de KITS Application Note

AN62 – Pengaman Telepon Digital

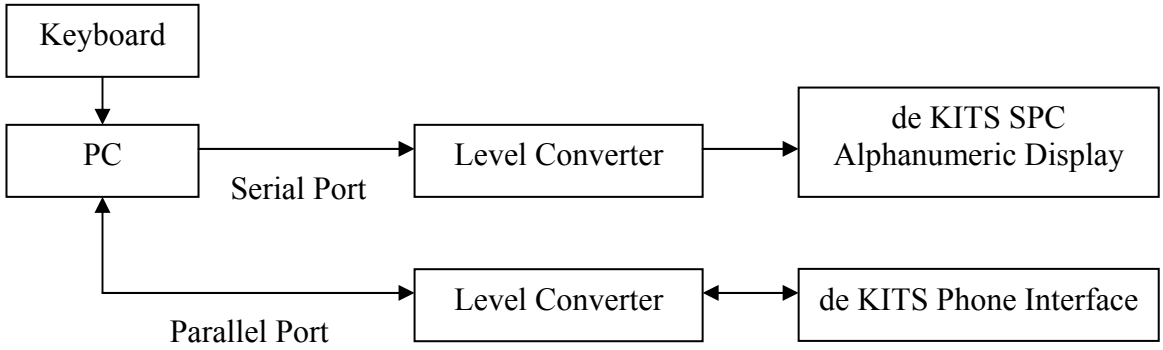
Oleh: Tim IE & Johanes (Universitas Kristen Petra)

AN ini menjelaskan tentang penggunaan de KITS Phone Interface dengan PC dengan antarmuka parallel port disertai tampilan de KITS SPC Alphanumeric Display dengan antarmuka COM port, dengan menggunakan Phone Interface Driver Delphi Component. AN ini memiliki fitur sederhana untuk memblokir nomor telepon dengan awalan tertentu yang dapat diprogram.

Modul-modul yang digunakan adalah:
- 1 de KITS Phone Interface
- 2 de KITS SPC Alphanumeric

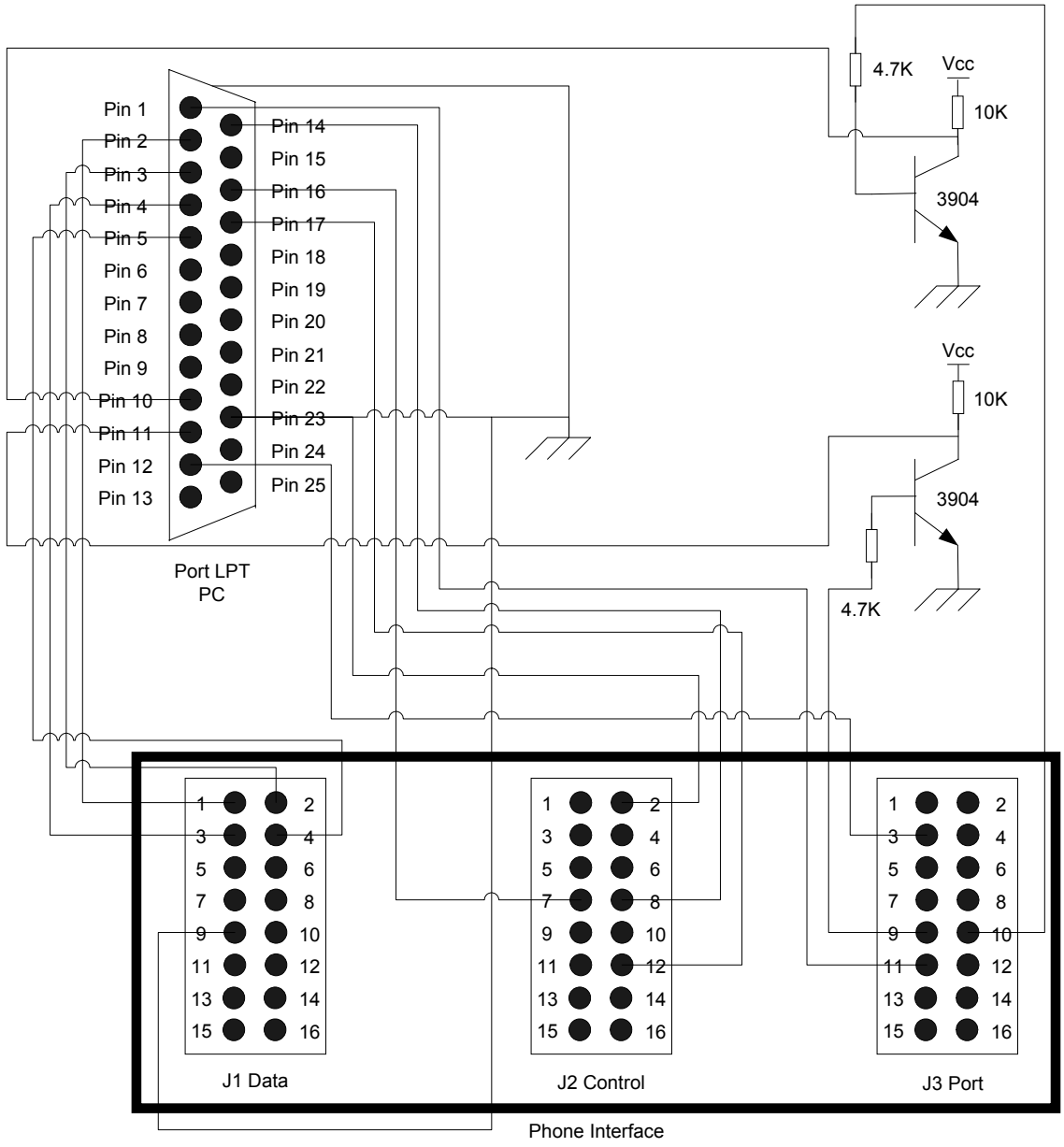
Hardware lain yang diperlukan:
- 1 Unit Personal Computer yang setara dengan Pentium III 600Mhz atau lebih tinggi dengan memori minimal 128 MB
- Rangkaian *level converter* untuk de KITS SPC Alphanumeric Display
- Rangkaian *level converter* untuk de KITS Phone Interface

Adapun blok diagram sistem secara keseluruhan adalah sebagai berikut:



Gambar 1
Blok Diagram Keseluruhan

Hubungan komputer dengan de KITS Phone Interface serta rangkaian Level Converter adalah sebagai berikut:



Gambar 2
Hubungan Komputer dengan de KITS Phone Interface

Komponen yang dibutuhkan adalah:

- 1 konektor DB25 Female + cover
- 2 Resistor 10 K Ohm
- 2 Resistor 4K7 Ohm
- 2 Transistor 2N3904
- Kabel parallel dengan panjang secukupnya
- 3 buah kabel ampenol 16 pin dengan panjang secukupnya
- Soket Ampenol 16 pin 3 Buah
- PCB Lubang untuk membuat rangkaian

