISOURCE = 1mA		VDD - 0.5			V
Leakage Current	ILKG	VIN = 0V or VDD, D11-D0				±5	µA
Output Capacitance (Note 4)						10	pF
POWER REQUIREMENTS							
Positive Supply Voltage	VDD	Guaranteed by supply rejection test		4.75		5.25	V
Negative Supply Voltage	VSS	Guaranteed by supply rejection test		-10.80		-15.75	V
Positive Supply Current (Note 6)	IDD	VDD = 5.25V, VSS = -15.75V, AIN = 0V			9	15	mA
Negative Supply Current (Note 6)	ISS	VDD = 5.25V, VSS = -15.75V, AIN = 0V			14	20	mA
Power Dissipation (Note 6)		VDD = 5V, VSS = -12V, AIN = 0V			210	315	mW

Note 1: These tests are performed at VDD = 5V, VSS = -15V. Operation over supply is guaranteed by supply rejection tests.
Note 2: Ideal full-scale transition is at +5V - 3/2LSB = +4.9963V, adjusted for offset error.
Note 3: Supply rejection defined as change in full-scale transition voltage with the specified change in supply voltage = (FS at nominal supply) - (FS at nominal supply ± tolerance), expressed in LSBs.
Note 4: For design guidance only, not tested.
Note 5: Temperature drift is defined as the change in output voltage from +25°C to TMIN or TMAX. It is calculated as TC = (ΔVREF/VREF)/(ΔT).
Note 6: CS = RD = CONVST = 0V, MODE = 5V.

500ksps, 12-Bit ADCs with Track/Hold and Reference

TIMING CHARACTERISTICS

(VDD = +5V, VSS = -12V to -15V, 100% tested, TA = TMIN to TMAX, unless otherwise noted.) (Note 7)

PARAMETER	SYMBOL	CONDITIONS	TA = +25°C			MAX12_C/E			MAX12_M			UNITS
			MIN	TYP	MAX	MIN	TYP	MAX	MIN	TYP	MAX	
$\overline{CS}$ to $\overline{RD}$ Setup Time	tCS		0			0			0			ns
$\overline{CS}$ to $\overline{RD}$ Hold Time	tCH		0			0			0			ns
$\overline{CONVST}$ Pulse Width	tCW		30			30			30			ns
$\overline{RD}$ Pulse Width	tRD		tDA			tDA			tDA			ns
Data-Access Time	tDA	CL = 100pF		40	75			100			120	ns
Bus-Relinquish Time	tDH			30	50			65			80	ns
$\overline{RD}$ or $\overline{CONVST}$ to $\overline{BUSY}$	tBD	CL = 50pF		30	75			100			120	ns
CLKIN to $\overline{BUSY}$ or $\overline{INT}$	tBI	CL = 50pF		70	110			150			180	ns
CLKIN to $\overline{BUSY}$ Low	tBZ	In mode 5		45	90			120			150	ns
$\overline{RD}$ to $\overline{INT}$ High	tRI	CL = 50pF		30	50			75			90	ns
$\overline{BUSY}$ or $\overline{INT}$ to Data Valid	tBD	CL (Data) = 100pF, CL ($\overline{INT}$, $\overline{BUSY}$) = 50pF			20			30			35	ns
Acquisition Time (Note 8)	tAQ		350			350			400			ns
Aperture Delay (Note 8)	tAP			10								ns
Aperture Jitter (Note 8)				30								ps

Note 7: Control inputs specified with tr = tf = 5ns (10% to 90% of +5V) and timed from a 1.6V voltage level. Output delays are measured to +0.8V if going low, or -2.4V if going high. For bus-relinquish time, a change of 0.5V is measured. See figures 1 and 2 for load circuits.

Note 8: For design guidance only, not tested.

Pin Description

PIN	NAME	FUNCTION
1	MODE	Mode Input - hardwire to set operational mode VSS: Single conversion, $\overline{INT}$ Output OPEN: Single conversion, $\overline{BUSY}$ Output DGND: Continuous conversions, $\overline{BUSY}$ Output
2	VSS	Negative Power Supply: -12V or -15V
3	VDD	Positive Power Supply: +5V
4	AIN	Sampling Analog Input: ±5V bipolar input range
5	VREF	+5V Reference Output - bypass to AGND with 22µF II 0.1µF

Pin Description (continued)

PIN	NAME	FUNCTION
6	AGND	Analog Ground
7-11, 13-19	D11-D0	Three-State Data Outputs D11 (MSB) to D0 (LSB)
12	DGND	Digital Ground
20	CONVST	Convert Start Input. Initiates conversions on its falling edge.
21	CLKIN	Clock Input. Drive with TTL-compatible clock from 0.1MHz to 8MHz (MAX120), 0.1MHz to 5MHz (MAX122).
22	INT/BUSY	Interrupt or Busy Output indicates converter status. If MODE is connected to VDD, configure for an INT output. If MODE is open or connected to DGND, configure for a BUSY output. See operational diagrams.
23	$\overline{CS}$	Chip Select Input - active low. When $\overline{RD}$ is low, enables the three-state outputs. If CONVST and RD are low, a conversion is initiated on the falling edge of $\overline{CS}$.
24	RD	Read Input - active low. When $\overline{CS}$ is low, RD enables the three-state outputs. If CONVST and CS are low, a conversion is initiated on the falling edge of $\overline{RD}$.

Detailed Description

ADC Operation

The MAX120/MAX122 use successive approximation and input T/H circuitry to convert an analog signal to a series of 12-bit digital-output codes. The control logic interfaces easily to most µPs, requiring only a few passive components for most applications. The T/H does not require an external capacitor. Figure 3 shows the MAX120/MAX122 in the simplest operational configuration.

Analog Input Track/Hold

Figure 4 shows the equivalent input circuit, illustrating the sampling architecture of the ADC's analog comparator. An internal buffer charges the hold capacitor to minimize the required acquisition time between conversions. The analog input appears as a 6kΩ resistor in parallel with a 10pF capacitor.

Between conversions, the buffer input is connected to AIN through the input resistance. When a conversion starts, the buffer input disconnects from AIN, thus sampling the input. At the end of the conversion, the buffer input reconnects to AIN, and the hold capacitor once again charges to the input voltage.

The T/H is in tracking mode whenever a conversion is NOT in progress. Hold mode starts approximately 10ns after a conversion is initiated. Variation in this delay from one conversion to the next (aperture jitter) is typically 30ps. Figures 7 through 11 detail the T/H mode and interface timing for the various interface modes.

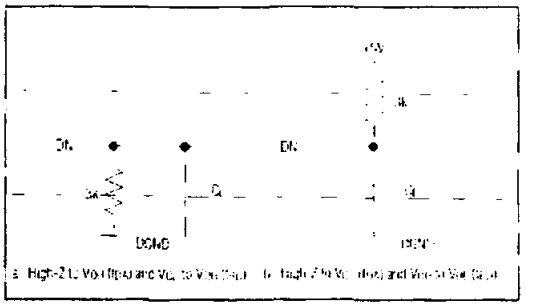


Figure 1. Load Circuits for Access Time

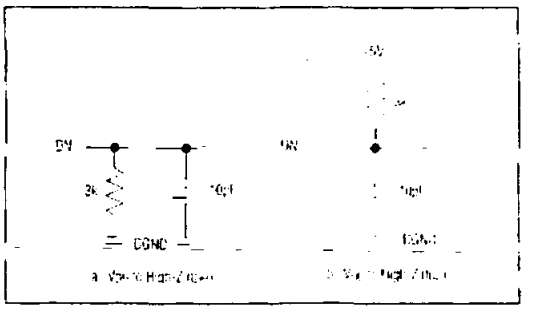


Figure 2. Load Circuits for Bus-Relinquish Time

500ksps, 12-Bit ADCs with Track/Hold and Reference

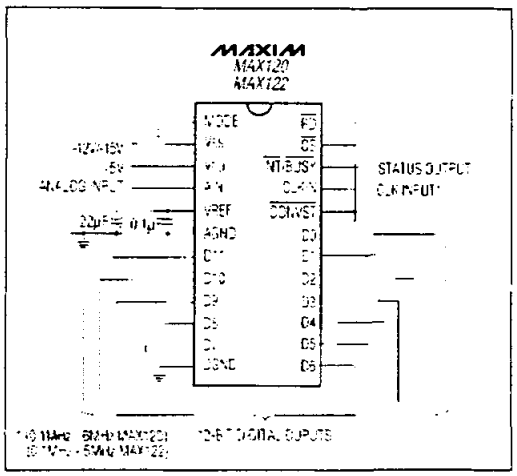


Figure 3. MAX120/MAX122 in the Simplest Operational Mode (Continuous Conversion)

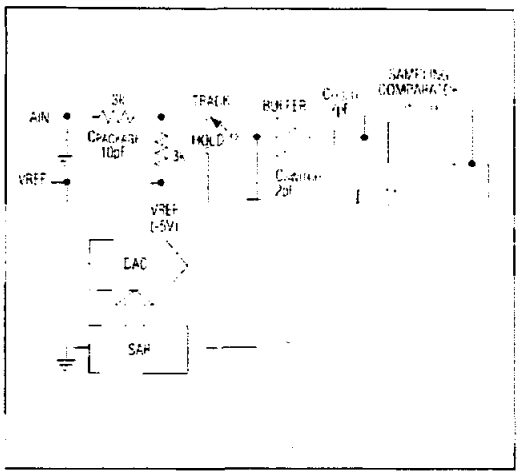


Figure 4. Equivalent Input Circuit

Internal Reference

The MAX120/MAX122 -5.00V buried-zener reference biases the internal DAC. The reference output is available at the VREF pin and must be bypassed to the AGND pin with a 0.1µF ceramic capacitor in parallel with a 22µF or greater electrolytic capacitor. The electrolytic capacitor's equivalent series resistance (ESR) must be 100mΩ or less to properly compensate the reference output buffer. Sanyo's organic semiconductor works well.

Sanyo Video Components (USA)
Phone: (619) 661-6835 FAX: (619) 661-1055

Sanyo Electric Company, LTD. (Japan)
Phone: 0720-70-1005 FAX: 0720-70-1174

Sanyo Fisher Vertrieos GmbH (Germany)
Phone: 06102-27041, ext. 44 FAX: 06102-27045

Proper bypassing minimizes reference noise and maintains a low impedance at high frequencies. The internal reference output buffer can sink up to a 5mA external load.

An external reference voltage can be used to overdrive the MAX120/MAX122's internal reference. It ranges from -5.05V to -5.10V and is capable of sinking a minimum of 5mA. The external VREF bypass capacitors are still required.

Digital Interface

External Clock

The MAX120/MAX122 require a TTL-compatible clock for proper operation. The MAX120 accepts clocks in the 0.1MHz to 8MHz frequency range when operating in modes 1-4 (see *Operating Modes* section). The maximum clock frequency is limited to 6MHz when operating in mode 5. The MAX122 requires a 0.1MHz to 5MHz clock for operation in all five modes. The minimum clock frequency for both the MAX120 and MAX122 is limited to 0.1MHz, due to the T/H's droop rate.

Clock and Control Synchronization

If the clock and convert start inputs (CONVST or RD and $\overline{CS}$ - see *Operating Modes* section) are not synchronized, the conversion time can vary from 13 to 14 clock cycles. The successive approximation register (SAR) always changes state on the CLKIN input's rising edge. To ensure a fixed conversion time, refer to Figure 5 and the following guidelines.

For a conversion time of 13 clock cycles, the convert start input(s) should go low at least 50ns before CLKIN's next rising edge. For a conversion time of 14 clock cycles, the convert start input(s) should go low within 10ns of CLKIN's next rising edge. If the convert start input(s) go low from 10ns to 50ns before CLKIN's next rising edge, the number of clock cycles required is undefined and can be either 13 or 14. For best analog performance, synchronize the convert start inputs with the clock input.

