DT-51 Application Note

AN14 - How 2 Use DT-51 KND with DT-51 MinSys ver 3.0

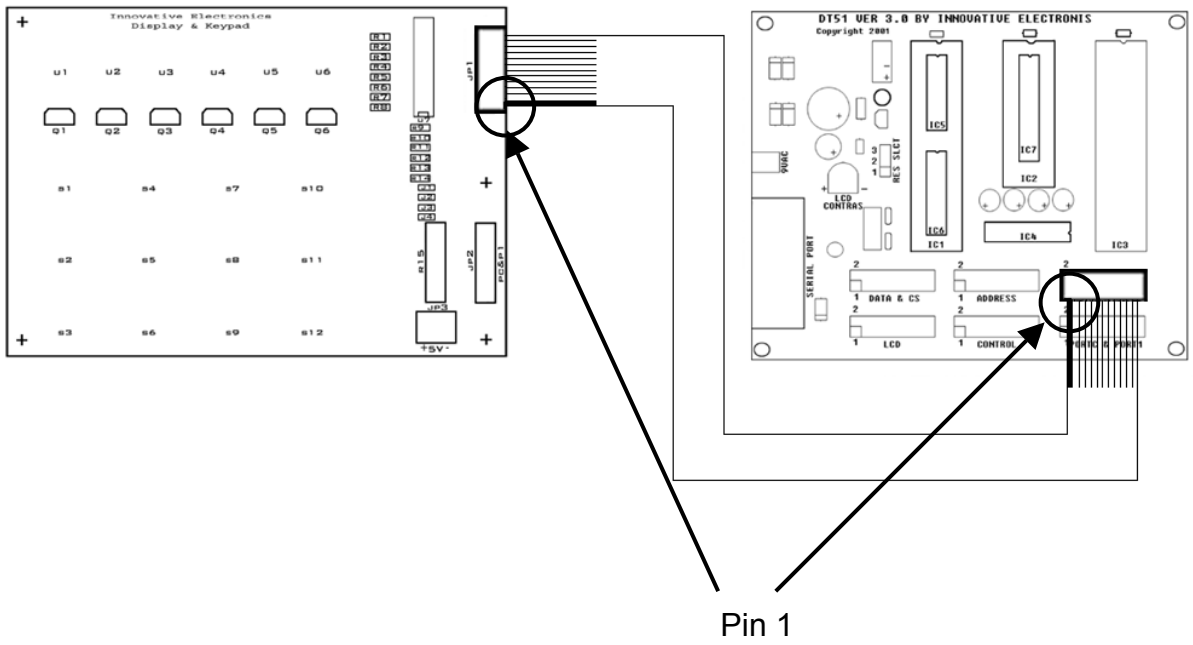
oleh: Tim IE & Sapto Jayadi Sutandi (Universitas Kristen Petra)

Sesuai namanya, application note ini menjelaskan tentang penggunaan DT-51 KND. AN ini dibuat agar pengguna dapat mengerti bagaimana menggunakan produk ini, bahkan oleh pengguna awam sekalipun. Sebagai mikrokontroler, AN ini menggunakan DT-51 MinSys ver 3.0. tetapi tidak menutup kemungkinan jika ada pengguna yang menggunakan mikrokontroler lain.

Modul-modul yang digunakan adalah:
- DT-51 MinSys Ver 3.0
- DT-51 KND

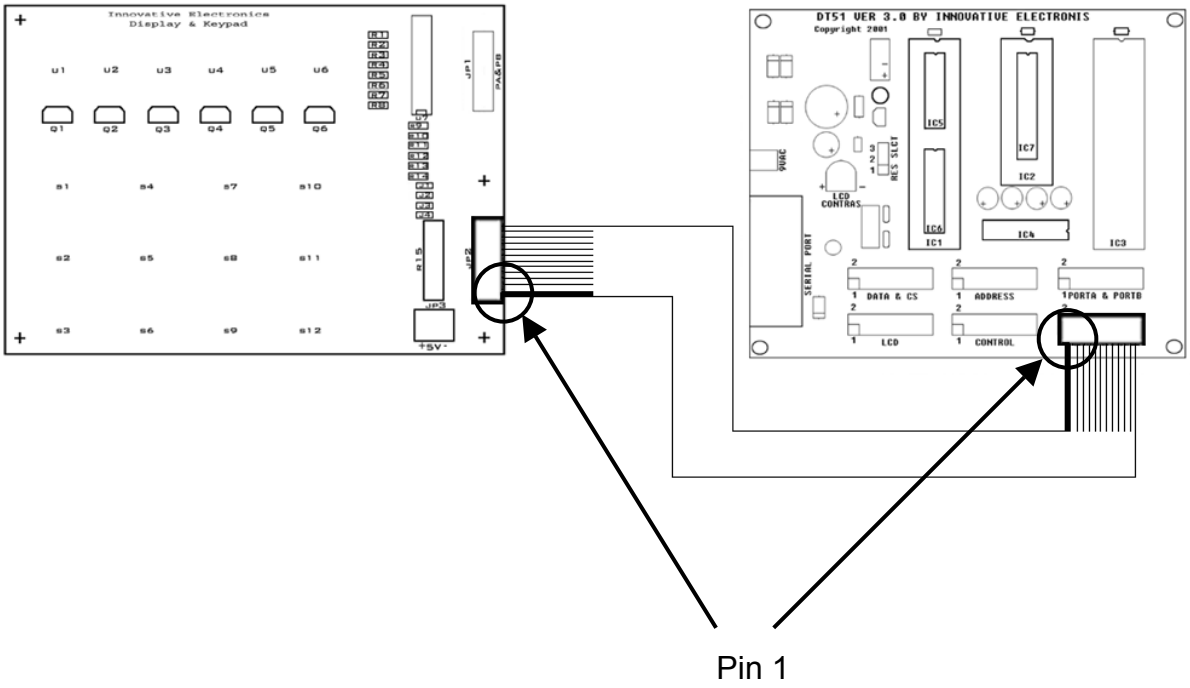
MENGHUBUNGKAN DT-51 KND DENGAN DT-51 ver 3.0