Penjelasan hubungan komputer dengan de KITS Phone Interface beserta fungsi pin-pin yang dihubungkan adalah:

Input/Output data bus Phone Interface:

- J1.1 ←→ DB25.2 (Databus.0)
- J1.2 ←→ DB25.3 (Databus.1)
- J1.3 ←→ DB25.4 (Databus.2)
- J1.4 ←→ DB25.5 (Databus.3)

Input dari Phone Interface:

- J3.10 ← Inverter (3904) → DB25.10 (HK to Base+1.6)
- J3.9 ← Inverter (3904)→ DB25.11 (RI to Base+1.7)
- J2.3 ←→ DB25.12 (IRQ/CP to Base+1.5)

Output ke Phone Interface:

- J3.11 ←→ DB25.1 (RE to Base+2.0)
- J2.8 ←→ DB25.14 (RD to Base+2.1)
- J2.7 ←→ DB25.16 (WR to Base+2.2)
- J3.12 ←→ DB25.17 (RS0 to Base+2.3)

Koneksi Ground:

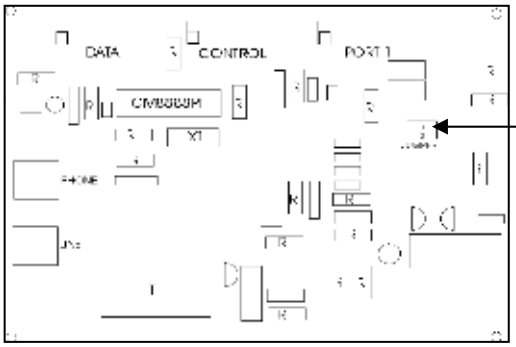
- J2.2 ←→ DB25.23 (GND Phone Interface to GND Parallel Port)
- J1.9 ←→ DB25.23 (CS to GND)

de KITS Phone Interface memiliki 2 mode yang bisa diatur melalui jumper:
Mode 1:

- Jumper mode tidak terhubung.
- De KITS Phone Interface bekerja dengan bantuan pesawat telepon untuk melakukan off hook.
- Mode ini cocok untuk aplikasi antara lain: pengaman telepon dan anti interlokal.
- Untuk melakukan pulse dialing, jumper harus berada pada mode 1.

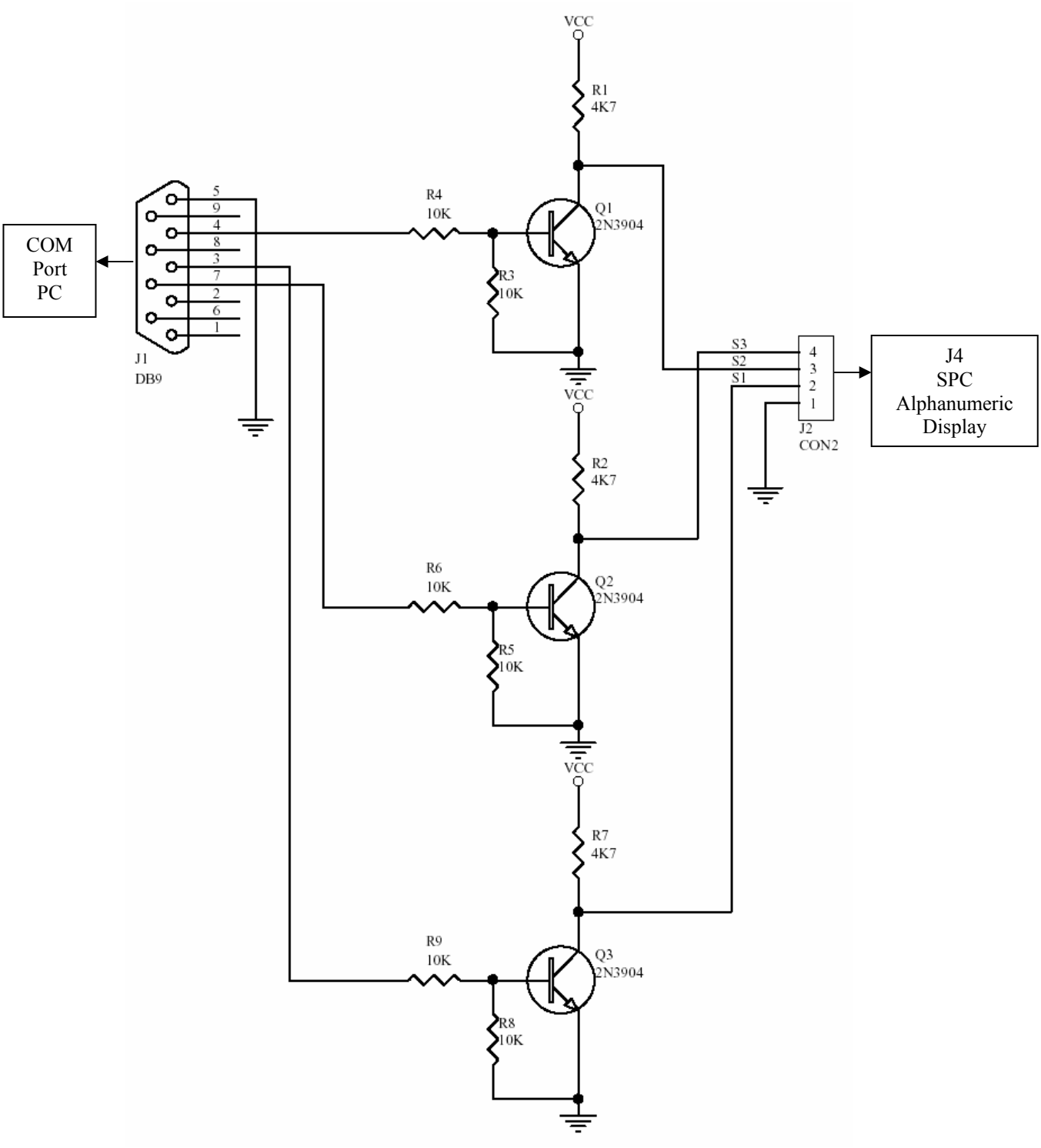
Mode 2:

- Jumper mode terhubung.
- De KITS Phone Interface dapat bekerja dengan atau tanpa pesawat telepon untuk melakukan off hook.
- Mode ini cocok untuk aplikasi antara lain: home automation dan home security.
- Mode ini sifatnya lebih general dibandingkan dengan mode 1.