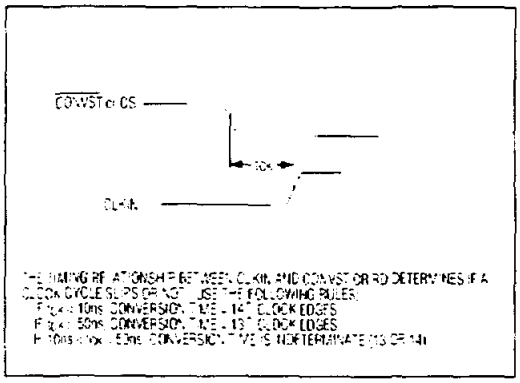


Figure 5. Clock and Control Synchronization

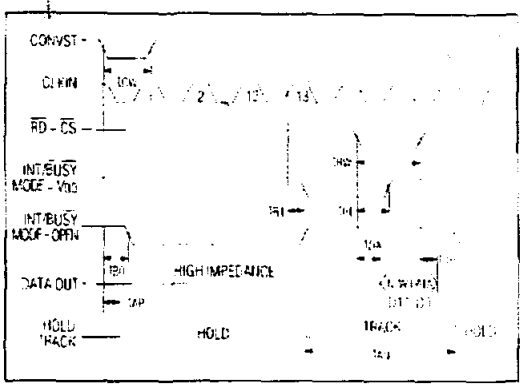


Figure 7. Full-Control Mode (Mode 1)

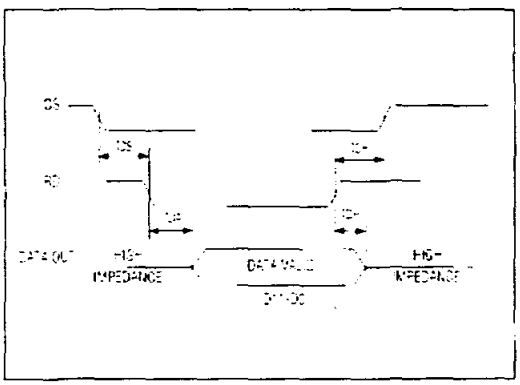


Figure 6. Data-Access and Bus-Relinquish Timing

Output Data Format

The conversion result is output on a 12-bit data bus with a 75ns data-access time. The output data format is twos-complement. The three input control signals (CS, RD, and CONVST), the INT/BUSY converter status output, and the 12 bits of output data can interface directly to a 16-bit data bus. See Figure 6 for data-access timing.

Timing and Control

The MAX120/MAX122 have five operational modes as outlined in Figures 7–11 and discussed in the *Operating Modes* section.

Full-control mode (mode 1) provides maximum control to the user for convert start and data-read operations. Full-control mode is for µPs with or without wait-state capability. Stand-alone mode (mode 2) and continuous-conversion mode (mode 5) are for systems without µPs, or for µP-based systems where the ADC and the µP are linked through first in, first out (FIFO) buffers or direct memory access (DMA) ports. Slow-memory mode (mode 3) is intended for µPs that can be forced into a wait state during the ADC's conversion time. ROM mode (mode 4) is for µPs that cannot be forced into a wait state.

In all five operating modes, the start of a conversion is controlled by one of three digital inputs: CONVST, RD, or CS. Figure 12 shows the logic equivalent for the conversion circuitry. In any operating mode, CONVST must be low for a conversion to occur. Once the conversion is in progress, it cannot be restarted.

Read operations are controlled by RD and CS. Both of these digital inputs must be low to read output data. The INT/BUSY output indicates the converter's status and determines when the data from the most recent conversion is available. The MODE input configures the INT/BUSY output as follows:

If MODE = VDD, INT/BUSY functions as an INTERRUPT output. In this configuration, INT/BUSY goes low when the conversion is complete and returns high after the conversion data has been read.

If MODE is left open or tied to DGND, INT/BUSY functions as a BUSY output. In this case, INT/BUSY goes low at the start of a conversion and remains low until the conversion is complete and the data is available at D0–D11.

500ksps, 12-Bit ADCs with Track/Hold and Reference

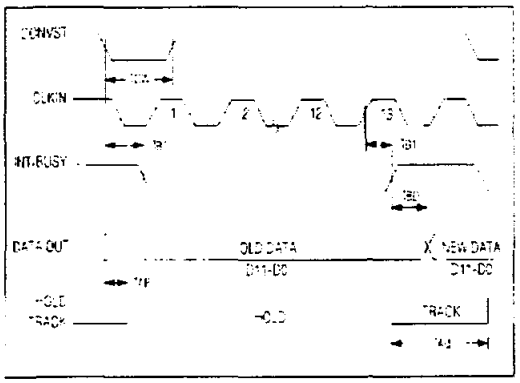


Figure 8 Stand-Alone Mode (Mode 2)

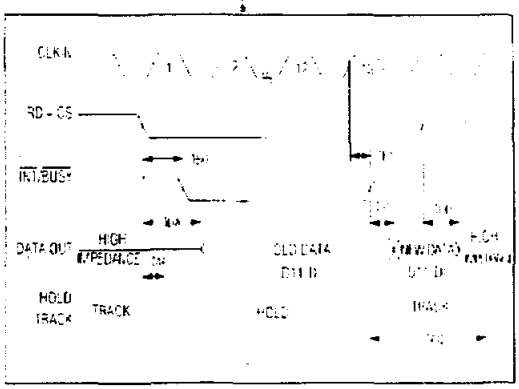


Figure 9 Slow-Memory Mode (Mode 3)

Initialization After Power-Up

On power-up, the first MAX120/MAX122 conversion is valid if the following conditions are met:

1) Allow 14 clock cycles for the internal T/H to enter the track mode, plus a minimum of 350ns in the track mode for the data-acquisition time.
2) Make sure the reference voltage has settled. Allow 0.5ms for each 1µF of reference bypass capacitance (11ms for a 22µF capacitor).

Operating Modes

Mode 1: (Full-Control Mode)

Figure 7 shows the timing diagram for full-control mode (mode 1). In this mode, the µP controls the conversion-start and data-read operations independently.

A falling edge on $\overline{CONVST}$ places the T/H into hold mode and starts a conversion in the SAR. The conversion is complete in 13 or 14 clock cycles as discussed in the *Clock and Control Synchronization* section. A change in the INT/BUSY output state signals the end of a conversion as follows:

If MODE = V_{DD}, the end of conversion is signaled by the INT/BUSY output falling edge.

If MODE = OPEN or DGND, the $\overline{INT}/\overline{BUSY}$ output goes low while the conversion is in progress and returns high when the conversion is complete.

When the conversion is complete, the data can be read without initiating a new conversion by pulling RD and CS low and leaving CONVST high. To start a new conversion without reading data, RD and CS should remain high while $\overline{CONVST}$ is driven low. To simultaneously read data and initiate a new conversion, CONVST, RD, and CS should all be pulled low. Note: Allow at least 350ns for T/H acquisition time between the end of one conversion and the beginning of the next.

Mode 2: Stand-Alone Operation (MODE = OPEN, $\overline{RD}$ = CS = DGND)

For systems that do not use or require full-bus interfacing, the MAX120/MAX122 can be operated in stand-alone mode directly linked to memory through DMA ports or a FIFO buffer. In stand-alone mode, a conversion is initiated by a falling edge on $\overline{CONVST}$. The data outputs are always enabled; data changes at the end of a conversion as indicated by a rising edge on INT/BUSY. See Figure 8 for stand-alone mode timing.

Mode 3: Slow-Memory Mode (CONVST = GND, MODE = OPEN)

Taking RD and CS low places the T/H into hold mode and starts a conversion. INT/BUSY remains low while the conversion is in progress and can be used as a wait input to the µP. Data from the previous conversion appears on the data bus until the conversion end is indicated by INT/BUSY. See Figure 9 for slow-memory mode timing.

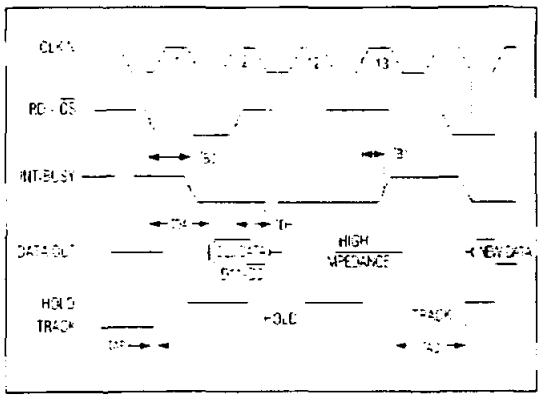


Figure 10. ROM Mode (Mode 4)

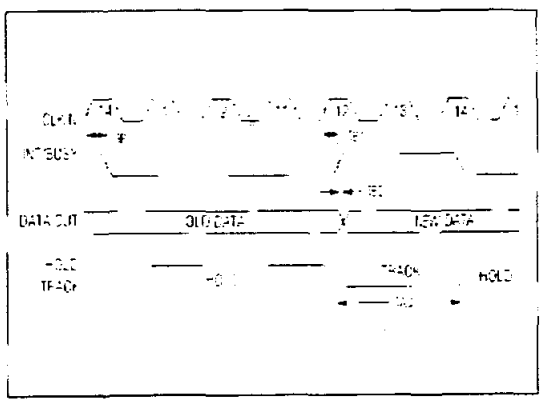


Figure 11. Continuous-Conversion Mode (Mode 5)

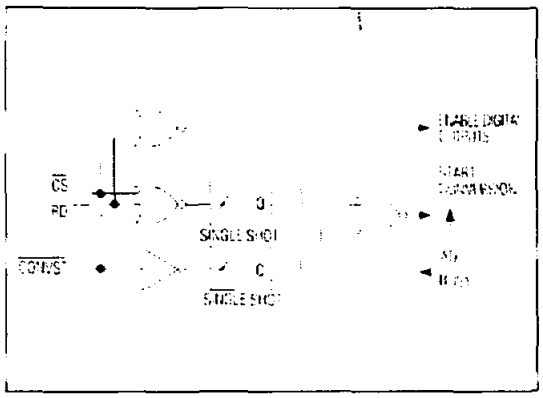


Figure 12. Conversion-Control Logic

Mode 4: ROM Mode (MODE = OPEN, CONVST = GND)

In ROM mode, the MAX120/MAX122 behave like a fast-access memory location and avoid placing the µP into a wait state. Pulling RD and CS low places the T/H in hold mode, starts a conversion, and reads data from the previous conversion. Data from the first read in a sequence is often disregarded when this interface mode is used. A second read operation accesses the first conversion's result and also starts a new conversion. The time between successive read operations must be longer than the sum of the T/H acquisition time and the MAX120/MAX122 conversion time. See Figure 10 for ROM-mode timing.

Mode 5: Continuous-Conversion Mode (CONVST = RD = CS = MODE = GND)

For systems that do not use or require full-bus interfacing, the MAX120/MAX122 can operate in continuous-conversion mode, directly linked to memory through DMA ports or a FIFO buffer. In this mode, conversions are performed continuously at the rate of one conversion for every 14 clock cycles, which includes 2 clock cycles for the T/H acquisition time. To satisfy the 350ns minimum acquisition time requirement within 2 clock cycles, the MAX120's maximum clock frequency is 6MHz when operating in mode 5.

The data outputs are always enabled and new data appears on the output bus at the end of a conversion as indicated by the INT/BUSY output rising edge. The MODE input should be hard-wired to GND. Pulling CS, RD, or CONVST high stops conversions. See Figure 11 for continuous-conversion mode timing.

Applications Information

Using FIFO Buffers

Using FIFO memory to buffer blocks of data from the MAX120 reduces µP interrupt overhead time by enabling the µP to process data while the MAX120, unassisted, writes conversion results to the FIFO. To retrieve a block of data, the µP reads from the FIFO via a read-interrupt cycle. Read and write operations for the FIFO are completely asynchronous. Figure 13 shows the MAX120 operating in continuous-conversion mode (mode 5), writing data directly into the two IDT7200 256 x 9 FIFO buffers at the rate of 428ksps. The µP is interrupted to read the accumulated data by the FIFO's half-full (HF) flag approximately three times per millisecond. For operation at 500ksps, use an 8MHz clock, and pulse CONVST at 500kHz. The full flag (FF) indicates that the FIFO is full. If this flag is ignored, data may be lost. If necessary, conversions can be inhibited by pulling CS, RD, or CONVST high. The FIFO's read cycle times are as fast as 15ns, satisfying most system speed requirements. The RESET input resets all data in the FIFO to zero.