1. Hubungkan JP1 pada DT-51 KND dengan PORTA & PORTB pada DT-51 MinSys Ver 3.0 dengan menggunakan kabel pita.



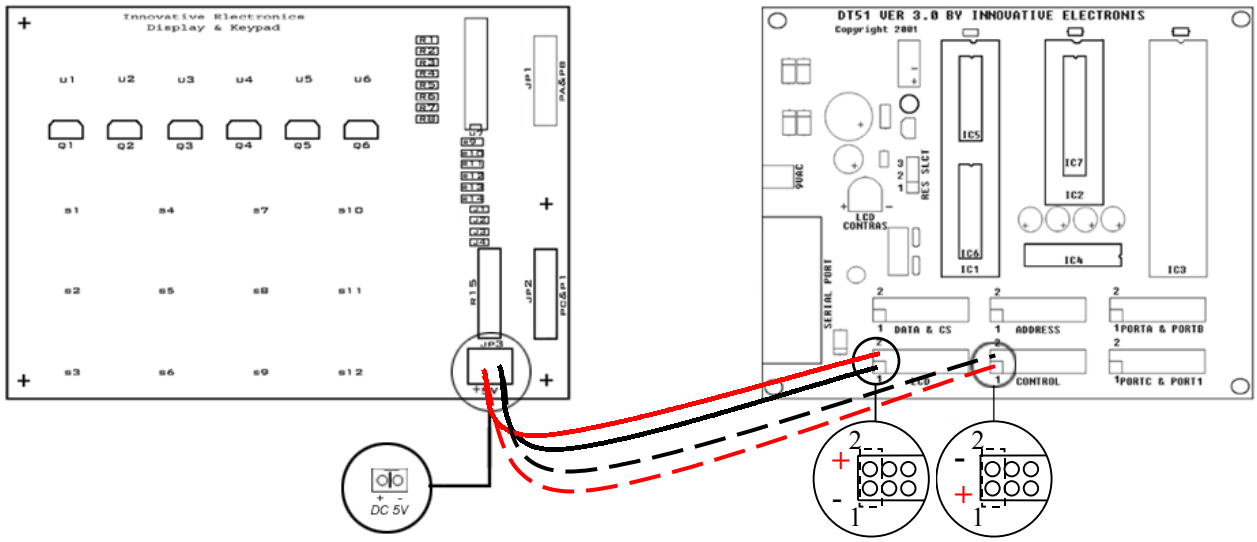
Gambar 1. Menghubungkan JP1 pada DT-51 KND dengan PORTA & PORTB pada DT-51 MinSys ver. 3.0

2. Hubungkan JP2 pada DT-51 KND dengan PORTC & PORT1 pada DT-51 MinSys Ver 3.0 dengan menggunakan kabel pita.



Gambar 2. Menghubungkan JP2 pada DT-51 KND dengan PORTC & PORT1 pada DT-51 MinSys ver. 3.0

3. Berikan tegangan sebesar 5V DC pada JP3, tegangan bisa diambil dari sumber tegangan independen atau dari DT-51 MinSys Ver 3.0.
   Jika mengambil tegangan dari Port LCD DT-51 MinSys, perhatikan bahwa pin 1 adalah GND dan pin 2 adalah VCC.
   Jika mengambil tegangan dari Port Control DT-51 MinSys, perhatikan bahwa pin 1 adalah VCC dan pin 2 adalah GND.