Gambar 3
Posisi Jumper de KITS Phone Interface

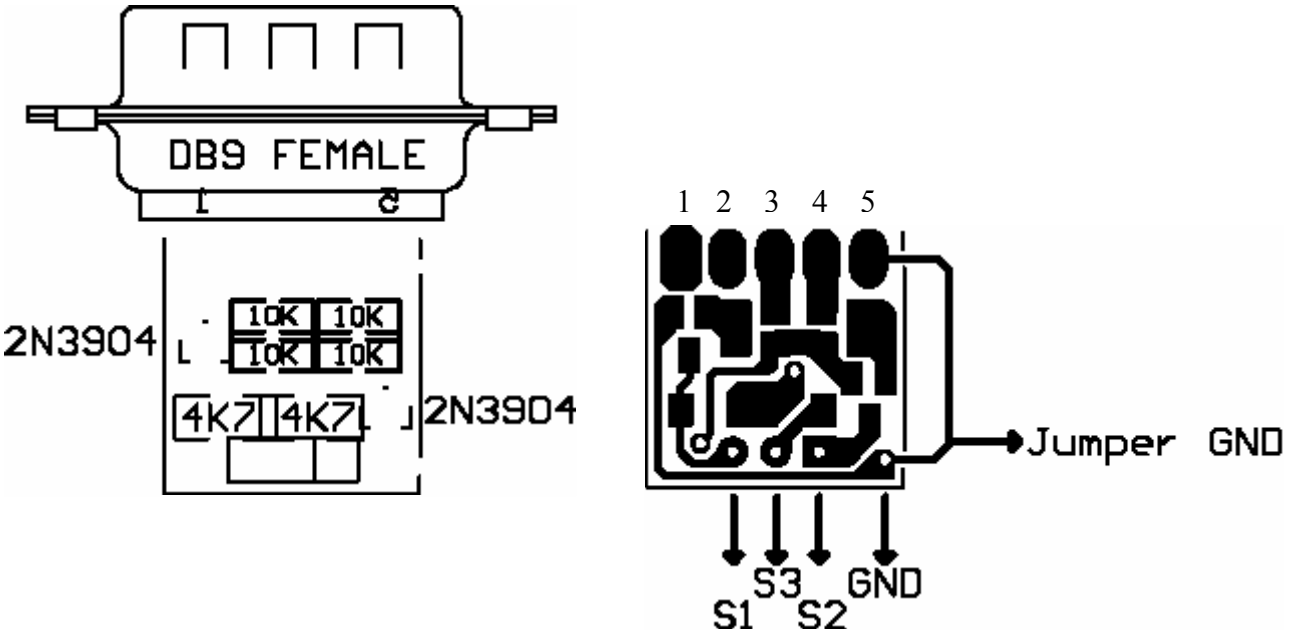
Hubungan Komputer dengan SPC Alphanumeric Display serta rangkaian Level Converter adalah sebagai berikut:



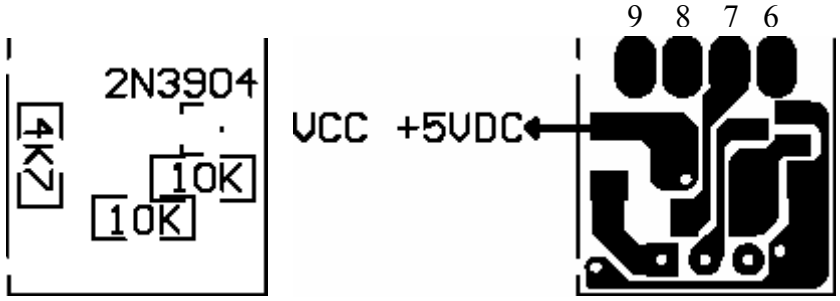
Gambar 4
Hubungan Komputer dengan de KITS SPC Alphanumeric Display

Komponen yang dibutuhkan adalah:

- 1 konektor DB9 Female + cover
- 6 Resistor 10 K Ohm SMD
- 3 Resistor 4K7 Ohm SMD
- 3 Transistor 2N3904 SMD
- Kabel Serial isi 5 dengan panjang secukupnya
- Black Housing isi 4 lengkap dengan pin



Gambar 3
Tata Letak dan Gambar Bidang PCB Bagian Atas



Gambar 5
Tata Letak dan Gambar Bidang PCB Bagian Bawah

Gambar 6
Hasil Jadi Kabel Serial

Karena de KITS SPC Alphanumeric Display tidak memiliki konektor VCC pada J4 (jalur *interface* Synchronous Serial) maka jalur VCC pada *level converter* akan dipisah dari kabel yang lain untuk dihubungkan ke VCC SPC Alphanumeric Display (seperti tampak pada gambar 5).

Patut diperhatikan bahwa semua komponen menggunakan komponen SMD. Hal ini bertujuan agar *level converter* dapat dimasukkan ke dalam cover DB9.

Aturlah *jumper* SPC Alphanumeric Display pada alamat 6 dan 7, juga pasanglah kabel serial tersebut pada Serial Port COM1 pada PC, karena sample program yang tersedia menggunakan setting tersebut.

Hal berikutnya yang harus dilakukan adalah melakukan instalasi *driver*. Langkah-langkah instalasi **Phone Interface Driver Delphi Component** adalah sebagai berikut:

- *Driver* tersebut telah diuji dapat berjalan dengan baik pada Delphi versi 5 dan sistem operasi Windows 98 dan Windows 2000, namun kemungkinan besar dapat berjalan dengan baik pula pada versi Delphi maupun sistem operasi yang lain.
- Buka **Driver Component.ZIP**, ekstrak semua isinya pada sebuah *folder*.
- Pada IDE Delphi, pilih menu **File** → **Open**.
- Akan tampil dialog **Open**. Pada **File of type**, pilih **Delphi package (*.dpk)**.
- Pilih **PIPackage.dpk** pada *folder* yang berisi *driver component* lalu klik **Open**.
- Akan tampil dialog **Package** – **PIPackage.dpk**. Klik tombol **Install** pada *toolbar dialog*.
- Bila telah ter-*install* dengan benar maka pada *component pallete* Delphi akan tampil seperti gambar berikut ini. Komponen *TPhoneInterface* adalah komponen yang dilingkari.

Gambar 7
Phone Interface Driver Delphi Component yang telah ter-*install* (bagian yang dilingkari)

Penjelasan komponen *TPhoneInterface*:

Tabel 1. *Public* dan *Published Property TPhoneInterface*

Property	**Keterangan**
PerformCPDetection	Mengaktifkan proses deteksi *call progress*. Bila bernilai TRUE maka *driver* akan melakukan deteksi *call progress*, proses ini akan menyebabkan *event* OnCallProgressDialTone, OnCallProgressBusy, OnCallProgressRingBack dipanggil sesuai dengan kondisi jalur telepon. Proses deteksi akan mengaktifkan *thread* tambahan yang otomatis dibuat oleh *driver* dan akan menggunakan *CPU Resource* yang cukup banyak. Bila tidak diperlukan sebaiknya di-a*ssign* dengan nilai FALSE.
In_IRQ (ReadOnly)	Merepresentasikan pin *interrupt request* IC8888 pada de KITS Phone Interface. Bila TRUE, berarti ada *interrupt*. Bila FALSE, berarti tidak ada *interrupt*. *Interrupt* dapat dibersihkan dengan memanggil *method* ReadStatus. Untuk penjelasan lebih lanjut, lihat *datasheet* IC8888.
In_RI (ReadOnly)	Merepresentasikan pin *ring exists* pada de KITS Phone Interface. Bila TRUE, berarti sedang ada *ring in / offering ring.* Bila FALSE, berarti tidak ada *ring in / offering ring*. Bila sedang ada panggilan masuk namun nada dering tidak dalam kondisi berbunyi maka pin ini tetap bernilai FALSE.
In_HK (ReadOnly)	Merepresentasikan pin *hook status* pada de KITS Phone Interface. Bila TRUE, berarti gagang telepon pada kondisi *off hook*. Bila FALSE, berarti gagang telepon dalam kondisi *on hook.*
Out_RE	Mengatur kondisi *on hook* atau *off hook*. Bila TRUE, berarti mengkondisikan *on hook*, bila FALSE, berarti mengkondisikan *off hook.*
Out_RS0	Merepresentasikan pin pengatur pemilihan *register* IC8888 yaitu RS0. Bila FALSE berarti memilih *data register.* Bila TRUE berarti memilih *status register.* Untuk penjelasan lebih lanjut, lihat *datasheet* IC8888.
Out_RD	Mengatur kondisi transmisi data. Digunakan untuk membaca *data register* maupun *status register*. Merepresentasikan pin RD pada IC8888. Pada saat proses pembacaan, harus bernilai TRUE. Untuk penjelasan lebih lanjut, lihat *datasheet* IC8888.
Out_WR	Mengatur kondisi transmisi data. Digunakan untuk menulis *data register* maupun *status register*. Merepresentasikan pin WR pada IC8888. Pada saat proses penulisan, harus bernilai TRUE. Untuk penjelasan lebih lanjut, lihat *datasheet* IC8888.
Data	Merepresentasikan 4 bit *data register* IC8888. Bila dibaca, akan mengakses langsung dari *register* IC8888 yang aktif. Bila di-*assign*, akan menulis langsung ke *register* IC8888 yang aktif.
BufData	*Buffer* 4 bit yang menyimpan data yang terakhir kali dituliskan ke atau dibaca dari register IC8888.

RingInCount	Menyatakan adanya *ring in* yang sedang berlangsung, nilainya menunjukkan berapa banyak dering yang sudah diterima. Bila bernilai 0, berarti tidak ada *ring in* yang sedang berlangsung.

Tabel 2. *Event TPhoneInterface*

Event	**Keterangan**
OnCallProgressBusy	*Event* ini dipanggil pada saat terdeteksi adanya pola *busy tone*, *event* ini juga dipanggil bila *busy tone* berhenti. Gunakan *method* CekBusy untuk mengambil status *busy tone*.
OnCallProgressDialTone	*Event* ini dipanggil pada saat status *call progress* berubah dari *high* ke *low* atau *low* ke *high*. Untuk mengambil nilai status *call progress*, gunakan *method* CekDialTone. Catatan: *Event* ini tetap dipanggil meskipun *ring back* maupun *busy tone* sedang terdeteksi. Gunakan *method* CekBusy, CekRingBack, atau CekEarlyIndicationFindRingBackOrBusy untuk membantu pendeteksian status.
OnCallProgressRingBack	*Event* ini dipanggil pada saat terdeteksi adanya pola *ring back tone*, *event* ini juga dipanggil bila *ring back tone* berhenti. Gunakan method CekRingBack untuk mengambil status *ring back tone*.
OnIn_HKChanged	*Event* ini dipanggil saat nilai pin HK de KITS Phone Interface berubah dari *low* ke *high* atau dari *high* ke *low*.
OnIn_RIChanged	*Event* ini dipanggil saat nilai pin RI de KITS Phone Interface berubah dari *low* ke *high* atau dari *high* ke *low*.
OnRingIn	*Event* ini dipanggil saat terjadi *ring in / offering ring* yang berakibat nilai RingInCount bertambah, *event* ini juga dipanggil saat RingInCount berubah menjadi 0. Disarankan penggunaan *event* ini dibantu dengan penggunaan *property* RingInCount.

Tabel 3. *Method TPhoneInterface*

Method	**Keterangan**
ReadDTMF	Deklarasi: function ReadDTMF: TPhNumber; Deskripsi: Membaca *data register* yang berisi angka yang ditekan oleh pengguna pada telepon dengan mode DTMF. Pembacaan ini valid jika *property* In_IRQ bernilai TRUE. Setelah membaca *data register*, panggil *method* ReadStatus untuk membersihkan kondisi In_IRQ. Proses ini terkait dengan pin RD dan RS0 IC8888. Untuk penjelasan lebih lanjut, lihat *datasheet* IC8888.
WriteDTMF	Deklarasi: procedure WriteDTMF(Value: TPhNumber); Deskripsi: Menulis *data register* dengan angka nomor telepon dan memerintahkan IC8888 untuk men-*generate* DTMF pada line telepon. Karena lama waktu *generate* DTMF adalah 50-100 ms, Maka panggil fungsi WinAPI Sleep(150) untuk menunggu IC8888 menyelesaikan prosesnya. Setelah memanggil fungsi Sleep, panggil ReadStatus untuk membersihkan kondisi In_IRQ. Proses ini terkait dengan pin WR dan RS0 IC8888. Untuk penjelasan lebih lanjut, lihat *datasheet* IC8888.
ReadStatus	Deklarasi: function ReadStatus: TPIState; Deskripsi: Membaca status IC8888, berisi 4 bit data dari *status register*. Proses ini terkait dengan pin RD dan RS0 IC8888. Untuk penjelasan lebih lanjut, lihat *datasheet* IC8888.
WriteStatus	Deklarasi: procedure WriteStatus(Value: TPIState);

	Deskripsi: Menulis status IC8888, argumen Value berisi 4 bit data yang dikirim ke *status register*. Proses ini terkait dengan pin WR dan RS0 IC8888. Untuk penjelasan lebih lanjut, lihat *datasheet* IC8888.
ReadDP	Deklarasi: function ReadDP: Integer; Deskripsi: *Method* ini adalah fasilitas untuk melakukan pembacaan *dialing pulse*. Penggunaannya cukup kompleks dan akan dijelaskan tersendiri dalam contoh aplikasi. Bila sebuah *dialing pulse* ditemukan dalam rentang waktu 20-150 ms ke depan, fungsi akan mengembalikan nilai 1. Bila sebuah *dialing pulse* ditemukan dalam rentang waktu <20 ms atau ada indikasi *dialing pulse* yang ditemukan tidak valid, fungsi akan mengembalikan nilai 0. Bila sebuah *dialing pulse* ditemukan dalam rentang waktu >150 ms, fungsi akan mengembalikan nilai 2.
WriteDP	Deklarasi: procedure WriteDP(Value: TPhNumber); Deskripsi: Memerintahkan de KITS Phone Interface untuk melakukan *dialing pulse*. Argumen Value berisi nilai angka yang di-*dial.*
DTMFDial	Deklarasi: procedure DTMFDial(DialNum: String); Deskripsi: Melakukan *dial* dengan DTMF terhadap sebuah nomor telepon lengkap sesuai dengan parameter string angka dalam kode ASCII, misalnya: ‘111’.
DPDial	Deklarasi: procedure DPDial(DialNum: String); Deskripsi: Melakukan *dial* dengan *pulse dialing* terhadap sebuah nomor telepon lengkap sesuai dengan parameter string angka dalam kode ASCII, misalnya: ‘111’.
Reset	Deklarasi: procedure Reset; Deskripsi: Me-*reset* kinerja phone interface, selanjutnya Anda harus memanggil *method* Init.