500ksps, 12-Bit ADCs with Track/Hold and Reference

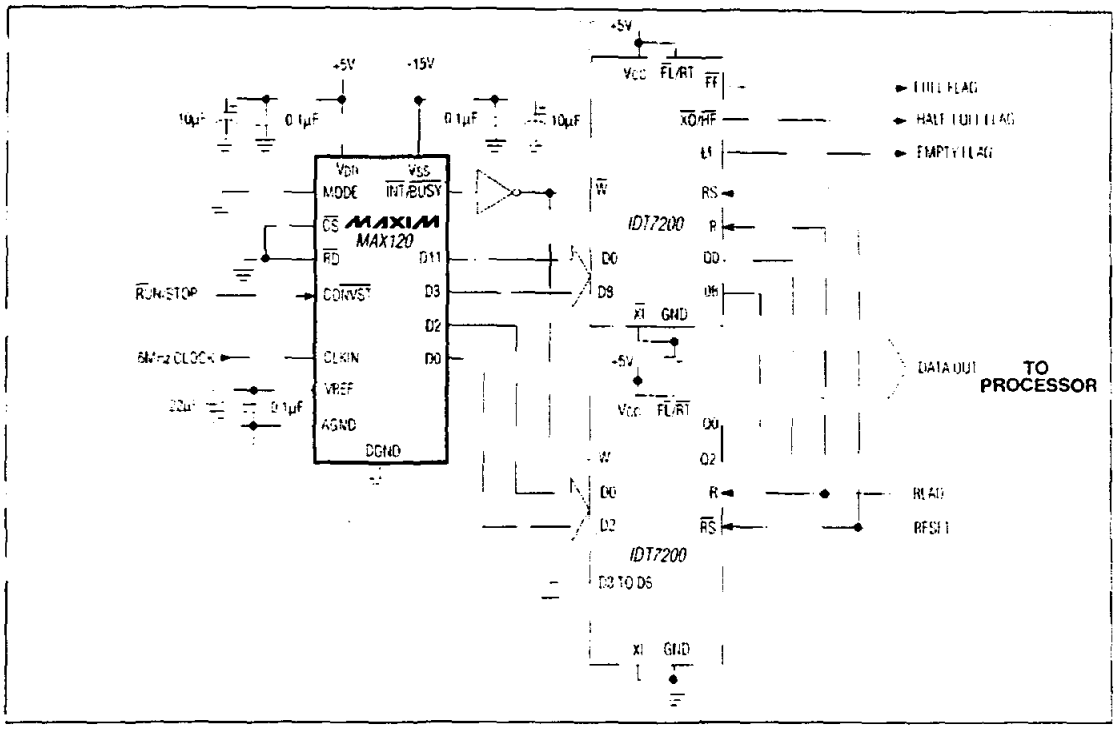


Figure 13. Using MAX120 with FIFO Memory

For synchronous operation, the $\overline{CONVST}$ pin may be used to initiate conversions, as described in the *Operating Modes* section (Mode 2: Stand-Alone Operation).

Digital-Bus Noise

If the ADC's data bus is active during a conversion, coupling from the data pins to the ADC comparator can cause errors. Using slow-memory mode (mode 3) avoids this problem by placing the µP in a wait state during the conversion. If the data bus is active during the conversion in either mode 1 or 4, use three-state drivers to isolate the bus from the ADC.

In ROM mode (mode 4), considerable digital noise is generated in the ADC when $\overline{RD}$ or $\overline{CS}$ go high, disabling the output buffers after a conversion is started. This noise can cause errors if it occurs at the same instant the SAR latches a comparator decision. To avoid this problem, RD and CS should be active for less than 1 clock cycle. If this is not possible, $\overline{RD}$ or $\overline{CS}$ should go high coinciding with CLKIN's falling edge, since the comparator output is always latched at CLKIN's rising edge.

Layout, Grounding, and Bypassing

For best system performance, use printed circuit boards with separate analog and digital ground planes. Wire-wrap boards are not recommended. The two ground planes should be tied together at the low-impedance power-supply source, as shown in Figure 14.

The board layout should ensure that digital and analog signal lines are kept separate from each other as much as possible. Do not run analog and digital (especially clock) lines parallel to one another.

The ADC's high-speed comparator is sensitive to high-frequency noise in the V_{DD} and V_{SS} power supplies. Bypass these supplies to the analog ground plane with 0.1µF and 10µF bypass capacitors. Minimize capacitor lead lengths for best noise rejection. If the -5V power supply is very noisy, connect a 5Ω resistor, as shown in Figure 14. Figure 15 shows the negative power-supply (V_{SS}) rejection vs. frequency. Figure 16 shows the positive power-supply (V_{DD}) rejection vs. frequency, with and without the optional 5Ω resistor.

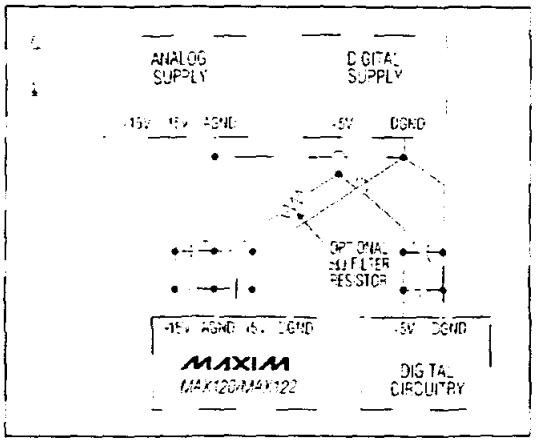


Figure 14. Power-Supply Grounding

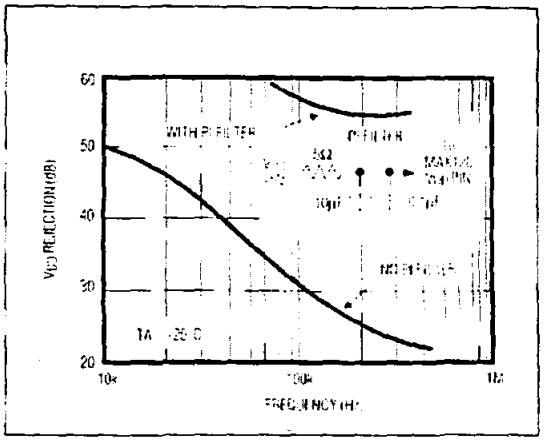


Figure 16. V_{DD} Power-Supply Rejection vs. Frequency

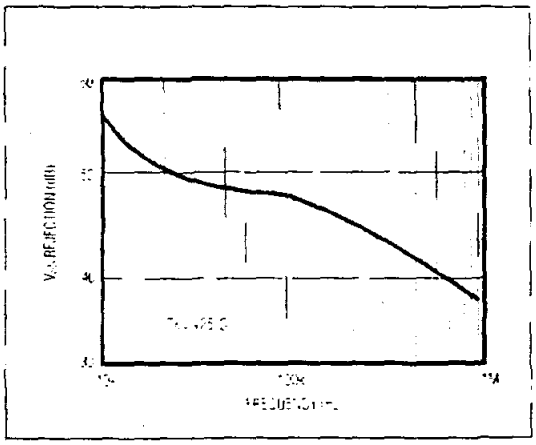


Figure 15. V_{SS} Power-Supply Rejection vs. Frequency

Gain and Offset Adjustment

Figure 17 plots the bipolar input/output transfer function for the MAX120/MAX122. Code transitions occur halfway between successive integer LSB values. Output coding is twos-complement binary with 1LSB = 2.44mV (10V/4096).

In applications where gain (full-scale range) adjustment is required, Figure 18's circuit can be used. If both offset and gain (full-scale range) need adjustment, either of the circuits in Figures 19 and 20 can be used. Offset should be adjusted before gain for either of these circuits.

To adjust bipolar offset with Figure 19's circuit, apply +1/2LSB (0.61mV) to the noninverting amplifier input and adjust R4 for output-code flicker between 0000 0000 0000 and 0000 0000 0001. For full scale, apply FS - 1/2LSB (2.4988V) to the amplifier input and adjust R2 so the output code flickers between 0111 1111 1110 and 0111 1111 1111. There may be some interaction between these adjustments. The MAX120/MAX122 transfer function used in conjunction with Figure 19's circuit is the same as Figure 17, except the full-scale range is reduced to 2.5V.

To adjust bipolar offset with Figure 20's circuit, apply -1/2LSB (-1.22mV) at V_{IN} and adjust R5 for output-code flicker between 0000 0000 0000 and 0000 0000 0001. For gain adjustment, apply -FS - 1/2LSB (-4.995V) at V_{IN} and adjust R1 so the output code flickers between 0111 1111 1110 and 0111 1111 1111. As with Figure 20's circuit, the offset and gain adjustments may interact. Figure 21 plots the transfer function for Figure 20's circuit.

Dynamic Performance

High-speed sampling capability and 500ksps throughput (333ksps for the MAX122) make the MAX120/MAX122 ideal for wideband-signal processing. To support these and other related applications, fast fourier transform (FFT) test techniques are used to guarantee the ADC's dynamic frequency response, distortion, and noise at the rated throughput. Specifically, this involves applying a low-distortion sine wave to the ADC input and recording the digital conversion results for a specified time. The data is then analyzed using an FFT algorithm, which determines its spectral content.

500ksps, 12-Bit ADCs with Track/Hold and Reference

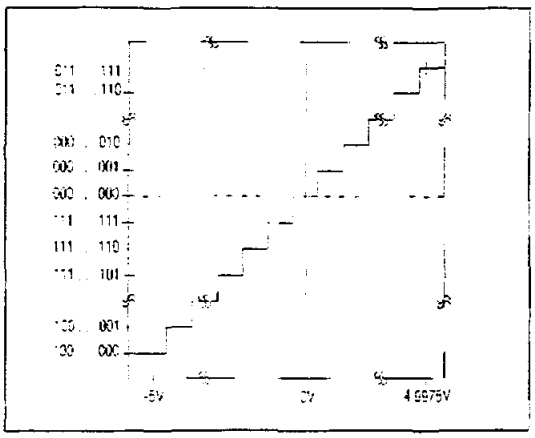


Figure 17. Bipolar Transfer Function

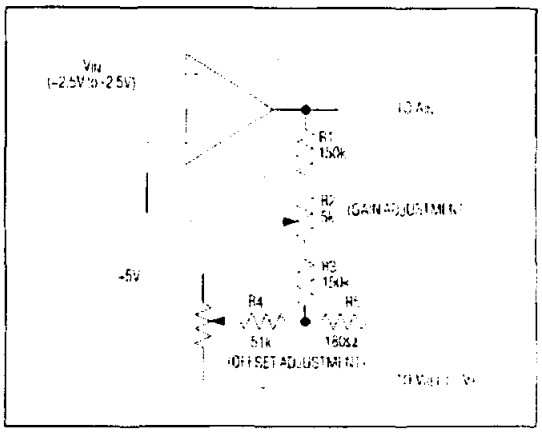


Figure 19. Offset and Gain Adjustment (Noninverting)

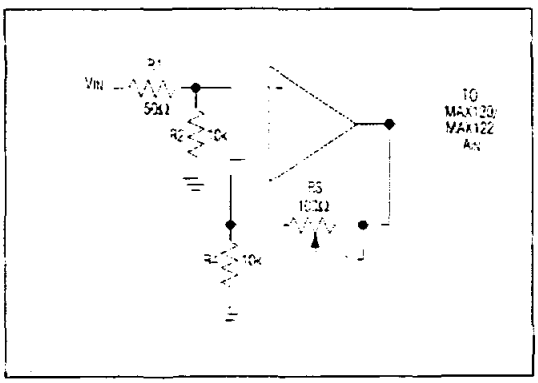


Figure 18. Trim Circuit for Gain Only

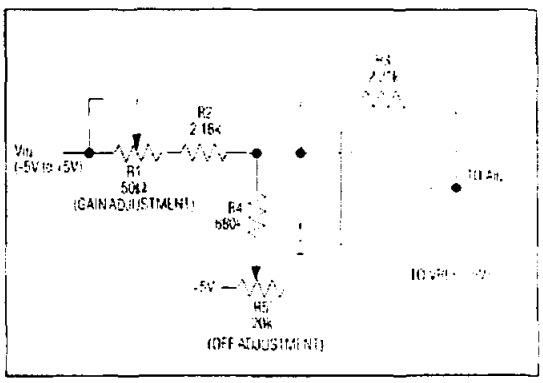


Figure 20. Offset and Gain Adjustment (Inverting)

ADCs have traditionally been evaluated by specifications such as zero and full-scale error, integral nonlinearity (INL), and differential nonlinearity (DNL). Such parameters are widely accepted for specifying performance with DC and slowly varying signals, but are less useful in signal processing applications where the ADC's impact on the system transfer function is the main concern. The significance of various DC errors does not translate well to the dynamic case, so different tests are required.