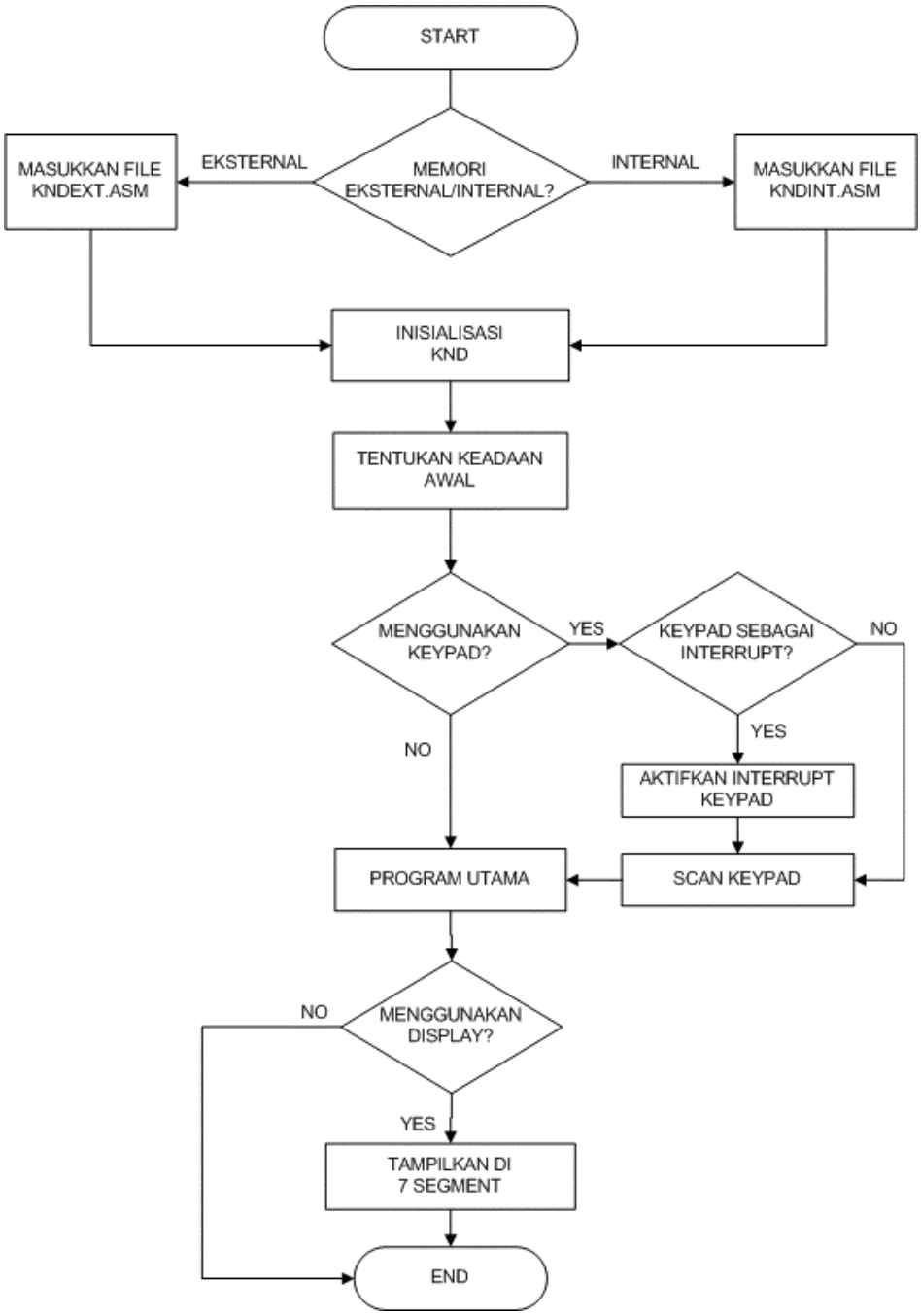


Perhatikan polaritas kabel jangan sampai terbalik karena dapat menimbulkan kerusakan pada DT-51 KND

Gambar 3. Menghubungkan JP3 pada DT-51 KND dengan pin 1 & 2 Port LCD pada DT-51 MinSys ver. 3.0 atau dengan pin 1 & 2 Port Control pada DT-51 MinSys ver 3.0

MEMPROGRAM DT-51 KND

Secara garis besar urutan–urutan untuk memprogram DT-51 KND adalah seperti flowchart pada gambar 4. Hal ini akan memudahkan dalam memprogram DT-51 KND melalui Minimum System DT-51 MinSys Ver 3.0.



Gambar 4. Flowchart pemrograman DT-51 KND secara umum

Untuk proses inisialisasi DT-51 KND menggunakan rutin yang disertakan dalam modul DT-51 KND dengan memasukkan salah satu dari file KNDEXT.ASM atau KNDINT.ASM ke dalam awal program.
Demikian juga dengan proses scanning keypad dan tampilan di 7 Segment, anda hanya perlu memanggil rutin-rutin yang telah disediakan.
Pada bagian program utama diisi dengan proses yang diinginkan.

PENJELASAN SINGKAT

- Memori Eksternal/Internal:
  Perbedaan antara memori eksternal dan internal adalah jenis memori yang digunakan.
  Memori Internal menggunakan memori internal mikrokontroler 89C51 sehingga memori yang dapat digunakan lebih kecil namun kecepatannya lebih cepat. Umumnya memori internal digunakan untuk stack program. Untuk menggunakan memori internal, file KNDINT.ASM dimasukkan dalam program.
  Memori Eksternal menggunakan EEPROM 28C64 sehingga memori yang dapat digunakan lebih besar namun kecepatannya agak berkurang meski hampir tidak berarti. Biasanya memori eksternal digunakan bila memori internal pada 89C51 sudah habis. Untuk menggunakan memori eksternal, file KNDEXT.ASM dimasukkan dalam program.

- Inisialisasi KND:
  Inisialisasi dilakukan dengan cara memanggil rutin InitKND.

- Tentukan keadaan awal:
  Keadaan awal (meliputi nilai variabel) yang diinginkan diatur terlebih dahulu.

- Menggunakan Keypad:
  Jika dikehendaki untuk menggunakan keypad, prosedur Scanning dimasukkan pada program dengan alamat 400Bh.
  Catatan: Keypad bisa berfungsi sebagai interrupt eksternal atau bukan.

- Program Utama:
  Bagian ini diisi dengan proses yang diinginkan.

- Aktifkan Interrupt Keypad:
  Jika dikehendaki keypad sebagai interrupt eksternal, proses untuk mengaktifkannya dengan cara memanggil rutin EnbKeyInt.
  Saat keypad berfungsi sebagai interrupt eksternal, maka tiap penekanan keypad akan dianggap sebagai interrupt eksternal 1 sehingga program tidak perlu melakukan polling untuk mengetahui bilamana keypad ditekan.
  Tapi jika keypad tidak berfungsi sebagai interrupt eksternal, maka harus dibuat prosedur singkat untuk polling (lihat SAMPLE1.ASM pada disket DT-51 KND), dengan memeriksa variabel KeyPressed.

- Matikan Interrupt Keypad:
  Jika dikehendaki keypad bukan sebagai interrupt eksternal atau mematikan interrupt eksternal untuk sementara waktu (mungkin untuk menghindari gangguan selama program utama berjalan), dapat dilakukan dengan cara memanggil rutin DisKeyInt.

- Scan Keypad:
  Proses scan keypad digunakan untuk memeriksa keypad mana yang ditekan.

- Menampilkan karakter di 7 Segment:
  Untuk menampilkan karakter di 7 Segment, panggil rutin Write7S