Init	Deklarasi:
	procedure Init(Value,Value2: TPIState);
	
	Deskripsi:
	Melakukan inisialisasi de KITS Phone Interface dengan nilai *register* A dan B yang direpresentasikan oleh argumen Value dan Value2. Gunakan *standard constant pair* yang telah tersedia seperti berikut ini:
	
	InitDTMF1Val = $D;
	InitDTMF1Val2 = $0; //TOUT,DTMF,IRQ,BURST(50ms)
	
	InitDTMF2Val = $F;
	InitDTMF2Val2 = $0; //TOUT,CP,IRQ,BURST(100ms)
	
	InitToneModeVal = $C;
	InitToneModeVal2 = $1; //not TOUT,CP,IRQ,BURST(100ms)
	
	InitCPVal = $E;
	InitCPVal2 = $0; //not TOUT,DTMF,IRQ,BURST(100ms)
	
	InitToneEnbVal = $5; //TOUT,DTMF,IRQ
	InitToneEnbVal2 = $0; //Register B not set
	
	InitToneDisVal = $4; //not TOUT,CP,IRQ
	InitToneDisVal2 = $0; //Register B not set

CekDialTone	Deklarasi: function CekDialTone: Boolean; Deskripsi: Membaca kondisi *call progress*. Bernilai TRUE bila *dial tone* terdeteksi. Bernilai FALSE bila *dial tone* tidak terdeteksi.
CekBusy	Deklarasi: function CekBusy: Boolean; Deskripsi: Membaca kondisi *busy tone*. Bernilai TRUE bila *busy tone* sedang berlangsung dan bernilai FALSE bila busy tone tidak terdeteksi.
CekRingBack	Deklarasi: function CekRingBack: Boolean; Deskripsi: Membaca kondisi *ring back tone*. Bernilai TRUE bila *ring back tone* sedang berlangsung. Bernilai FALSE bila *ring back tone* tidak terdeteksi.
CekEarlyIndicationFind RingBackOrBusy	Deklarasi: function CekEarlyIndicationFindRingBackOrBusy: Boolean; Deskripsi: Memeriksa apakah indikasi awal adanya *ring back* atau *busy tone*, terdeteksi. Saat indikasi awal sudah ditemukan, belum tentu CekRingBack atau CekBusy sudah bernilai TRUE. Fungsi ini bisa saja akan mengembalikan nilai TRUE meskipun *event* OnRingBackTone atau OnBusyTone belum dipanggil.
WaitTillRingOrOffHook	Deklarasi: function WaitTillRingOrOffHook(TimeOut: Integer): Integer; Deskripsi: Menunggu sampai In_RE atau In_HK bernilai TRUE, argumen TimeOut menyatakan dalam milidetik seberapa lama harus menunggu. Bila In_RE atau In_HK tidak bernilai TRUE sampai waktu *time out* terlampaui, fungsi ini mengembalikan nilai 0. Bila terjadi In_RE bernilai TRUE, fungsi ini mengembalikan nilai 1. Bila terjadi In_HK bernilai TRUE, fungsi ini mengembalikan nilai 2.

Variable Global unit PhoneInterface.pas	**Keterangan**
BasePortNumber	Pengaturan alamat dasar LPT, *default*-nya $378. Pengaturan ini mempengaruhi semua *instance* komponen Phone Interface yang ada dalam aplikasi.
PhoneInterfaceManager	Merupakan obyek pengendali semua Phone Interface yang di-*create*. Penggunaannya hanya untuk yang sudah mahir dan tidak dijelaskan disini.

Langkah-langkah *compiling* dan *testing sample application*:

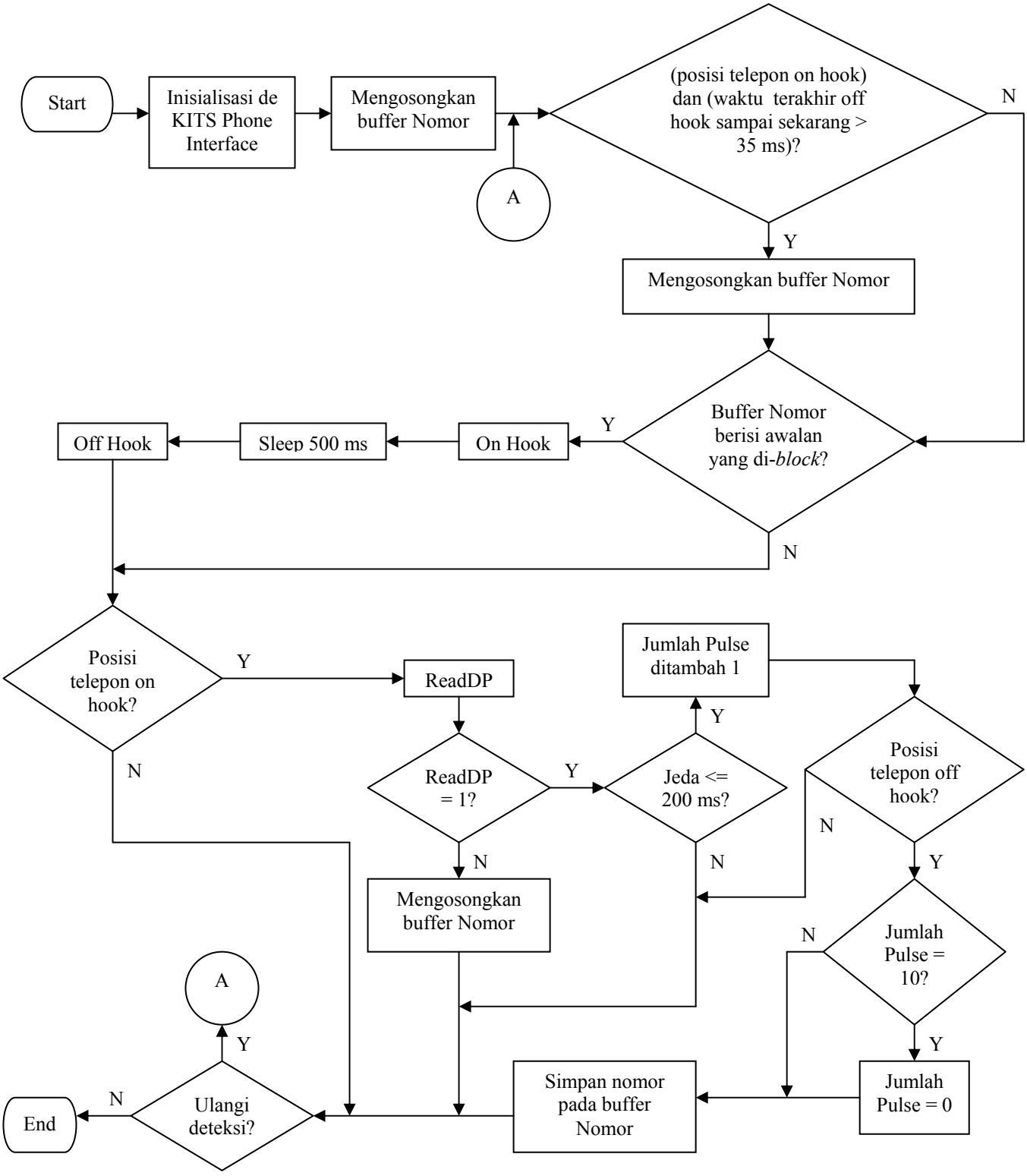
- Hubungkan semua modul sesuai penjelasan gambar 2 dan 4. Hubungkan juga sumber tegangan yang sesuai.
- Syaratnya adalah *driver component* telah ter-*install* dengan benar
- Buka **Sample Application.ZIP**, ekstrak semua isinya pada sebuah *folder*
- Pada IDE Delphi, pilih menu **File → Open Project**
- Pilih Project1.dpr pada *folder* yang berisi *sample application* lalu klik **Open**
- Setelah *file* telah di-*load* oleh Delphi, pilih menu **Project → Options**
- Tampil dialog **Project Options**, pilih tab **Directories/Conditionals**
- Pada **Input Search Path** isilah dengan alamat *folder driver component* yang telah anda buat
- Pilih menu **Project → Build All Projects**
- Pilih menu **Run → Run**
- Selamat mencoba