Signal-to-Noise Ratio and Effective Number of Bits

The signal-to-noise plus distortion ratio (SINAD) is the ratio of the fundamental input frequency's RMS amplitude to the RMS amplitude of all other ADC output signals. The output band is limited to frequencies above DC and below one-half the ADC sample rate.

The theoretical minimum ADC noise is caused by quantization error and is a direct result of the ADC's resolution: SNR = (6.02N + 1.76)dB, where N is the number of bits of resolution. A perfect 12-bit ADC can, therefore, do no better than 74dB. An FFT plot shows the output level in various spectral bands. Figure 22 shows the result of sampling a pure 100kHz sinusoid at a 500ksps rate with the MAX120.

By transposing the equation that converts resolution to SNR, we can, from the measured SINAD, determine the effective resolution (or effective number of bits) the ADC provides: N = (SINAD - 1.76)/6.02. Figure 22 shows the effective number of bits as a function of the input frequency for the MAX120. The MAX122 performs similarly.

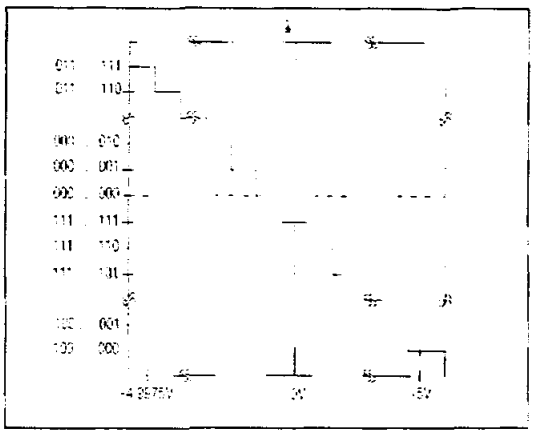


Figure 21. Inverting Bipolar Transfer Function

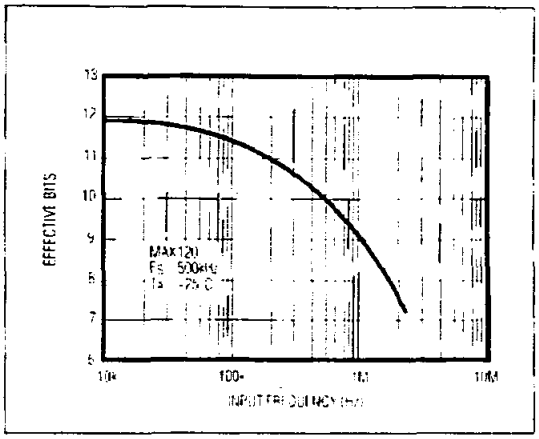


Figure 23. Effective Bits vs. Input Frequency

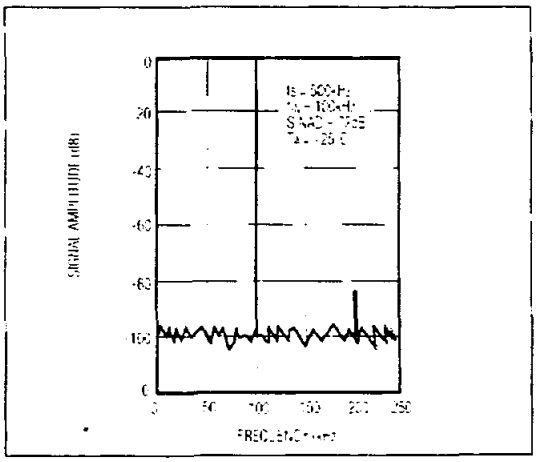


Figure 22. MAX120 FFT Plot

Total Harmonic Distortion

If a pure sine wave is sampled by an ADC at greater than the Nyquist frequency, the nonlinearities in the ADC's transfer function create harmonics of the input frequency in the sampled output data.

Total harmonic distortion (THD) is the ratio of the RMS sum of all harmonics (in the frequency band above DC and below one-half the sample rate, but not including the DC component) to the RMS amplitude of the fundamental frequency. This is expressed as follows:

$$THD = 20\log \frac{\sqrt{V_2^2 + V_3^2 + V_4^2 + \ldots + V_N^2}}{V_1}$$

where V_1 is the fundamental RMS amplitude, and V_2 to V_N are the amplitudes of the 2nd through Nth harmonics. The THD specification in the *Electrical Characteristics* table includes the 2nd through 5th harmonics.

Intermodulation Distortion

If the ADC input signal consists of more than one spectral component, the ADC transfer function nonlinearities produce intermodulation distortion (IMD) in addition to THD. IMD is the change in one sinusoidal input caused by the presence of another sinusoidal input at a different frequency.

If two pure sine waves of frequency fa and fb are applied to the ADC input, nonlinearities in the ADC transfer function create distortion products at sum and difference frequencies of mfa ± nfb, where m and n = 0, 1, 2, 3, etc. THD includes those distortion products with m or n equal to zero. Intermodulation distortion consists of all distortion products for which neither m nor n equal zero. For example, the 2nd-order IMD terms include (fa + fb) and (fa - fb) while the 3rd-order IMD terms include (2fa + fb), (2fa - fb), (fa + 2fb), and (fa - 2fb).

If the two input sine waves are equal in magnitude, the value (in decibels) of the 2nd-order IMD products can be expressed by the following formula:

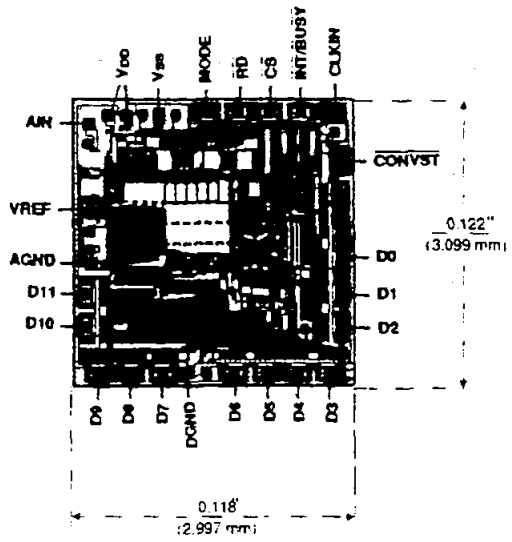
$$IMD\ (fa \pm fb) = 20\log \left[\frac{\text{amplitude at } (fa \pm fb)}{\text{amplitude at fa}} \right]$$

500ksps, 12-Bit ADCs with Track/Hold and Reference

Spurious-Free Dynamic Range

Spurious-free dynamic range is the ratio of the fundamental RMS amplitude to the amplitude of the next largest spectral component (in the frequency band above DC and below one-half the sample rate). Usually the next largest spectral component occurs at some harmonic of the input frequency. However, if the ADC is exceptionally linear, it may occur only at a random peak in the ADC's noise floor.

Chip Topography



MAX120/MAX122 TRANSISTOR COUNT: 1920
SUBSTRATE CONNECTED TO V_{DD}.

Ordering Information (continued)

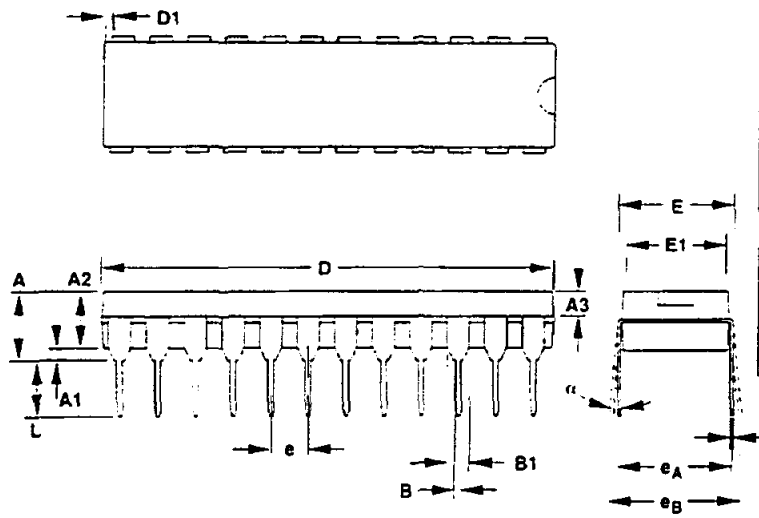
PART	TEMP. RANGE	PIN-PACKAGE	INL (LSBs)
MAX120EAG	-40°C to +85°C	24 SSOP	±1
MAX120MRG	-55°C to +125°C	24 Narrow CERDIP	±2
MAX122ACNG	0°C to +70°C	24 Narrow Plastic DIP	±3/4
MAX122BCNG	0°C to +70°C	24 Narrow Plastic DIP	±1
MAX122ACWG	0°C to +70°C	24 Wide SO	±3/4
MAX122BCWG	0°C to +70°C	24 Wide SO	±1
MAX122ACAG	0°C to +70°C	24 SSOP	±3/4
MAX122BCAG	0°C to +70°C	24 SSOP	±1
MAX122BC/D	0°C to +70°C	Dice*	±1
MAX122AENG	-40°C to +85°C	24 Narrow Plastic DIP	±3/4
MAX122BENG	-40°C to +85°C	24 Narrow Plastic DIP	±1
MAX122AEWG	-40°C to +85°C	24 Wide SO	±3/4
MAX122BEWG	-40°C to +85°C	24 Wide SO	±1
MAX122AEAG	-40°C to +85°C	24 SSOP	±3/4
MAX122BEAG	-40°C to +85°C	24 SSOP	±1
MAX122BMRG	-55°C to +125°C	24 Narrow CERDIP	±1
MAX120EVKIT-DIP†	0°C to +70°C	Plastic DIP - Through Hole	

** Contact factory for dice specifications.*

† MAX120 EV kit can be used to evaluate the MAX122; when ordering the EV kit, ask for a free sample of the MAX122.

500ksps, 12-Bit ADCs with Track/Hold and Reference

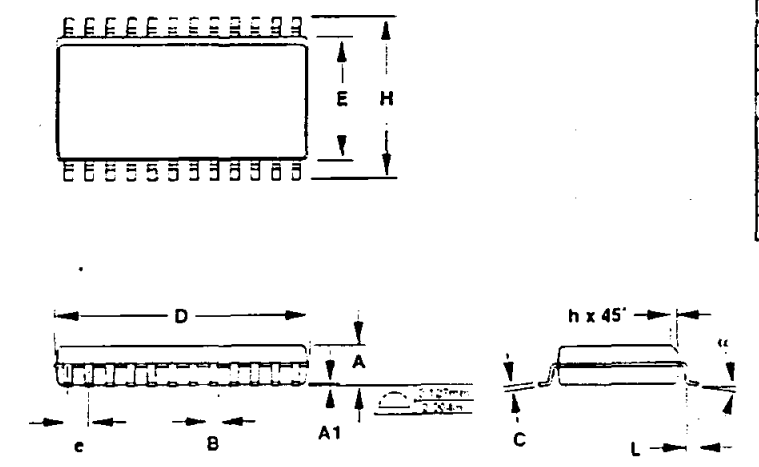
Package Information



DIM	INCHES		MILLIMETERS	
	MIN	MAX	MIN	MAX
A	–	0.200	–	5.08
A1	0.015	–	0.38	–
A2	0.125	0.150	3.18	3.81
A3	0.055	0.080	1.40	2.03
B	0.016	0.022	0.41	0.56
B1	0.050	0.065	1.27	1.65
C	0.008	0.012	0.20	0.30
D	1.235	1.265	31.37	32.13
D1	0.050	0.080	1.27	2.03
E	0.300	0.325	7.62	8.26
E1	0.240	0.280	6.10	7.11
e	0.100 BSC		2.54 BSC	
e_A	0.300 BSC		7.62 BSC	
e_B	–	0.400	–	10.16
L	0.115	0.150	2.92	3.81
α	0°	15°	0°	15°

21-337A

24-PIN PLASTIC DUAL-IN-LINE (NARROW) PACKAGE



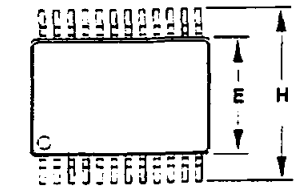
DIM	INCHES		MILLIMETERS	
	MIN	MAX	MIN	MAX
A	0.093	0.104	2.35	2.65
A1	0.004	0.012	0.10	0.30
B	0.014	0.019	0.35	0.49
C	0.009	0.013	0.23	0.32
D	0.598	0.614	15.20	15.60
E	0.291	0.299	7.40	7.60
e	0.050 BSC		1.27 BSC	
H	0.394	0.419	10.00	10.65
h	0.010	0.030	0.25	0.75
L	0.016	0.050	0.40	1.27
α	0°	8°	0°	8°