PROGRAM CONTOH

Contoh 1 (EXAMPLE1.ASM):

```
MOD51
$TITLE(Example Program Using DT-51 KND Routine)

;*********************************************
```

```
;*          Up Counter                  *
;*          Using InitKND,Write7S,      *
;*          LDelay,ClrDisMem            *
;*********************************************
;Program ini akan menampilkan karakter-karakter yang ada di Library KND mulai dari angka 0 - 9 secara
;berurutan pada 7 segment. Dengan menggunakan ClrDisMem di akhir setiap penampilan angka, maka
;akan terlihat hanya 1 buah 7 segment yang digunakan.

            CSEG
            ORG     4000H
            LJMP    Start

            ORG     4100H
            $INCLUDE(KNDINT.ASM)

Start:      MOV     SP, #53H        ;Menentukan stack pointer untuk posisi awal program
            LCALL   InitKND         ;Inisialisasi KND

Loop:       MOV     A, #0h          ;Isi ACC dengan karakter yang akan ditampilkan
            LCALL   Write7S         ;Prosedur untuk menampilkan karakter di 7 segment
            LCALL   LDelay          ;Prosedur delay (terdapat dalam KNDINT.ASM)
            LCALL   ClrDisMem       ;Hapus Layar

            MOV     A, #1h          ;Isi ACC dengan karakter yang akan ditampilkan
            LCALL   Write7S         ;Prosedur untuk menampilkan karakter di 7 segment
            LCALL   LDelay          ;Prosedur delay (terdapat dalam KNDINT.ASM)
            LCALL   ClrDisMem       ;Hapus Layar

            MOV     A, #2h          ;Isi ACC dengan karakter yang akan ditampilkan
            LCALL   Write7S         ;Prosedur untuk menampilkan karakter di 7 segment
            LCALL   LDelay          ;Prosedur delay (terdapat dalam KNDINT.ASM)
            LCALL   ClrDisMem       ;Hapus Layar

            MOV     A, #3h          ;Isi ACC dengan karakter yang akan ditampilkan
            LCALL   Write7S         ;Prosedur untuk menampilkan karakter di 7 segment
            LCALL   LDelay          ;Prosedur delay (terdapat dalam KNDINT.ASM)
            LCALL   ClrDisMem       ;Hapus Layar

            MOV     A, #4h          ;Isi ACC dengan karakter yang akan ditampilkan
            LCALL   Write7S         ;Prosedur untuk menampilkan karakter di 7 segment
            LCALL   LDelay          ;Prosedur delay (terdapat dalam KNDINT.ASM)
            LCALL   ClrDisMem       ;Hapus Layar

            MOV     A, #5h          ;Isi ACC dengan karakter yang akan ditampilkan
            LCALL   Write7S         ;Prosedur untuk menampilkan karakter di 7 segment
            LCALL   LDelay          ;Prosedur delay (terdapat dalam KNDINT.ASM)
            LCALL   ClrDisMem       ;Hapus Layar

            MOV     A, #6h          ;Isi ACC dengan karakter yang akan ditampilkan
            LCALL   Write7S         ;Prosedur untuk menampilkan karakter di 7 segment
            LCALL   LDelay          ;Prosedur delay (terdapat dalam KNDINT.ASM)
            LCALL   ClrDisMem       ;Hapus Layar

            MOV     A, #7h          ;Isi ACC dengan karakter yang akan ditampilkan
            LCALL   Write7S         ;Prosedur untuk menampilkan karakter di 7 segment
            LCALL   LDelay          ;Prosedur delay (terdapat dalam KNDINT.ASM)
            LCALL   ClrDisMem       ;Hapus Layar

            MOV     A, #8h          ;Isi ACC dengan karakter yang akan ditampilkan
            LCALL   Write7S         ;Prosedur untuk menampilkan karakter di 7 segment
```

```
                LCALL   LDelay          ;Prosedur delay (terdapat dalam KNDINT.ASM)
                LCALL   ClrDisMem       ;Hapus Layar

                MOV     A, #9h          ;Isi ACC dengan karakter yang akan ditampilkan
                LCALL   Write7S         ;Prosedur untuk menampilkan karakter di 7 segment
                LCALL   LDelay          ;Prosedur delay (terdapat dalam KNDINT.ASM)
                LCALL   ClrDisMem       ;Hapus Layar

                AJMP    Loop

                END
```

Contoh 2 (EXAMPLE2.ASM):

```
$MOD51
$TITLE(Example Program Using DT-51 KND Routine)

;*******************************************************
;*              Up Counter  2                          *
;*              Using InitKND,Write7S,                 *
;*              LDelay,RealDis,ClrDisMem               *
;*******************************************************
;Program ini akan menunjukkan hasil tampilan RealDis.
;Pertama-tama display akan dituliskan angka 0 - 5 (6 karakter). Bila ada penulisan karakter ke 7 maka
;semua tampilan akan dihapus digantikan dengan karakter ke 7, dst.

                CSEG
                ORG     4000H
                LJMP    Start

                ORG     4100H
                $INCLUDE(KNDINT.ASM)

Start:          MOV     SP, #53H        ;Menentukan Stack Pointer program
                LCALL   InitKND         ;Inisialisasi KND
                LCALL   RealDis         ;Menggunakan RealDis
                LCALL   ClrDisMem       ;Hapus tampilan

Awal:           MOV     R6, #6H
                MOV     A, #0H          ;Isi ACC dengan karakter yang akan ditampilkan
Loop:           ACALL   Write7S         ;Prosedur untuk menampilkan karakter di 7 segment
                LCALL   LDelay          ;Prosedur delay (terdapat dalam KNDINT.ASM)
                INC     A
                DJNZ    R6, Loop        ;Looping kembali ke 0 bila telah mencapai 5

                LCALL   LDelay
                LCALL   LDelay
                LCALL   LDelay
                LCALL   LDelay

                MOV     A, #6H          ;Jika ada penulisan karakter ke 7 maka
                ACALL   Write7S         ;6 karakter sebelumnya akan dihapus
                LCALL   LDelay          ;digantikan dengan karakter ke 7
                LCALL   LDelay

                LJMP    Awal

                END
```