Sample application terdiri atas sebuah *form* utama (*main form*) yaitu pada Unit1.pas dan dua buah *form* tambahan (*additional forms*) yaitu uDialPad.pas dan uBlockList.pas uBlockList.pas berisi program yang berhubungan dengan *database* nomor yang diblokir dan tidak dibahas disini.

Contoh-contoh aplikasi pada Unit1.pas
Keterangan yang perlu diperhatikan:

- Pada unit ini terdapat Form1 yang mempunyai *subclass* bernama PI.
- PI merupakan nama dari suatu komponen yang bertipe TPhoneInterface yang diletakkan pada *form.*

Contoh aplikasi yang menggunakan *method* ReadDP:



Gambar 8
***Flowchart* untuk *Listing* 1**

***Listing* 1**
Contoh Aplikasi yang Menggunakan *Method* ReadDP

```
{=============================================================}
{ This method gives an example how to use ReadDP method.      }
{ To check if there is an early indication of decadic dialing  }
{ pulse, you can use In_HK propery.                           }
{ The use of In_HK property must be combined with a timer to   }
{ check if the hook signal satisfies the timing requirement to }
{ be a decadic dialing pulse.                                 }
{ The Winapi functions CreateWaitableTimer and                }
{ SetWaitableTimer are used in this example for working with   }
{ timer in Windows programming.                               }
{=============================================================}
procedure TForm1.DoPulseDetection;
var
  LastTime: THandle;
  LastHK: Boolean;
  Skip: Boolean;
  Pulse: Integer;
  NoPulse: Boolean;
  RDP: Integer;
  LastTime2: THandle;
  Tmp64: Int64;
begin
  LabelDTMF.Caption:='';
  PulseDetectionStopped:=false;
  LastTime2:=CreateWaitableTimer(nil,False,nil);
  LastTime:=CreateWaitableTimer(nil,False,nil);
  PhoneInterfaceManager.Thread.Suspend;
  try
    StartPulseDetection1.Caption:='Stop &Pulse Detection';
    try
      PI.Init(InitDTMF1Val,InitDTMF1Val2);
      RDP:=0;
      NoPulse:=True;
      LastTime:=0;
      Pulse:=0;
      LastHK:=True;
      repeat
        if not Skip then
        begin
          Application.ProcessMessages;
          if not PI.In_HK then
          begin
            if LastHK then
            begin
              Tmp64:=-350000;
              SetWaitableTimer(LastTime,tmp64,0,nil,nil,False);
              LastHK:=false;
            end;
          end;
          if PI.In_HK then
          begin
            LastHK:=True;
          end;
          if not PI.In_HK and not LastHK and
            (WaitForSingleObject(LastTime,0)=WAIT_OBJECT_0) then
          begin
            LabelDTMF.Caption:='';
            Pulse:=0;
            NoPulse:=True;
            Application.ProcessMessages;
          end;
```