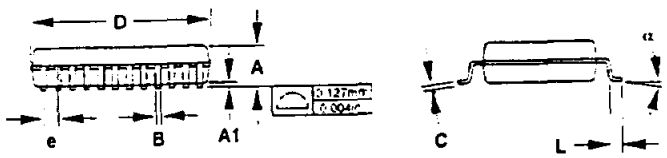
21-336A

24-PIN PLASTIC SMALL-OUTLINE PACKAGE

500ksps, 12-Bit ADCs with Track/Hold and Reference

Package Information (continued)





DIM	INCHES		MILLIMETERS	
	MIN	MAX	MIN	MAX
A	0.068	0.078	1.73	1.99
A1	0.002	0.008	0.05	0.21
B	0.010	0.015	0.25	0.38
C	0.005	0.009	0.13	0.22
D	0.317	0.328	.807	8.33
E	0.205	0.212	5.20	5.38
e	0.0256 BSC		0.65 BSC	
H	0.301	0.311	7.65	7.90
L	0.022	0.037	0.55	0.95
α	0°	8	0	8

21-0036A

24-PIN PLASTIC SHRINK SMALL-OUTLINE PACKAGE

DC CHARACTERISTICS

(V_{CC} = 5.0 V ± 5%, V_{SS} = 0, T_A = T_L to T_H, unless otherwise noted)

Parameter	Symbol	Min	Typ	Max	Unit	Test Conditions
Input High Voltage Except XTALI and XTALO XTALI and XTALO	V_{IH}	+2.0 +2.4	--	V_{CC} + 0.3 V_{CC} + 0.3	V	
Input Low Voltage Except XTALI and XTALO XTALI and XTALO	V_{IL}	-0.3 -0.3	--	+0.8 +0.4	V	
Input Leakage Current R/W, RES, RS0, RS1, RS2, RxD, CTS, DCD, $\overline{DSR}$, RxC, TxC, CS	I_{IN}	--	10	50	µA	V_{IN} = 0V to 5.0V V_{CC} = 5.25V
Input Leakage Current for Three State Off D0-D7	I_{TSI}	--	±2	10	µA	V_{IN} = 0.4V to 2.4V V_{CC} = 5.25V
Output High Voltage D0-D7, TxD, CLK OUT, RTS, DTR	V_{OH}	+2.4	--	--	V	V_{CC} = 4.75V I_{LOAD} = -100 µA
Output Low Voltage D0-D7, TxD, CLK OUT, RTS, DTR	V_{OL}	--	--	+0.4	V	V_{CC} = 4.75V I_{LOAD} = 1.6 mA
Output Leakage Current (Off State) IRQ	I_{OFF}	--	±2	±10	µA	V_{CC} = 5.25V V_{OUT} = 0 to 2.4V
Power Dissipation	P_D	--	--	10	mW/MHz	
Input Capacitance Except XTALI and XTALO XTALI and XTALO	C_{IN}	-- --	-- --	5 10	pF pF	V_{CC} = 5.0V V_{IN} = 0V f = 2 MHz T_A = 25°C
Output Capacitance	C_{OUT}	--		10	pF	

Notes:
1. All units are direct current (dc) except for capacitance.
2. Negative sign indicates outward current flow, positive indicates inward flow.
3. Typical values are shown for V_{CC} = 5.0V and T_A = 25°C.

- **Compatible with All Intel and Most Other Microprocessors**
- **Handles Inputs from DC to 10 MHz**
  - **— 5 MHz 8254-5**
  - **— 8 MHz 8254**
  - **— 10 MHz 8254-2**
- **Status Read-Back Command**
- **Six Programmable Counter Modes**
- **Three Independent 16-Bit Counters**
- **Binary or BCD Counting**
- **Single +5V Supply**
- **Available in EXPRESS**
  - **— Standard Temperature Range**

The Intel® 8254 is a counter/timer device designed to solve the common timing control problems in microcomputer system design. It provides three independent 16-bit counters, each capable of handling clock inputs up to 10 MHz. All modes are software programmable. The 8254 is a superset of the 8253.

The 8254 uses HMOS technology and comes in a 24-pin plastic or CERDIP package.

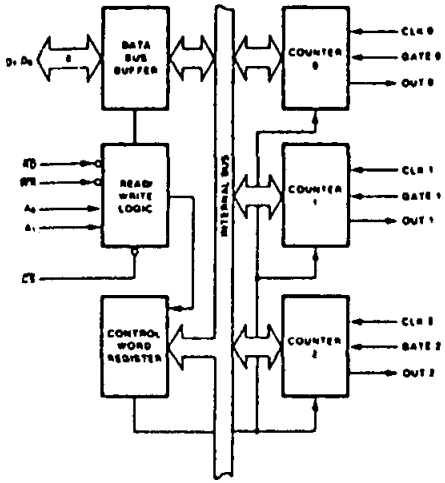


Figure 1. 8254 Block Diagram

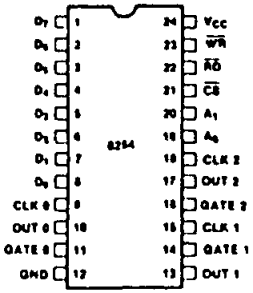


Figure 2. Pin Configuration

FUNCTIONAL DESCRIPTION

General

The 8254 is a programmable interval timer/counter designed for use with Intel microcomputer systems. It is a general purpose, multi-timing element that can be treated as an array of I/O ports in the system software.

The 8254 solves one of the most common problems in any microcomputer system, the generation of accurate time delays under software control. Instead of setting up timing loops in software, the programmer configures the 8254 to match his requirements and programs one of the counters for the desired delay. After the desired delay, the 8254 will interrupt the CPU. Software overhead is minimal and variable length delays can easily be accommodated.

Some of the other counter/timer functions common to microcomputers which can be implemented with the 8254 are:

- Real time clock
- Event-counter
- Digital one-shot
- Programmable rate generator
- Square wave generator
- Binary rate multiplier
- Complex waveform generator
- Complex motor controller

Block Diagram

DATA BUS BUFFER

This 3-state, bi-directional, 8-bit buffer is used to interface the 8254 to the system bus (see Figure 3).

Table 1. Pin Description

Symbol	Pin No.	Type	Name and Function
D_7-D_0	1-8	I/O	DATA: Bi-directional three state data bus lines, connected to system data bus.
CLK 0	9	I	CLOCK 0: Clock input of Counter 0.
OUT 0	10	O	OUTPUT 0: Output of Counter 0.
GATE 0	11	I	GATE 0: Gate input of Counter 0.
GND	12		GROUND: Power supply connection.
V_{CC}	24		POWER: +5V power supply connection.
$\overline{WR}$	23	I	WRITE CONTROL: This input is low during CPU write operations.
$\overline{RD}$	22	I	READ CONTROL: This input is low during CPU read operations.
$\overline{CS}$	21	I	CHIP SELECT: A low on this input enables the 8254 to respond to $\overline{RD}$ and $\overline{WR}$ signals. $\overline{RD}$ and $\overline{WR}$ are ignored otherwise.
A_1, A_0	20-19	I	ADDRESS: Used to select one of the three Counters or the Control Word Register for read or write operations. Normally connected to the system address bus.

A_1	A_0	Selects
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Control Word Register

CLK 2	18	I	CLOCK 2: Clock input of Counter 2.
OUT 2	17	O	OUT 2: Output of Counter 2.
GATE 2	16	I	GATE 2: Gate input of Counter 2.
CLK 1	15	I	CLOCK 1: Clock input of Counter 1.
GATE 1	14	I	GATE 1: Gate input of Counter 1.
OUT 1	13	O	OUT 1: Output of Counter 1.

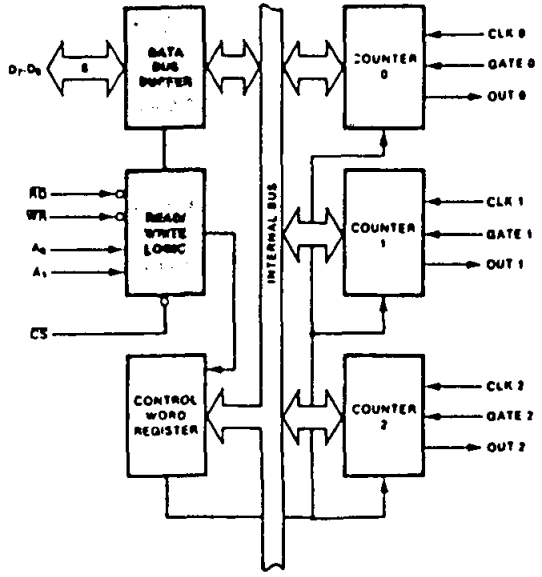


Figure 3. Block Diagram Showing Data Bus Buffer and Read/Write Logic Functions

READ/WRITE LOGIC

The Read/Write Logic accepts inputs from the system bus and generates control signals for the other functional blocks of the 8254. A_1 and A_0 select one of the three counters or the Control Word Register to be read from/written into. A "low" on the $\overline{RD}$ input tells the 8254 that the CPU is reading one of the counters. A "low" on the $\overline{WR}$ input tells the 8254 that the CPU is writing either a Control Word or an initial count. Both $\overline{RD}$ and $\overline{WR}$ are qualified by $\overline{CS}$; $\overline{RD}$ and $\overline{WR}$ are ignored unless the 8254 has been selected by holding $\overline{CS}$ low.

CONTROL WORD REGISTER

The Control Word Register (see Figure 4) is selected by the Read/Write Logic when $A_1,A_0 = 11$. If the CPU then does a write operation to the 8254, the data is stored in the Control Word Register and is interpreted as a Control Word used to define the operation of the Counters.

The Control Word Register can only be written to; status information is available with the Read-Back Command.

COUNTER 0, COUNTER 1, COUNTER 2

These three functional blocks are identical in operation, so only a single Counter will be described. The internal block diagram of a single counter is shown in Figure 5.

The Counters are fully independent. Each Counter may operate in a different Mode.

The Control Word Register is shown in the figure; it is not part of the Counter itself, but its contents determine how the Counter operates.

The status register, shown in Figure 5, when latched, contains the current contents of the Control Word Register and status of the output and null count flag. (See detailed explanation of the Read-Back command.)

The actual counter is labelled CE (for "Counting Element"). It is a 16-bit presettable synchronous down counter.

OL_M and OL_L are two 8-bit latches. OL stands for "Output Latch"; the subscripts M and L stand for "Most significant byte" and "Least significant byte" respectively. Both are normally referred to as one unit and called just OL. These latches normally "follow" the CE, but if a suitable Counter Latch Command is sent to the 8254, the latches "latch" the present count until read by the CPU and then return to "following" the CE. One latch at a time is enabled by the counter's Control Logic to drive the internal bus. This is how the 16-bit Counter communicates over the 8-bit internal bus. Note that the CE itself cannot be read; whenever you read the count, it is the OL that is being read.

Similarly, there are two 8-bit registers called CR_M and CR_L (for "Count Register"). Both are normally

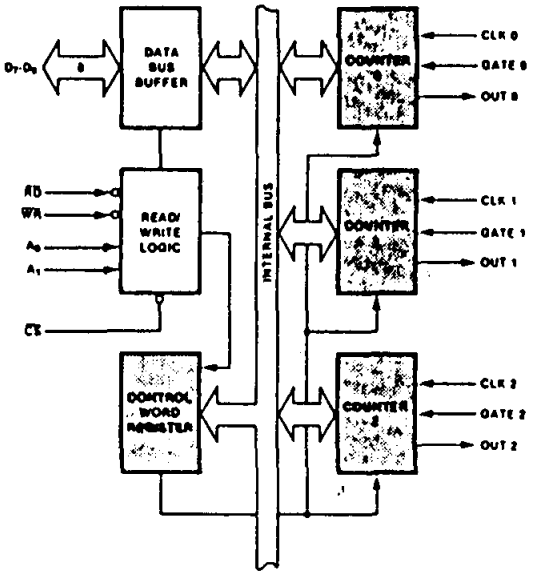


Figure 4. Block Diagram Showing Control Word Register and Counter Functions

referred to as one unit and called just CR. When a new count is written to the Counter, the count is stored in the CR and later transferred to the CE. The Control Logic allows one register at a time to be loaded from the internal bus. Both bytes are transferred to the CE simultaneously. CR_M and CR_L are cleared when the Counter is programmed. In this way, if the Counter has been programmed for one byte counts (either most significant byte only or least significant byte only) the other byte will be zero. Note that the CE cannot be written into; whenever a count is written, it is written into the CR.