Contoh 3 (EXAMPLE3.ASM):

```
$MOD51
$TITLE(Example Program Using DT-51 KND Routine)

;************************************************
;*          Scan Keypad without interrupt       *
;*          Using InitKND,Write7S,              *
;*          LDelay,CursorON,CursorOFF           *
;*          RealDis,CLrDisMem                   *
;************************************************
;Program ini akan menampilkan karakter-karakter yang ada di Library KND.
;Pertama akan menampilkan kata PUSANY dari display 6 buah 7 segment yang ada. Kemudian tekan
;sembarang tombol
;Tombol 1-9 akan menampilkan karakter 0AH-12H
;Tombol 10:Menghidupkan Cursor setelah ketik sesuatu
;Tombol 11:Mematikan Cursor setelah ketik sesuatu
;Tombol 12:Menghapus layar

                CSEG
                ORG     4000H
                LJMP    Start

                ORG     400BH
                LJMP    Scanning

                ORG     4100H
                $INCLUDE(KNDINT.ASM)

JMPTABLE:       LJMP    S1Act
                LJMP    S2Act
                LJMP    S3Act
                LJMP    S4Act
                LJMP    S5Act
                LJMP    S6Act
                LJMP    S7Act
                LJMP    S8Act
                LJMP    S9Act
                LJMP    S10Act
                LJMP    S11Act
                LJMP    S12Act

S1Act:          MOV     A, #0Ah
                ACALL   Write7S
                LJMP    Loop
S2Act:          MOV     A, #0BH
                ACALL   Write7S
                LJMP    Loop
S3Act:          MOV     A, #0CH
                ACALL   Write7S
                LJMP    Loop
S4Act:          MOV     A, #0DH
                ACALL   Write7S
                LJMP    Loop
S5Act:          MOV     A, #0EH
                ACALL   Write7S
                LJMP    Loop
S6Act:          MOV     A, #0FH
                ACALL   Write7S
```

```
                LJMP    Loop
S7Act:          MOV     A, #10H
                ACALL   Write7S
                LJMP    Loop
S8Act:          MOV     A, #11H
                ACALL   Write7S
                LJMP    Loop
S9Act:          MOV     A, #12H
                ACALL   Write7S
                LJMP    Loop
S10Act:         LCALL   CursorON
                LJMP    Loop
S11Act:         ACALL   CursorOFF
                LJMP    Loop
S12Act:         ACALL   ClrDisMem
                LJMP    Loop

PusAny:         MOV     A, #16h         ;Isi ACC dengan karakter "P"
                ACALL   Write7S         ;Prosedur menampilkan karakter di 7 segment
                LCALL   LDelay          ;Prosedur delay (terdapat dalam KNDINT.ASM)

                MOV     A, #17h         ;Isi ACC dengan karakter "U"
                ACALL   Write7S         ;Prosedur menampilkan karakter di 7 segment
                LCALL   LDelay          ;Prosedur delay (terdapat dalam KNDINT.ASM)

                MOV     A, #05h         ;Isi ACC dengan karakter "S"
                ACALL   Write7S         ;Prosedur menampilkan karakter di 7 segment
                LCALL   LDelay          ;Prosedur delay (terdapat dalam KNDINT.ASM)

                MOV     A, #11h         ;Isi ACC dengan karakter "A"
                ACALL   Write7S         ;Prosedur menampilkan karakter di 7 segment
                LCALL   LDelay          ;Prosedur delay (terdapat dalam KNDINT.ASM)

                MOV     A, #0Ah         ;Isi ACC dengan karakter "N"
                ACALL   Write7S         ;Prosedur menampilkan karakter di 7 segment
                LCALL   LDelay          ;Prosedur delay (terdapat dalam KNDINT.ASM)

                MOV     A, #0Bh         ;Isi ACC dengan karakter "Y"
                ACALL   Write7S         ;Prosedur menampilkan karakter di 7 segment
                LCALL   LDelay          ;Prosedur delay (terdapat dalam KNDINT.ASM)
                RET

Start:          MOV     SP, #53H
                LCALL   InitKND
                LCALL   RealDis
                LCALL   ClrDisMem
                LCALL   PusAny

Loop:           JNB     KeyPressed, $   ;memeriksa variabel KeyPressed sebagai polling
                                        ;untuk penekanan keypad
                CLR     KeyPressed
                MOV     A, KeyCode
                MOV     DPTR, #JMPTABLE
                DEC     A
                MOV     B, #3
                MUL     AB
                JMP     @A+DPTR
                END
```

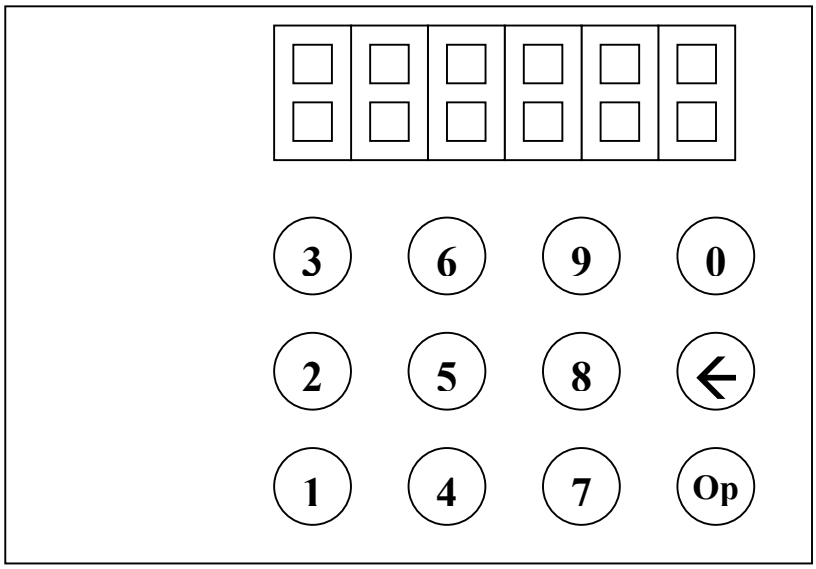
Contoh 4 (EXAMPLE4.ASM):

```
$MOD51
$TITLE(Example Program Using DT-51 KND Routine)

;*******************************************************
;*      Character Display with cursor                  *
;*      Using InitKND,Write7S,                         *
;*      LDelay,CursorON                                *
;*      VirtualDis,ClrDisMem                           *
;*      ShftDisLeft,ShftDisRight                       *
;*******************************************************
;Program ini akan menampilkan karakter-karakter yang ada di Library KND satu persatu dari 0H sampai
;12H. Tampilan yang digunakan seolah-olah ada 18 karakter meski 7 segment yang ada hanya 6. Caranya
;dengan menggeser tampilan ke kiri-kanan dengan prosedur ShftDisLeft dan ShftDisRight setiap kali ada
;interrupt keypad 1 dan 2 Jika ada penekanan keypad 3 maka akan menghapus tampilan. Sedangkan jika
;ada penekanan keypad 4 maka akan menulis lagi. Keypad yg lain tidak digunakan.

                CSEG
                ORG     4000H
                LJMP    Start

                ORG     400BH
                LJMP    Scanning
                ORG     4013H
                LJMP    ISR_EI1

                ORG     4100H
                $INCLUDE(KNDINT.ASM)

ISR_EI1:        MOV     A, KeyCode
                MOV     DPTR, #JMPTABLE
                DEC     A
                MOV     B, #3
                MUL     AB
                JMP     @A+DPTR

JMPTABLE:       LJMP    S1Act
                LJMP    S2Act
                LJMP    S3Act
                LJMP    S4Act
                LJMP    S5Act
                LJMP    S6Act
                LJMP    S7Act
                LJMP    S8Act
                LJMP    S9Act
                LJMP    S10Act
                LJMP    S11Act
                LJMP    S12Act

S1Act:          LCALL   ShftDisRight    ;Geser tampilan ke kanan
                RETI
S2Act:          LCALL   ShftDisLeft     ;Geser tampilan ke kiri
                RETI
S3Act:          LCALL   ClrDisMem       ;Hapus tampilan
                RETI
S4Act:          LCALL   Writing         ;Panggil prosedur Writing
                RETI
S5Act:          RETI
```