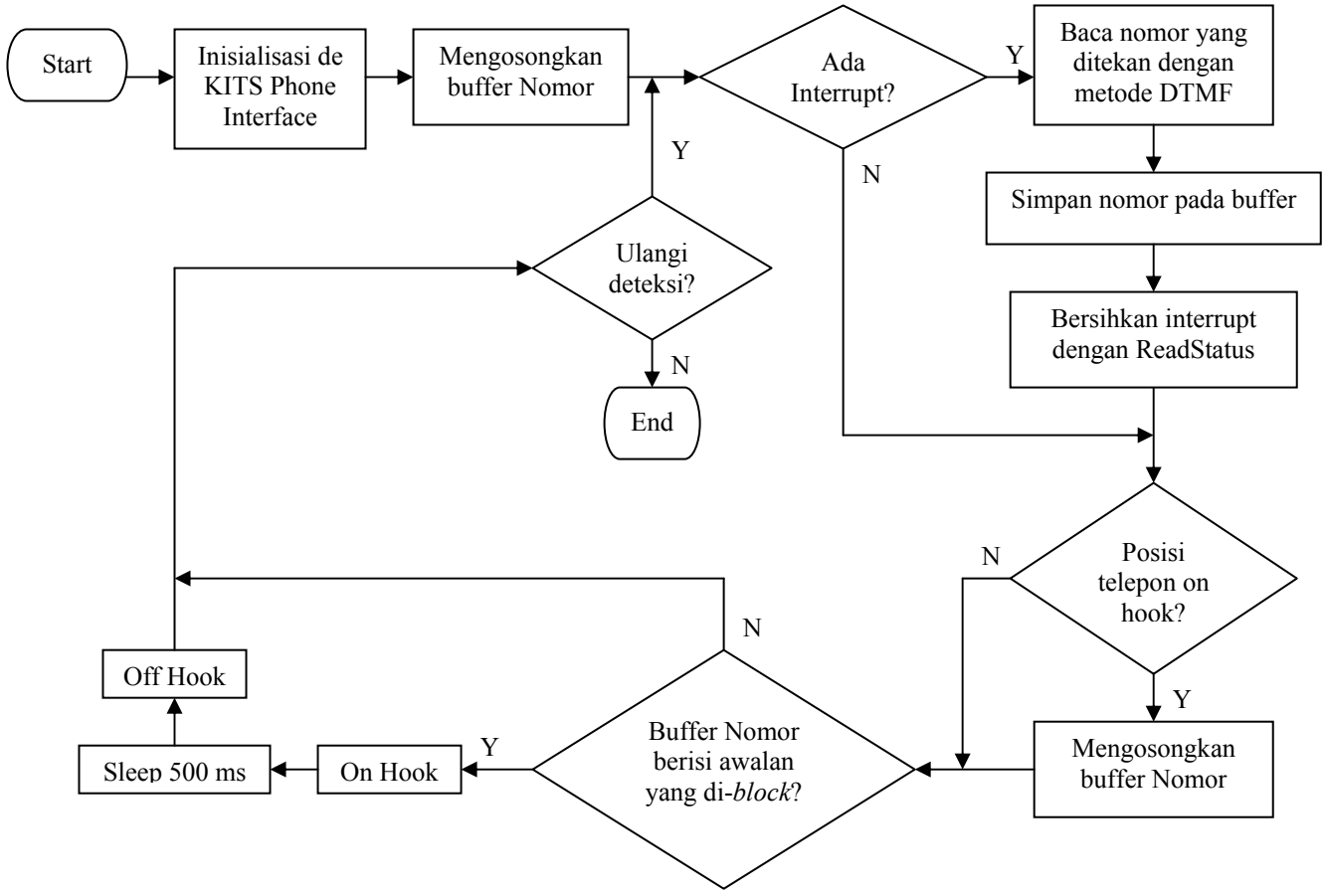
```
          // if Number is found in blocked list
          if Self.Tb.FindKey([LabelDTMF.Caption]) then
          begin
            // On Hook
            PI.Out_RE:=True;
            SleepButProcessMessage(500,MainThreadID);
            // Off Hook
            PI.Out_RE:=False;
            PI.ReadStatus;
            LabelDTMF.Caption:='';
            Pulse:=0;
            NoPulse:=True;
            Application.ProcessMessages;
          end;
        end;

        if not PI.In_HK then
        begin
          RDP:=PI.ReadDP;
          if RDP=1 then
          begin
            Tmp64:=-2000000;
            SetWaitableTimer(LastTime2,Tmp64,0,nil,nil,False);
            Inc(Pulse);
            Skip:=True;
            NoPulse:=False;
          end else begin
            Skip:=false;
            if RDP=2 then
            begin
              LabelDTMF.Caption:='';
              Pulse:=0;
              NoPulse:=True;
              Application.ProcessMessages;
            end;
            if RDP=0 then
            begin
              LabelDTMF.Caption:='';
              Pulse:=0;
              NoPulse:=True;
              Application.ProcessMessages;
            end;
          end;
        end else begin
          Skip:=false;
          Application.ProcessMessages;
        end;
        if (RDP=1) and PI.In_HK and
          (WaitForSingleObject(LastTime2,0)=WAIT_OBJECT_0) then
        begin
          if not NoPulse then
          begin
            if Pulse=10 then Pulse:=0;
            LabelDTMF.Caption:=LabelDTMF.Caption+IntToStr(Pulse);
            RDP:=0;
            Pulse:=0;
            NoPulse:=True;
          end;
        end;
      until Application.Terminated or PulseDetectionStopped;
    finally
      StartPulseDetection1.Caption:='Start &Pulse Detection';
    end;
  finally
```

```
      PhoneInterfaceManager.Thread.Resume;
      CloseHandle(LastTIme);
      CloseHandle(LastTIme2);
    end;
 end;
```

Pada *Listing* 1 tersebut, gambaran umum langkah-langkah yang dilakukan adalah:

1. Melakukan inisialisasi terhadap de KITS Phone Interface dan mengosongkan *buffer* nomor yang ditekan.
2. Memeriksa apakah posisi telepon *on hook* dan apakah waktu terakhir *off hook* sampai waktu sekarang lebih besar dari 35 ms?
3. Jika ya, kosongkan *buffer* nomor yang ditekan.
4. Memeriksa (apakah *buffer* nomor berisi awalan nomor yang di-*block*)?
5. Jika ya, maka lakukan *on hook*, *sleep* 500 ms, lalu *off hook* lagi.
6. Jika (posisi telepon *off hook*) {yaitu kemungkinan ada *pulse*}, jalankan ReadDP,
7. Jika (hasil ReadDP=1) {berarti ditemukan *valid dialing pulse*}, ulangi terus langkah 7 terus sampai terdapat jeda 200 ms tanpa ditemukan *dialing pulse*.
8. Jika (hasil ReadDP<>1), kosongkan *buffer* nomor yang ditekan.
9. Jika ditemukan 10 kali *pulse dialing* berarti yang dimaksud angka nol, maka ubah variabel pulse menjadi 0.
10. Simpan nomor yang didapat pada *buffer* nomor.
11. Jika proses pembacaan *pulse dialing* masih diinginkan, program akan kembali ke langkah 2. Jika tidak, maka program akan diakhiri.

Contoh aplikasi yang menggunakan *method* ReadDTMF:



Gambar 9
***Flowchart* untuk *Listing* 2**

***Listing* 2**
Contoh Aplikasi yang Menggunakan *Method* ReadDTMF

```
{=============================================================}
{ This method gives an example how to use ReadDTMF method.     }
{ use the In_IRQ property to check if the Phone Interface is   }
{ ready with a data in its buffer.                             }
{ After reading the DTMF Value, you should clear the In_IRQ    }
{ interrupt using ReadStatus method.                           }
{=============================================================}
procedure TForm1.DoToneDetection;
begin
  LabelDTMF.Caption:='';
  ToneDetectionStopped:=false;
  StartDTMFDetection1.Caption:='Stop &DTMF Detection';
  try
    PI.Init(InitDTMF1Val,InitDTMF1Val2);
    repeat
      Application.ProcessMessages;
      if PI.In_IRQ then
      begin
        LabelDTMF.Caption:=LabelDTMF.Caption+IntToStr(PI.ReadDTMF and $0000000F);
        Application.ProcessMessages;
        PI.ReadStatus;
        SleepButProcessMessage(50,MainThreadID);
      end;
      if not PI.In_HK then
      begin
        LabelDTMF.Caption:='';
        Application.ProcessMessages;
      end;
      // if Number is found in blocked list
      if Self.Tb.FindKey([LabelDTMF.Caption]) then
      begin
        // On Hook
        PI.Out_RE:=True;
        SleepButProcessMessage(500,MainThreadID);
        // Off Hook
        PI.Out_RE:=False;
        PI.ReadStatus;
        LabelDTMF.Caption:='';
        Application.ProcessMessages;
      end;
    until Application.Terminated or ToneDetectionStopped;
  finally
    StartDTMFDetection1.Caption:='Start &DTMF Detection';
  end;
end;
```

Pada *Listing* 2 tersebut, gambaran umum langkah-langkah yang dilakukan adalah:
1. Melakukan inisialisasi terhadap de KITS Phone Interface dan mengosongkan *buffer* nomor yang ditekan.
2. Memeriksa (apakah ada *interrupt*)?
3. Jika ya, baca nomor yang ditekan dengan *method* ReadDTMF lalu simpan di *buffer* nomor yang ditekan, bersihkan *interrupt* dengan ReadStatus.
4. Memeriksa (apakah telepon dalam kondisi *on hook*)?
5. Jika ya, kosongkan *buffer* nomor yang ditekan.
6. Memeriksa (apakah *buffer* nomor berisi awalan nomor yang di-*block*)?
7. Jika ya, maka lakukan *on hook*, *sleep* 500 ms, lalu *off hook* lagi.
8. Jika proses pembacaan DTMF masih diinginkan, maka program akan kembali ke langkah 2. Jika tidak, maka program akan diakhiri.