The Control Logic is also shown in the diagram. CLK n, GATE n, and OUT n are all connected to the outside world through the Control Logic.

8254 SYSTEM INTERFACE

The 8254 is a component of the Intel Microcomputer Systems and interfaces in the same manner as all other peripherals of the family. It is treated by the system's software as an array of peripheral I/O ports; three are counters and the fourth is a control register for MODE programming.

Basically, the select inputs A_0,A_1 connect to the A_0, A_1 address bus signals of the CPU. The CS can be derived directly from the address bus using a linear select method. Or it can be connected to the output of a decoder, such as an Intel 8205 for larger systems.

OPERATIONAL DESCRIPTION

General

After power-up, the state of the 8254 is undefined. The Mode, count value, and output of all Counters are undefined.

How each Counter operates is determined when it is programmed. Each Counter must be programmed before it can be used. Unused counters need not be programmed.

Programming the 8254

Counters are programmed by writing a Control Word and then an initial count.

The Control Words are written into the Control Word Register, which is selected when A_1,A_0 = 11. The Control Word itself specifies which Counter is being programmed.

By contrast, initial counts are written into the Counters, not the Control Word Register. The A_1,A_0 inputs are used to select the Counter to be written into. The format of the initial count is determined by the Control Word used.

Write Operations

The programming procedure for the 8254 is very flexible. Only two conventions need to be remembered:

1) For each Counter, the Control Word must be written before the initial count is written.
2) The initial count must follow the count format specified in the Control Word (least significant byte only, most significant byte only, or least significant byte and then most significant byte).

Since the Control Word Register and the three Counters have separate addresses (selected by the A_1,A_0 inputs), and each Control Word specifies the Counter it applies to (SC0,SC1 bits), no special instruction sequence is required. Any programming sequence that follows the conventions in Figure 7 is acceptable.

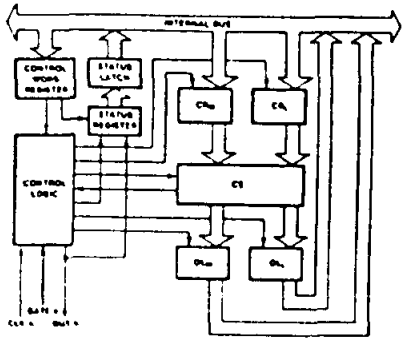


Figure 5. Internal Block Diagram of a Counter

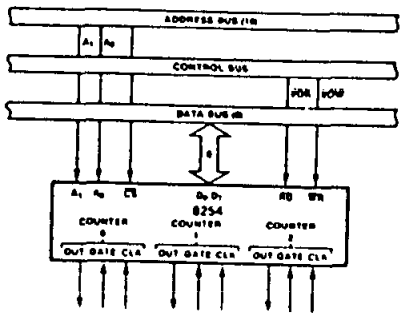


Figure 6. 8254 System Interface

Control Word Format

A_1,A_0 = 11 $\overline{CS}$ = 0 $\overline{RD}$ = 1 $\overline{WR}$ = 0

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD

SC—Select Counter

SC1	SC0	
0	0	Select Counter 0
0	1	Select Counter 1
1	0	Select Counter 2
1	1	Read-Back Command (see Read Operations)

M—Mode

M2	M1	M0	
0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

RW—Read/Write

RW1	RW0	
0	0	Counter Latch Command (see Read Operations)
0	1	Read/Write least significant byte only
1	0	Read/Write most significant byte only
1	1	Read/Write least significant byte first, then most significant byte

BCD

0	Binary Counter 16-bits
1	Binary Coded Decimal (BCD) Counter (4 Decades)

NOTE:
Don't care bits (X) should be 0 to insure compatibility with future Intel products.

Figure 7. Control Word Format

	A_1	A_0
Control Word—Counter 0	1	1
LSB of count—Counter 0	0	0
MSB of count—Counter 0	0	0
Control Word—Counter 1	1	1
LSB of count—Counter 1	0	1
MSB of count—Counter 1	0	1
Control Word—Counter 2	1	1
LSB of count—Counter 2	1	0
MSB of count—Counter 2	1	0

	A_1	A_0
Control Word—Counter 2	1	1
Control Word—Counter 1	1	1
Control Word—Counter 0	1	1
LSB of count—Counter 2	1	0
MSB of count—Counter 2	1	0
LSB of count—Counter 1	0	1
MSB of count—Counter 1	0	1
LSB of count—Counter 0	0	0
MSB of count—Counter 0	0	0

	A_1	A_0
Control Word—Counter 0	1	1
Control Word—Counter 1	1	1
Control Word—Counter 2	1	1
LSB of count—Counter 2	1	0
LSB of count—Counter 1	0	1
LSB of count—Counter 0	0	0
MSB of count—Counter 0	0	0
MSB of count—Counter 1	0	1
MSB of count—Counter 2	1	0

	A_1	A_0
Control Word—Counter 1	1	1
Control Word—Counter 0	1	1
LSB of count—Counter 1	0	1
Control Word—Counter 2	1	1
LSB of count—Counter 0	0	0
MSB of count—Counter 1	0	1
LSB of count—Counter 2	1	0
MSB of count—Counter 0	0	0
MSB of count—Counter 2	1	0

NOTE:
In all four examples, all Counters are programmed to read/write two-byte counts. These are only four of many possible programming sequences.

Figure 8. A Few Possible Programming Sequences

A new initial count may be written to a Counter at any time without affecting the Counter's programmed Mode in any way. Counting will be affected as described in the Mode definitions. The new count must follow the programmed count format.

If a Counter is programmed to read/write two-byte counts, the following precaution applies: A program must not transfer control between writing the first and second byte to another routine which also writes into that same Counter. Otherwise, the Counter will be loaded with an incorrect count.

Read Operations

It is often desirable to read the value of a Counter without disturbing the count in progress. This is easily done in the 8254.

There are three possible methods for reading the counters: a simple read operation, the Counter Latch Command, and the Read-Back Command. Each is explained below. The first method is to perform a simple read operation. To read the Counter, which is selected with the A1, A0 inputs, the CLK input of the selected Counter must be inhibited by using either the GATE input or external logic. Otherwise, the count may be in the process of changing when it is read, giving an undefined result.

COUNTER LATCH COMMAND

The second method uses the "Counter Latch Command". Like a Control Word, this command is written to the Control Word Register, which is selected when $A_1,A_0 = 11$. Also like a Control Word, the SC0, SC1 bits select one of the three Counters, but two other bits, D5 and D4, distinguish this command from a Control Word.

$A_1,A_0 = 11$; CS = 0; RD = 1; WR = 0

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC1	SC0	0	0	X	X	X	X

SC1,SC0—specify counter to be latched

SC1	SC0	Counter
0	0	0
0	1	1
1	0	2
1	1	Read-Back Command

D5,D4—00 designates Counter Latch Command

X—don't care

NOTE:
Don't care bits (X) should be 0 to insure compatibility with future Intel products.

Figure 9. Counter Latching Command Format

The selected Counter's output latch (OL) latches the count at the time the Counter Latch Command is received. This count is held in the latch until it is read by the CPU (or until the Counter is reprogrammed). The count is then unlatched automatically and the OL returns to "following" the counting element (CE). This allows reading the contents of the Counters "on the fly" without affecting counting in progress. Multiple Counter Latch Commands may be used to latch more than one Counter. Each latched Counter's OL holds its count until it is read. Counter Latch Commands do not affect the programmed Mode of the Counter in any way.

If a Counter is latched and then, some time later, latched again before the count is read, the second Counter Latch Command is ignored. The count read will be the count at the time the first Counter Latch Command was issued.

With either method, the count must be read according to the programmed format; specifically, if the Counter is programmed for two byte counts, two bytes must be read. The two bytes do not have to be read one right after the other; read or write or programming operations of other Counters may be inserted between them.

Another feature of the 8254 is that reads and writes of the same Counter may be interleaved; for example, if the Counter is programmed for two byte counts, the following sequence is valid.

1) Read least significant byte.
2) Write new least significant byte.
3) Read most significant byte.
4) Write new most significant byte.

If a Counter is programmed to read/write two-byte counts, the following precaution applies: A program must not transfer control between reading the first and second byte to another routine which also reads from that same Counter. Otherwise, an incorrect count will be read.

READ-BACK COMMAND

The third method uses the Read-Back Command. This command allows the user to check the count value, programmed Mode, and current states of the OUT pin and Null Count flag of the selected counter(s).

The command is written into the Control Word Register and has the format shown in Figure 10. The command applies to the counters selected by setting their corresponding bits D3, D2, D1 = 1.

The read-back command may be used to latch multiple counter output latches (OL) by setting the $\overline{\text{COUNT}}$ bit D5 = 0 and selecting the desired counter(s). This single command is functionally equivalent to several counter latch commands, one for each counter latched. Each counter's latched count is held until it is read (or the counter is reprogrammed). The counter is automatically unlatched when read, but other counters remain latched until they are read. If multiple count read-back commands are issued to the same counter without reading the count, all but the first are ignored; i.e., the count which will be read is the count at the time the first read-back command was issued.

A0, A1 = 11 $\quad \overline{CS} = 0 \quad \overline{RD} = 1 \quad \overline{WR} = 0$

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	1	COUNT	STATUS	CNT 2	CNT 1	CNT 0	0

D_5: 0 = Latch count of selected counter(s)
D_4: 0 = Latch status of selected counters(s)
D_3: 1 = Select Counter 2
D_2: 1 = Select Counter 1
D_1: 1 = Select Counter 0
D_0: Reserved for future expansion; Must be 0

Figure 10. Read-Back Command Format

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Output	Null Count	RW1	RW0	M2	M1	M0	BCD

D_7 1 = OUT Pin is 1
0 = OUT Pin is 0

D_6 1 = Null Count
0 = Count available for reading

D_5–D_0 Counter programmed mode (see Figure 7)

Figure 11. Status Byte

The counter status format is shown in Figure 11. Bits D5 through D0 contain the counter's programmed Mode exactly as written in the last Mode Control Word. OUTPUT bit D7 contains the current state of the OUT pin. This allows the user to monitor the counter's output via software, possibly eliminating some hardware from a system.

NULL COUNT bit D6 indicates when the last count written to the counter register (CR) has been loaded into the counting element (CE). The exact time this happens depends on the Mode of the counter and is described in the Mode Definitions, but until the count is loaded into the counting element (CE), it can't be read from the counter. If the count is latched or read before this time, the count value will not reflect the new count just written. The operation of Null Count is shown in Figure 12.

If multiple status latch operations of the counter(s) are performed without reading the status, all but the first are ignored; i.e., the status that will be read is the status of the counter at the time the first status read-back command was issued.

Both count and status of the selected counter(s) may be latched simultaneously by setting both

This Action	Causes
A. Write to the control word register;(1)	Null Count = 1
B. Write to the count register (CR);(2)	Null Count = 1
C. New Count is loaded into CE (CR → CE);	Null Count = 0

NOTE:
1. Only the counter specified by the control word will have its Null Count set to 1. Null count bits of other counters are unaffected.
2. If the counter is programmed for two-byte counts (least significant byte then most significant byte) Null Count goes to 1 when the second byte is written.

Figure 12. Null Count Operation

Command D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Description	Result
1	1	0	0	0	0	1	0	Read back count and status of Counter 0	Count and status latched for Counter 0
1	1	1	0	0	1	0	0	Read back status of Counter 1	Status latched for Counter 1
1	1	1	0	1	1	0	0	Read back status of Counters 2, 1	Status latched for Counter 2, but not Counter 1
1	1	0	1	1	0	0	0	Read back count of Counter 2	Count latched for Counter 2
1	1	0	0	0	1	0	0	Read back count and status of Counter 1	Count latched for Counter 1, but not status
1	1	1	0	0	0	1	0	Read back status of Counter 1	Command ignored, status already latched for Counter 1

Figure 13. Read-Back Command Example

tionally the same as issuing two separate read-back commands at once, and the above discussions apply here also. Specifically, if multiple count and/or status read-back commands are issued to the same counter(s) without any intervening reads, all but the first are ignored. This is illustrated in Figure 13.

If both count and status of a counter are latched, the first read operation of that counter will return latched status, regardless of which was latched first. The next one or two reads (depending on whether the counter is programmed for one or two type counts) return latched count. Subsequent reads return unlatched count.