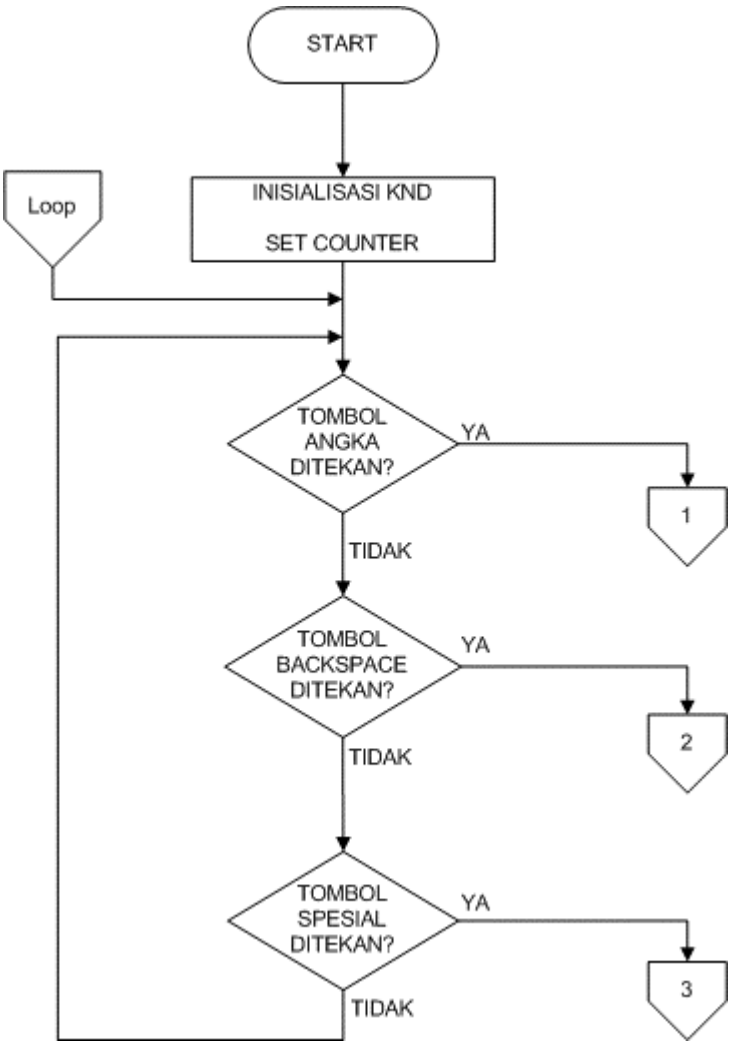
```
S6Act:      RETI
S7Act:      RETI
S8Act:      RETI
S9Act:      RETI
S10Act:     RETI
S11Act:     RETI
S12Act:     RETI

Writing:    LCALL   DisKeyInt
            MOV     R6, #18         ;Loop 18 kali untuk setiap karakter

            MOV     A, #00h         ;Isi ACC dengan karakter yang akan ditampilkan
Loop:       ACALL   Write7S         ;Prosedur untuk menampilkan karakter di 7 segment
            LCALL   LDelay          ;Prosedur delay (terdapat dalam KNDINT.ASM)
            INC     A

            DJNZ    R6, Loop        ;Cek apakah sudah menulis 18 kali? Kalau belum kembali
            LCALL   EnbKeyInt       ;ke Loop
            RET

Start:      LCALL   InitKND         ;Inisialisasi KND
            LCALL   CursorON        ;Menampilkan cursor
            LCALL   VirtualDis      ;Menggunakan tampilan 6 7 Segment
            LCALL   ClrDisMem       ;Hapus memori tampilan sebelumnya
            LCALL   DisKeyInt       ;Mematikan interrupt keypad
            LCALL   Writing         ;Prosedur menulis 18 karakter
            LCALL   EnbKeyInt       ;Mengaktifkan interrupt keypad
            AJMP    $
            END
```

PROGRAM KALKULATOR

Apabila Anda ingin belajar lebih lanjut tentang pemrograman DT-51 KND dengan DT-51 MinSys Ver 3.0, Anda dapat mempelajari program berikut, yaitu program kalkulator sederhana menggunakan DT-51 KND. Layout kalkulator dapat dilihat pada gambar 5. Secara garis besar algoritmanya seperti flowchart pada gambar 6. Flowchart tersebut akan diperjelas pada gambar 7 dan 8.