Contoh aplikasi deteksi *call progress* dan *ring in*:

***Listing* 3**
Contoh Aplikasi Deteksi *Call Progress* dan *Ring In*

```
{=============================================================}
{ This event handler demonstrate how to read the call progress }
{ status. The CekBusy, CekRingBack                              }
{ CekEarlyIndicationFindRingBackOrBusy methods are used to help}
{ this event handler to work effectively.                       }
{ This event is triggered each time the call progress status    }
{ changed from high to low or low to high. To get the call      }
{ progress status value, use CekDialTone method.                }
{ Note: This event will still be trigerred though the ring      }
{ back or busy tones is being detected.                         }
{=============================================================}
procedure TForm1.PICallProgressDialTone(Sender: TObject);
begin
  if not PI.CekBusy and not PI.CekRingBack and not PI.In_RI and not
PI.CekEarlyIndicationFindRingBackOrBusy then
  begin
    if PI.CekDialTone then
    begin
      if PI.RingInCount=0 then
        LabelDTMF.Caption:='DIAL TONE';
    end else begin
      if PI.RingInCount=0 then
        LabelDTMF.Caption:='NO DIAL TONE';
    end;
  end;
end;

{=============================================================}
{ The internal engine of the driver is capable to detect the    }
{ busy tone and call this event handler if a valid busy tone    }
{ has been detected. This event handler will also be called if  }
{ the busy tone has been stopped.                               }
{ Use CekBusy method to read the busy tone status.              }
{=============================================================}
procedure TForm1.PICallProgressBusy(Sender: TObject);
begin
  if not PI.CekBusy then
  begin
    if PI.CekDialTone then
    begin
      if PI.RingInCount=0 then
        LabelDTMF.Caption:='DIAL TONE';
    end else begin
      if PI.RingInCount=0 then
        LabelDTMF.Caption:='NO DIAL TONE';
    end;
  end else begin
    if PI.RingInCount=0 then
      LabelDTMF.Caption:='BUSY';
  end;
end;

{=============================================================}
{ The internal engine of the driver is capable to detect the    }
{ ring back tone and call this event handler if a valid ring    }
{ back tone has been detected. This event handler will also be  }
{ called if the ring back tone has been stopped.                }
{ Use CekRingBack method to read the ring back tone status.     }
{=============================================================}
```

```
procedure TForm1.PICallProgressRingBack(Sender: TObject);
begin
  if not PI.CekRingBack then
  begin
    if PI.CekDialTone then
    begin
      if PI.RingInCount=0 then
        LabelDTMF.Caption:='DIAL TONE';
    end else begin
      if PI.RingInCount=0 then
        LabelDTMF.Caption:='NO DIAL TONE';
    end;
  end else begin
    if PI.RingInCount=0 then
      LabelDTMF.Caption:='RING BACK';
  end;
end;

{=============================================================}
{ The internal engine of the driver is capable to detect the   }
{ ring in condition and call this event handler.               }
{ If there's a ring in progress the RingInCount property value }
{ is nonzero and indicating the number of how many ring has    }
{ been offered, otherwise it will be set to zero.              }
{ This event handler will be called directly after RingInCount }
{ has been incremented or set to zero by the engine.           }
{=============================================================}
procedure TForm1.PIRingIn(Sender: TObject);
begin
  if (StartPulseDetection1.Caption='Start &Pulse Detection') then
  begin
    if PI.RingInCount<>0 then
    begin
      LabelDTMF.Caption:='RING IN: '+IntToStr(PI.RingInCount);
    end else begin
      if PI.PerformCPDetection then
        PI.OnCallProgressDialTone(Sender)
      else
        LabelDTMF.Caption:='';
    end;
  end;
end;
```

Pada *Listing* 3 tersebut, terdapat 4 *event* yang di-*assign*, berikut penjelasannnya:

1. Ada tiga *event* yang dipanggil jika PerformCPDetection bernilai TRUE, ketiganya adalah OnCallProgressDialTone, OnCallProgressBusy, OnCallProgressRingBack. Jadi *property* PerformCPDetection harus bernilai TRUE untuk mengaktifkan deteksi *call progress*.
2. Pada *event* OnCallProgressDialTone, deteksi utama dilakukan dengan *method* CekDialTone, untuk memeriksa ada tidaknya *dial tone*. Ada tidaknya *dial tone* disampaikan ke pengguna melalui obyek LabelDTMF dengan memberi nilai pada *property* Caption. Namun karena label tersebut juga digunakan untuk menginformasikan ke pengguna ada tidaknya *ring back tone* maupun *busy tone*, maka CekBusy, CekRingBack, dan sebagainya juga dipanggil.
3. Untuk *event* yang lain prinsipnya sama dengan OnCallProgressDialTone, yaitu bertujuan menampilkan status *call progress* telepon ke pengguna.

Contoh aplikasi pada uDialPad.pas
Keterangan yang perlu diperhatikan:

- Pada unit ini terdapat fmDialPad yang mempunyai *subclass* bernama PI.
- PI merupakan nama dari suatu komponen yang bertipe TPhoneInterface yang diletakkan pada *form*.

Contoh penggunaan WriteDTMF dan WriteDP:

***Listing* 4**
Contoh Penggunaan WriteDTMF dan WriteDP

```
{=============================================================}
{ This method gives example how to use WriteDTMF and WriteDP   }
{ methods. After calling WriteDTMF, do Sleep(150) then you     }
{ should call ReadStatus to clear the In_IRQ interrupt.        }
{=============================================================}
procedure TfmDialPad.b1Click(Sender: TObject);
begin
  if not InProgress then
  begin
    InProgress:=True;
    try
      if rgDialMode.ItemIndex=0 then
      begin
        PI.WriteDTMF(1);
      end else begin
        PI.WriteDP(1);
      end;
      Sleep(150);
      PI.ReadStatus;
      pnNum.Caption:=pnNum.Caption+TButton(Sender).Caption;
    finally
      InProgress:=false;
    end;
  end;
end;
```

Pada contoh tersebut ditunjukkan bagaimana menulis angka 1 dalam bentuk DTMF maupun *dialing pulse* ke de KITS Phone Interface dengan menggunakan WriteDTMF dan WriteDP. Jangan lupa memanggil Sleep(150) dan ReadStatus bila selesai menggunakan WriteDTMF. RgDialMode.ItemIndex digunakan untuk menentukan apakah mode DTMF atau mode *dialing pulse*. Bila angka-angka dituliskan secara berurutan maka akan menjadi nomor telepon yang lengkap.

Listing program contoh dan *driver* terdapat pada **AN62.ZIP**.

Selamat berinovasi!