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A_1	A_0	
0	1	0	0	0	Write into Counter 0
0	1	0	0	1	Write into Counter 1
0	1	0	1	0	Write into Counter 2
0	1	0	1	1	Write Control Word
0	0	1	0	0	Read from Counter 0
0	0	1	0	1	Read from Counter 1
0	0	1	1	0	Read from Counter 2
0	0	1	1	1	No-Operation (3-State)
1	X	X	X	X	No-Operation (3-State)
0	1	1	X	X	No-Operation (3-State)

Figure 14. Read/Write Operations Summary

Mode Definitions

The following are defined for use in describing the operation of the 8254.

CLK Pulse: a rising edge, then a falling edge, in that order, of a Counter's CLK input.

Trigger: a rising edge of a Counter's GATE input.

Counter loading: the transfer of a count from the CR to the CE (refer to the "Functional Description")

MODE 0: INTERRUPT ON TERMINAL COUNT

Mode 0 is typically used for event counting. After the Control Word is written, OUT is initially low, and will remain low until the Counter reaches zero. OUT then goes high and remains high until a new count or a new Mode 0 Control Word is written into the Counter.

GATE = 1 enables counting; GATE = 0 disables counting. GATE has no effect on OUT.

After the Control Word and initial count are written to a Counter, the initial count will be loaded on the next CLK pulse. This CLK pulse does not decrement the count, so for an initial count of N, OUT does not go high until N + 1 CLK pulses after the initial count is written.

If a new count is written to the Counter, it will be loaded on the next CLK pulse and counting will continue from the new count. If a two-byte count is written, the following happens:

1) Writing the first byte disables counting. OUT is set low immediately (no clock pulse required)
2) Writing the second byte allows the new count to be loaded on the next CLK pulse.

This allows the counting sequence to be synchronized by software. Again, OUT does not go high until N+1 CLK pulses after the new count of N is written.

If an initial count is written while GATE = 0, it will still be loaded on the next CLK pulse. When GATE goes high, OUT will go high N CLK pulses later; no CLK pulse is needed to load the Counter as this has already been done.

MODE 1: HARDWARE RETRIGGERABLE ONE-SHOT

OUT will be initially high. OUT will go low on the CLK pulse following a trigger to begin the one-shot pulse, and will remain low until the Counter reaches zero. OUT will then go high and remain high until the CLK pulse after the next trigger.

After writing the Control Word and initial count, the Counter is armed. A trigger results in loading the Counter and setting OUT low on the next CLK pulse, thus starting the one-shot pulse. An initial count of N will result in a one-shot pulse N CLK cycles in duration. The one-shot is retriggerable, hence OUT will remain low for N CLK pulses after any trigger. The one-shot pulse can be repeated without rewriting the same count into the counter. GATE has no effect on OUT.

If a new count is written to the Counter during a one-shot pulse, the current one-shot is not affected unless the counter is retriggered. In that case, the Counter is loaded with the new count and the one-shot pulse continues until the new count expires.

MODE 2: RATE GENERATOR

This Mode functions like a divide-by-N counter. It is typically used to generate a Real Time Clock interrupt. OUT will initially be high. When the initial count has decremented to 1, OUT goes low for one CLK pulse. OUT then goes high again, the Counter reloads the initial count and the process is repeated. Mode 2 is periodic; the same sequence is repeated indefinitely. For an initial count of N, the sequence repeats every N CLK cycles.

GATE = 1 enables counting; GATE = 0 disables counting. If GATE goes low during an output pulse, OUT is set high immediately. A trigger reloads the Counter with the initial count on the next CLK pulse; OUT goes low N CLK pulses after the trigger. Thus the GATE input can be used to synchronize the Counter.

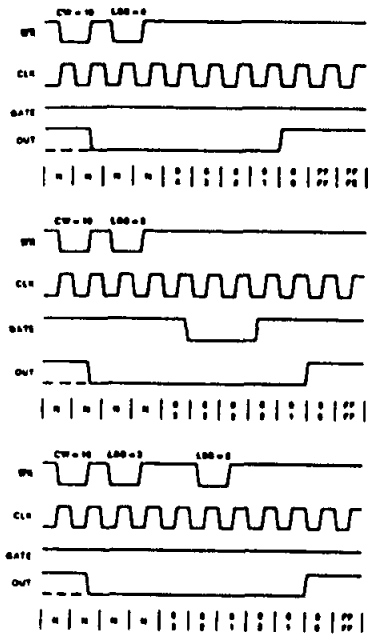
After writing a Control Word and initial count, the Counter will be loaded on the next CLK pulse. OUT goes low N CLK Pulses after the initial count is written. This allows the Counter to be synchronized by software also.

Writing a new count while counting does not affect the current counting sequence. If a trigger is received after writing a new count but before the end of the current period, the Counter will be loaded with the new count on the next CLK pulse and counting will continue from the new count. Otherwise, the new count will be loaded at the end of the current counting cycle. In mode 2, a COUNT of 1 is illegal.

MODE 3: SQUARE WAVE MODE

Mode 3 is typically used for Baud rate generation. Mode 3 is similar to Mode 2 except for the duty cycle of OUT. OUT will initially be high. When half the initial count has expired, OUT goes low for the remainder of the count. Mode 3 is periodic; the sequence above is repeated indefinitely. An initial count of N results in a square wave with a period of N CLK cycles.

GATE = 1 enables counting; GATE = 0 disables counting. If GATE goes low while OUT is low, OUT is



NOTE:
The Following Conventions Apply To All Mode Timing Diagrams:
1. Counters are programmed for binary (not BCD) counting and for Reading/Writing least significant byte (LSB) only.
2. The counter is always selected ($\overline{CS}$ always low).
3. CW stands for "Control Word"; CW = 10 means a control word of 10, hex is written to the counter.
4. LSB stands for "Least Significant Byte" of count.
5. Numbers below diagrams are count values.
The lower number is the least significant byte.
The upper number is the most significant byte. Since the counter is programmed to Read/Write LSB only, the most significant byte cannot be read.
N stands for an undefined count.
Vertical lines show transitions between count values.

Figure 15. Mode 0

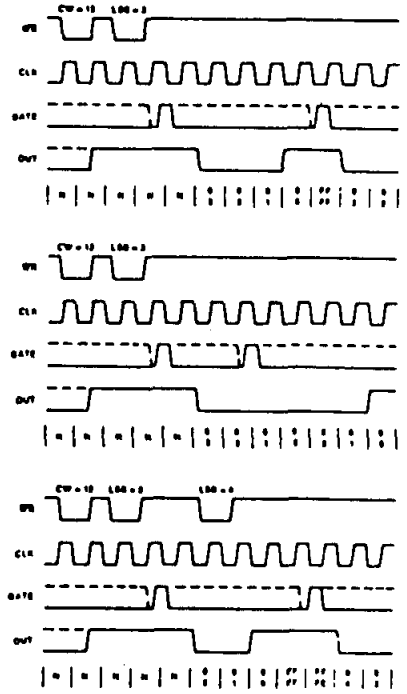


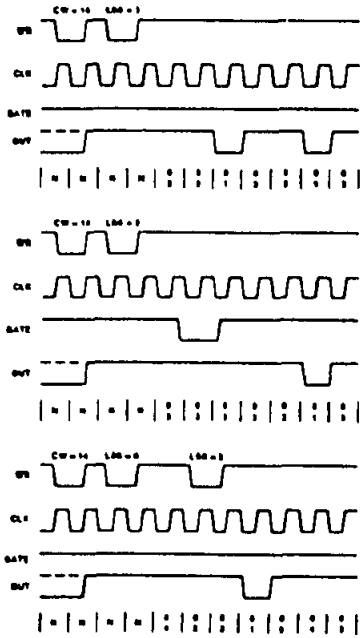
Figure 16. Mode 1

set high immediately; no CLK pulse is required. A trigger reloads the Counter with the initial count on the next CLK pulse. Thus the GATE input can be used to synchronize the Counter.
After writing a Control Word and initial count, the Counter will be loaded on the next CLK pulse. This allows the Counter to be synchronized by software also.

Writing a new count while counting does not affect the current counting sequence. If a trigger is received after writing a new count but before the end of the current half-cycle of the square wave, the Counter will be loaded with the new count on the next CLK pulse and counting will continue from the new count. Otherwise, the new count will be loaded at the end of the current half-cycle.

Mode 3 is implemented as follows:

Even counts: OUT is initially high. The initial count is loaded on one CLK pulse and then is decremented by two on succeeding CLK pulses. When the count expires OUT changes value and the Counter is reloaded with the initial count. The above process is repeated indefinitely.



NOTE:
A GATE transition should not occur one clock prior to terminal count.

Figure 17. Mode 2

Odd counts: OUT is initially high. The initial count minus one (an even number) is loaded on one CLK pulse and then is decremented by two on succeeding CLK pulses. One CLK pulse *after* the count expires, OUT goes low and the Counter is reloaded with the initial count minus one. Succeeding CLK pulses decrement the count by two. When the count expires, OUT goes high again and the Counter is reloaded with the initial count minus one. The above process is repeated indefinitely. So for odd counts, OUT will be high for (N + 1)/2 counts and low for (N − 1)/2 counts.

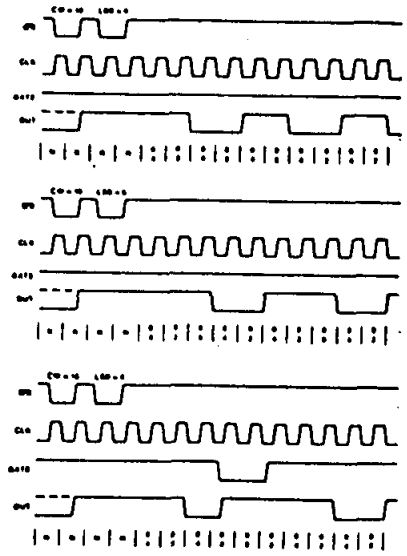
MODE 4: SOFTWARE TRIGGERED STROBE

OUT will be initially high. When the initial count expires, OUT will go low for one CLK pulse and then go high again. The counting sequence is "triggered" by writing the initial count.

GATE = 1 enables counting; GATE = 0 disables counting. GATE has no effect on OUT.

After writing a Control Word and initial count, the Counter will be loaded on the next CLK pulse. This CLK pulse does not decrement the count, so for an initial count of N, OUT does not strobe low until N + 1 CLK pulses after the initial count is written.

If a new count is written during counting, it will be



NOTE:
A GATE transition should not occur one clock prior to terminal count.

Figure 18. Mode 3

loaded on the next CLK pulse and counting will continue from the new count. If a two-byte count is written, the following happens:

1) Writing the first byte has no effect on counting.
2) Writing the second byte allows the new count to be loaded on the next CLK pulse.

This allows the sequence to be "retriggered" by software. OUT strobes low N + 1 CLK pulses after the new count of N is written.

MODE 5: HARDWARE TRIGGERED STROBE (RETRIGGERABLE)

OUT will initially be high. Counting is triggered by a rising edge of GATE. When the initial count has expired, OUT will go low for one CLK pulse and then go high again.

After writing the Control Word and initial count, the counter will not be loaded until the CLK pulse after a trigger. This CLK pulse does not decrement the count, so for an initial count of N, OUT does not strobe low until N + 1 CLK pulses after a trigger.

A trigger results in the Counter being loaded with the initial count on the next CLK pulse. The counting sequence is retriggerable. OUT will not strobe low

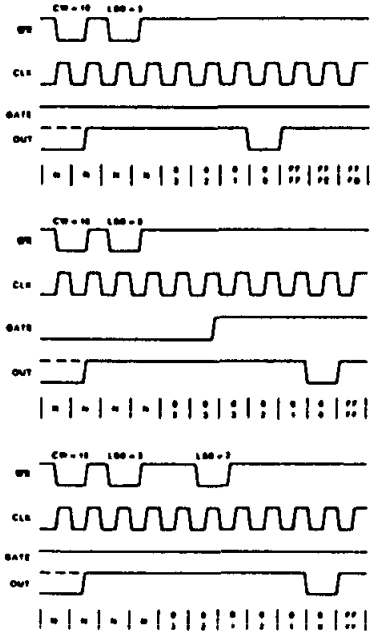


Figure 19. Mode 4

for N + 1 CLK pulses after any trigger. GATE has no effect on OUT.