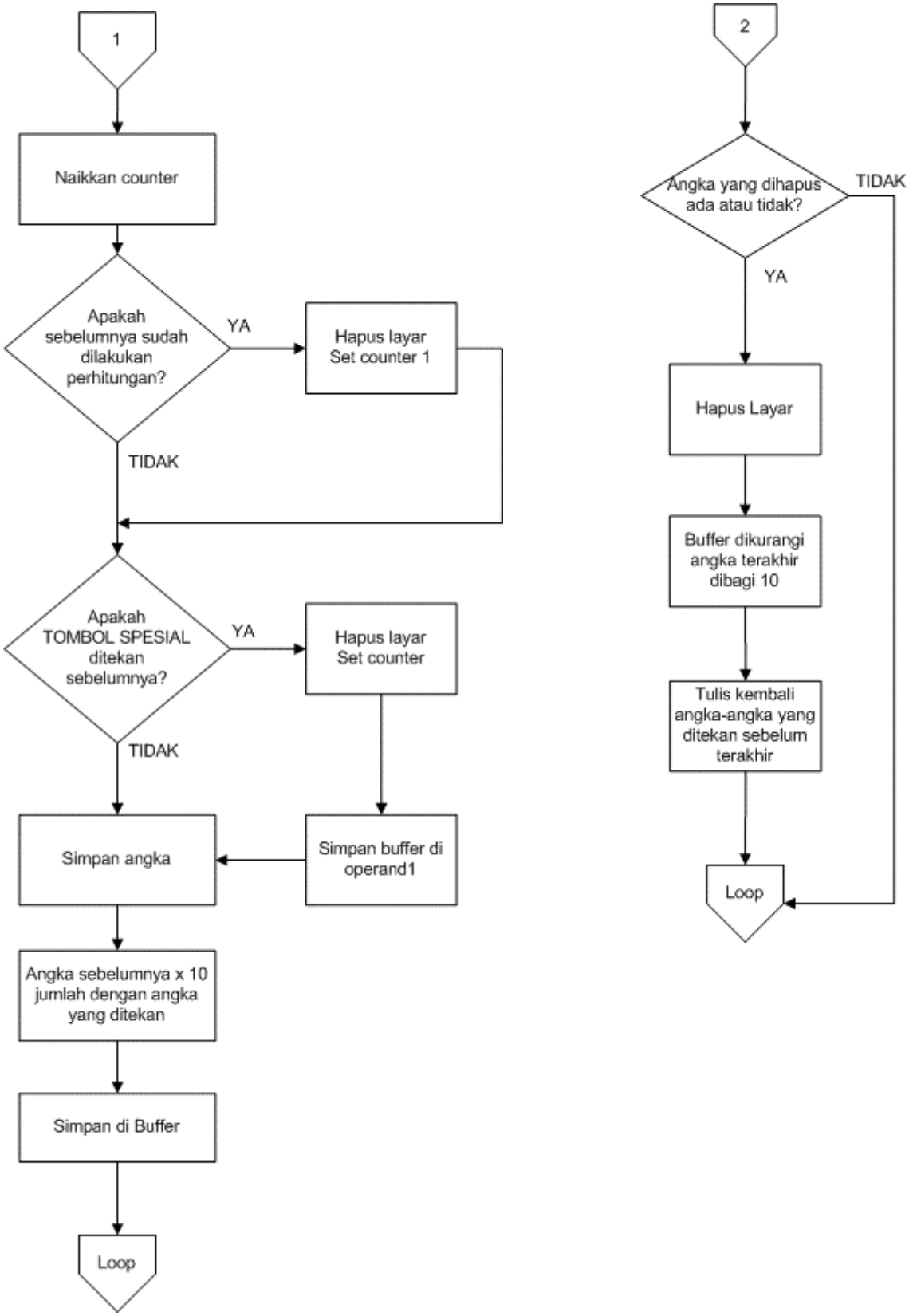


Gambar 5. Layout kalkulator DT-51 KND



Gambar 6. Flowchart program kalkulator secara garis besar

Pertama-tama program akan melakukan insialisasi dan mengisi nilai-nilai awal yang dibutuhkan. Kemudian program akan menunggu penekanan tombol. Tombol 1 sampai 10 adalah angka 0 sampai 9. Sedangkan tombol 11 akan menghapus 1 karakter ke belakang (backspace). Sedangkan tombol 12 (tombol spesial) bila ditekan akan bergantian memberi pilihan proses yang dikehendaki (+,-,x,/). Bila sudah ada 2 operand maka tombol 12 (tombol spesial) bila ditekan akan menampilkan hasil perhitungan (=).



Gambar 7. Flowchart penekanan tombol angka (kiri) dan penekanan backspace (kanan)

Gambar 8. Flowchart penekanan tombol spesial (proses +, -, x, /, atau =)