If a new count is written during counting, the current counting sequence will not be affected. If a trigger occurs after the new count is written but before the current count expires, the Counter will be loaded with the new count on the next CLK pulse and counting will continue from there.

Operation Common to All Modes

PROGRAMMING

When a Control Word is written to a Counter, all Control Logic is immediately reset and OUT goes to a known initial state; no CLK pulses are required for this.

GATE

The GATE input is always sampled on the rising edge of CLK. In Modes 0, 2, 3, and 4 the GATE input is level sensitive, and the logic level is sampled on the rising edge of CLK. In Modes 1, 2, 3, and 5 the GATE input is rising-edge sensitive. In these Modes, a rising edge of GATE (trigger) sets an edge-sensi-

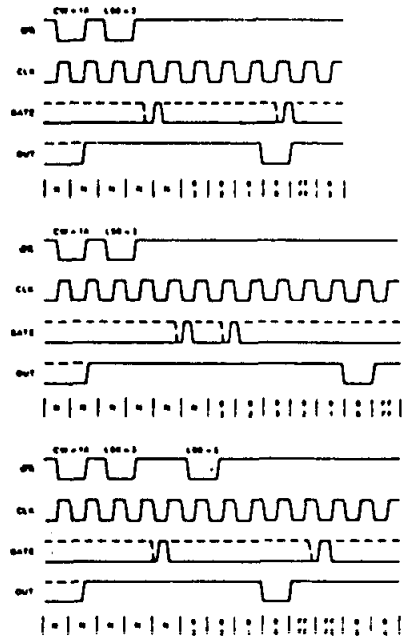


Figure 20. Mode 5

Signal Status Modes	Low Or Going Low	Rising	High
0	Disables Counting	— —	Enables Counting
1	— —	1) Initiates Counting 2) Resets Output after Next Clock	— —
2	1) Disables Counting 2) Sets Output Immediately High	Initiates Counting	Enables Counting
3	1) Disables Counting 2) Sets Output Immediately High	Initiates Counting	Enables Counting
4	Disables Counting	— —	Enables Counting
5	— —	Initiates Counting	— —

Figure 21. Gate Pin Operations Summary

Mode	Min Count	Max Count
0	1	0
1	1	0
2	2	0
3	2	0
4	1	0
5	1	0

NOTE:
0 is equivalent to 2^{16} for binary counting and 10^4 for BCD counting.

Figure 22. Minimum and Maximum Initial Counts

tive flip-flop in the Counter. This flip-flop is then sampled on the next rising edge of CLK; the flip-flop is reset immediately after it is sampled. In this way, a trigger will be detected no matter when it occurs—a high logic level does not have to be maintained until the next rising edge of CLK. Note that in Modes 2 and 3, the GATE input is both edge- and level-sensitive. In Modes 2 and 3, if a CLK source other than the system clock is used, GATE should be pulsed immediately following $\overline{WR}$ of a new count value.

COUNTER

New counts are loaded and Counters are decremented on the falling edge of CLK.
The largest possible initial count is 0; this is equivalent to 2^{16} for binary counting and 10^4 for BCD counting.
The Counter does not stop when it reaches zero. In Modes 0, 1, 4, and 5 the Counter "wraps around" to the highest count, either FFFF hex for binary counting or 9999 for BCD counting, and continues counting. Modes 2 and 3 are periodic; the Counter reloads itself with the initial count and continues counting from there.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias .....0°C to 70°C
Storage Temperature .....−65°C to +150°C
Voltage on Any Pin with
Respect to Ground.....−0.5V to +7V
Power Dissipation 1W

Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5V ± 10%

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage	−0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC} + 0.5V	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 2.0 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = −400 μA
I_{IL}	Input Load Current		± 10	μA	V_{IN} = V_{CC} to 0V
I_{OFL}	Output Float Leakage		± 10	μA	V_{OUT} = V_{CC} to 0.45V
I_{CC}	V_{CC} Supply Current		170	mA	
C_{IN}	Input Capacitance		10	pF	f_c = 1 MHz
$C_{I/O}$	I/O Capacitance		20	pF	Unmeasured pins returned to V_{SS}(4)

A.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5V ± 10%, GND = 0V

Bus Parameters(1)

READ CYCLE

Symbol	Parameter	8254-5		8254		8254-2		Unit
		Min	Max	Min	Max	Min	Max	
t_{AR}	Address Stable Before $\overline{RD}$ ↓	45		45		30		ns
t_{SR}	$\overline{CS}$ Stable Before $\overline{RD}$ ↓	0		0		0		ns
t_{RA}	Address Hold Time After $\overline{RD}$ ↑	0		0		0		ns
t_{RR}	$\overline{RD}$ Pulse Width	150		150		95		ns
t_{RD}	Data Delay from $\overline{RD}$ ↓		120		120		85	ns
t_{AD}	Data Delay from Address		220		220		185	ns
t_{DF}	$\overline{RD}$ ↑ to Data Floating	5	90	5	90	5	65	ns
t_{RV}	Command Recovery Time	200		200		165		ns

NOTE:
1. AC timings measured at V_{OH} = 2.0V, V_{OL} = 0.8V.

A.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5V ± 10%, GND = 0V (Continued)

WRITE CYCLE

Symbol	Parameter	8254-5		8254		8254-2		Unit
		Min	Max	Min	Max	Min	Max	
t_{AW}	Address Stable Before $\overline{WR}$ ↓	0		0		0		ns
t_{SW}	$\overline{CS}$ Stable Before $\overline{WR}$ ↓	0		0		0		ns
t_{WA}	Address Hold Time After $\overline{WR}$ ↓	0		0		0		ns
t_{WW}	$\overline{WR}$ Pulse Width	150		150		95		ns
t_{DW}	Data Setup Time Before $\overline{WR}$ ↑	120		120		95		ns
t_{WD}	Data Hold Time After $\overline{WR}$ ↑	0		0		0		ns
t_{RV}	Command Recovery Time	200		200		165		ns

CLOCK AND GATE

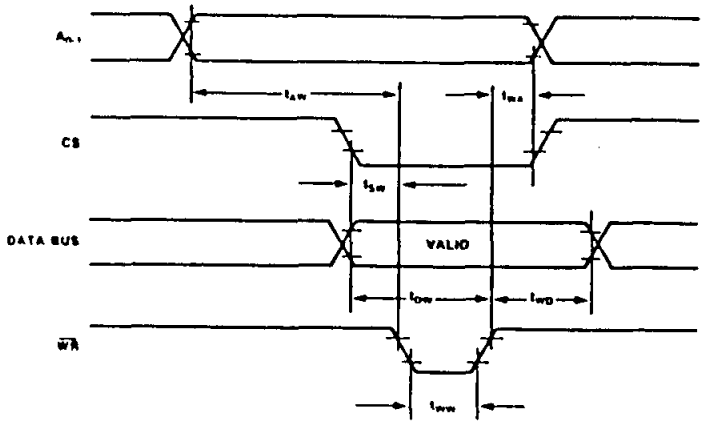
Symbol	Parameter	8254-5		8254		8254-2		Unit
		Min	Max	Min	Max	Min	Max	
t_{CLK}	Clock Period	200	DC	125	DC	100	DC	ns
t_{PWH}	High Pulse Width	60(3)		60(3)		30(3)		ns
t_{PWL}	Low Pulse Width	60(3)		60(3)		50(3)		ns
t_R	Clock Rise Time		25		25		25	ns
t_F	Clock Fall Time		25		25		25	ns
t_{GW}	Gate Width High	50		50		50		ns
t_{GL}	Gate Width Low	50		50		50		ns

t_{GS}	Gate Setup Time to CLK ↑	50		50		40		ns
t_{GH}	Gate Setup Time After CLK ↑	50(2)		50(2)		50(2)		ns
t_{OD}	Output Delay from CLK ↓		150		150		100	ns
t_{ODG}	Output Delay from Gate ↓		120		120		100	ns
t_{WC}	CLK Delay for Loading ↓	0	55	0	55	0	55	ns
t_{WG}	Gate Delay for Sampling	−5	50	−5	50	−5	40	ns
t_{WO}	OUT Delay from Mode Write		260		260		240	ns
t_{CL}	CLK Set Up for Count Latch	−40	45	−40	45	−40	40	ns

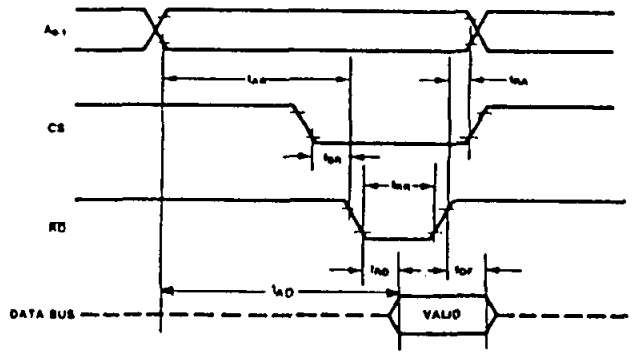
NOTES:
2. In Modes 1 and 5 triggers are sampled on each rising clock edge. A second trigger within 120 ns (70 ns for the 8254-2) of the rising clock edge may not be detected.
3. Low-going glitches that violate t_{PWH}, t_{PWL} may cause errors requiring counter reprogramming.
4. Sampled, not 100% tested. T_A = 25°C.
5. If CLK present at TWC min then Count equals N + 2 CLK pulses, TWC max equals Count N + 1 CLK pulse. TWC min to TWC max, count will be either N + 1 or N + 2 CLK pulses.
6. In Modes 1 and 5, if GATE is present when writing a new Count value, at TWG min Counter will not be triggered, at TWG max Counter will be triggered.
7. If CLK present when writing a Counter Latch or ReadBack Command, at TCL min CLK will be reflected in count value latched, at TCL max CLK will not be reflected in the count value latched.

WAVEFORMS

WRITE

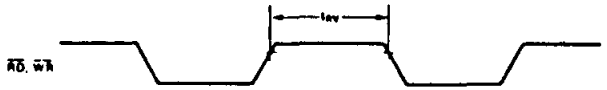


READ

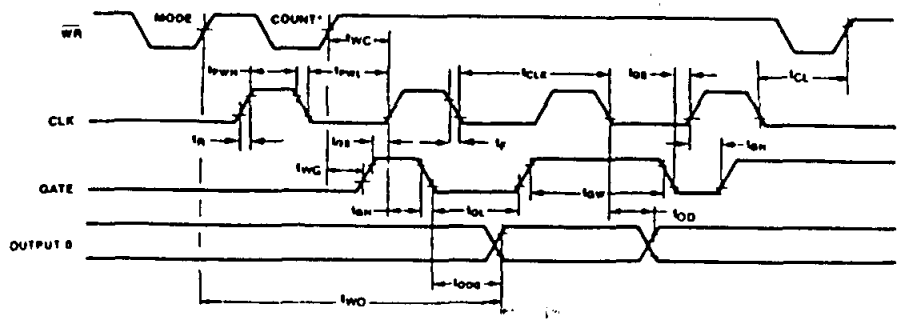


WAVEFORMS (Continued)

RECOVERY



CLOCK AND GATE

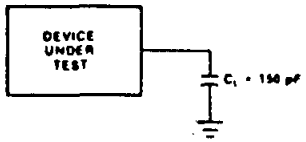


*Last byte of count being written.

A.C. TESTING INPUT, OUTPUT WAVEFORM

A.C. Testing Inputs are driven at 2.4V for a Logic "1" and 0.45V for a Logic "0". Timing measurements are made at 2.0V for a Logic "1" and 0.8V for a Logic "0".

A.C. TESTING LOAD CIRCUIT



C_L = 150 pF
C_L includes Jig Capacitance

DAFTAR RIWAYAT HIDUP

Nama	:	**Erwin Johanes**
NIRM	:	96.7.003.31073.58594
Tempat, Tanggal. Lahir	:	Surabaya, 16 Oktober 1977
Agama	:	Katolik
Alamat	:	Ciliwung 27, Surabaya

Riwayat Pendidikan :

- SD Katolik Santa Clara Surabaya tahun 1984 – 1990
- SMP Katolik Santa Clara Surabaya tahun 1990 – 1993
- SMA Katolik St. Louis I Surabaya tahun 1993 - 1996
- Universitas Katolik Widya Mandala Fakultas Teknik Jurusan Teknik Elektro tahun 1999

Selama Kuliah Aktif Sebagai :

- Asisten Laboratorium Komputer
- Asisten Praktikum Sistem Instrumentasi Elektronika