LISTING PROGRAM (CA.ASM)

```
$MOD51
$TITLE(Example Program Using DT-51 KND Routine)

;****************************************
;*        8 Bit Integer Calculator      *
;****************************************
;Program ini adalah sebuah calculator sederhana dengan menggunakan rutin-rutin yang ada di DT-;51 KND
dan instruksi-instruksi MCS51.
;Program ini dapat menghitung proses-proses aritmatik sederhana seperti tambah, kurang, kali, bagi antar
;bilangan 8 bit (0 - 255) adapun hasilnya adalah integer (bilangan bulat) 16 bit (0 - 65535) untuk bagi
;hasilnya adalah hasil bagi dibulatkan.
;Tombol 1-9: angka 1 s/d 9
;Tombol 10 : angka 0
;Tombol 11 : 'BackSpace'
;Tombol 12 : tombol spesial: toggle antar proses (+, -, x, /, =)
;*************************************************************************************************

CekSP       DATA    0027H
SPbut       bit     CekSP.1
FinishC     bit     CekSP.2
NoBack      bit     CekSP.3
NoSP        bit     CekSP.4
Coun        bit     CekSP.5

LastN       EQU     0028H
Digits      EQU     0029h
Buffer      EQU     0030h
Operand     EQU     0031h
Operand2    EQU     0032H

char1       DATA    0053h
char23      DATA    0054h
char45      DATA    0055h
temh        DATA    0056h
teml        DATA    0057h

            CSEG
            ORG     4000H
            LJMP    Start

            ORG     400BH
            LJMP    Scanning

            ORG     4200H
            $INCLUDE(KNDINT.ASM)

JMPTABLE:   LJMP    S1Act
            LJMP    S2Act
            LJMP    S3Act
            LJMP    S4Act
            LJMP    S5Act
            LJMP    S6Act
            LJMP    S7Act
            LJMP    S8Act
            LJMP    S9Act
            LJMP    S10Act
            LJMP    S11Act
            LJMP    S12Act
```

```
S1Act:  INC     Digits
        LCALL   Done
        LCALL   PushedSP
        MOV     A, #01H
        ACALL   Write7S
        ACALL   SaveNum
        ACALL   CalOp
        AJMP    Loop
S2Act:  INC     Digits
        LCALL   Done
        LCALL   PushedSP
        MOV     A, #02H
        ACALL   Write7S
        LCALL   SaveNum
        LCALL   CalOp
        AJMP    Loop
S3Act:  INC     Digits
        LCALL   Done
        LCALL   PushedSP
        MOV     A, #03H
        ACALL   Write7S
        LCALL   SaveNum
        LCALL   CalOp
        AJMP    Loop
S4Act:  INC     Digits
        LCALL   Done
        LCALL   PushedSP
        MOV     A, #06H
        ACALL   Write7S
        LCALL   SaveNum
        LCALL   CalOp
        AJMP    Loop
S5Act:  INC     Digits
        LCALL   Done
        LCALL   PushedSP
        MOV     A, #05H
        ACALL   Write7S
        LCALL   SaveNum
        LCALL   CalOp
        AJMP    Loop
S6Act:  INC     Digits
        LCALL   Done
        LCALL   PushedSP
        MOV     A, #04H
        ACALL   Write7S
        LCALL   SaveNum
        LCALL   CalOp
        AJMP    Loop
S7Act:  INC     Digits
        LCALL   Done
        LCALL   PushedSP
        MOV     A, #07H
        ACALL   Write7S
        LCALL   SaveNum
        LCALL   CalOp
        AJMP    Loop
```

```
S8Act:      INC     Digits
            LCALL   Done
            LCALL   PushedSP
            MOV     A, #08H
            ACALL   Write7S
            LCALL   SaveNum
            LCALL   CalOp
            AJMP    Loop
S9Act:      INC     Digits
            LCALL   Done
            LCALL   PushedSP
            MOV     A, #09H
            ACALL   Write7S
            LCALL   SaveNum
            LCALL   CalOp
            AJMP    Loop
S10Act:     INC     Digits
            LCALL   Done
            LCALL   PushedSP
            MOV     A, #00H
            ACALL   Write7S
            LCALL   SaveNum
            LCALL   CalOp
            AJMP    Loop

S11Act:     JNB     NoBack,Nothing

            ACALL   ClrDisMem
            CJNE    R5, #0, BB
            ACALL   ClrDisMem
            MOV     Buffer, #0
            AJMP    Loop

BB:         MOV     R1, #33h

            DEC     R5              ;jumlah penekanan tombol dikurangi satu
            MOV     A, R5
            MOV     R6, A

            CJNE    R5, #0, WAgain
            ACALL   ClrDisMem
            MOV     Buffer, #0
Nothing:    AJMP    Loop

WAgain:     MOV     A, @R1
            ACALL   Write7S
            INC     R1
            DJNZ    R6, WAgain
            MOV     Digits, R5

            LCALL   ClearBuf
            SETB    NoSP
            AJMP    Loop

S12Act:     JNB     NoSP, CL
            ACALL   ClrDisMem
            JB      Coun, Counting
            JB      SPbut, SkipSP
            SETB    SPbut
SkipSP:     INC     R4
```

```
C0:         CJNE    R4, #1, C1      ;Cek berapa kali penekanan tombol
            ACALL   WAdd
C1:         CJNE    R4, #2, C2
            ACALL   WSub
C2:         CJNE    R4, #3, C3
            ACALL   WMul
C3:         CJNE    R4, #4, C4
            ACALL   WDiv
C4:         CJNE    R4, #5, CL
            MOV     R4, #1
            SJMP    C0
CL:         AJMP    Loop

Counting:   CJNE    R4, #1, Pr1     ;Cek berapa kali penekanan tombol
            ACALL   Adding
            MOV     teml, A
            LCALL   Display
Pr1:        CJNE    R4, #2, Pr2
            ACALL   Subing
            MOV     temh, #0
            MOV     teml, A
            LCALL   Display
Pr2:        CJNE    R4, #3, Pr3
            ACALL   Muling
            MOV     temh, B
            MOV     teml, A
            LCALL   Display
Pr3:        CJNE    R4, #4, Pr4
            ACALL   Diving
            MOV     temh, #0
            MOV     teml, A
            LCALL   Display
Pr4:        MOV     R4, #0
            SETB    FinishC
            AJMP    Loop
;-------------------------------------------------------------------------------
;Prosedur buat menormalkan buffer di BackSpace
ClearBuf:   PUSH    ACC
            MOV     A, Buffer
            SUBB    A, LastN        ;Buffer dikurangi dengan angka terakhir
            MOV     B, #10          ;Bagi 10
            DIV     AB
            MOV     Buffer, A
            POP     ACC
            MOV     A, #0
            RET
;-------------------------------------------------------------------------------
SaveNum:    MOV     @R1, A
            INC     R1
            MOV     R5, DIGITS
            RET
;-------------------------------------------------------------------------------
;Periksa apakah sudah selesai menghitung?
Done:       JNB     FinishC, yup
            ACALL   ClrDisMem
            LCALL   SetReg
            MOV     Digits, #1
yup:        RET
;-------------------------------------------------------------------------------
;Periksa apa sudah menekan SPbut?
```

```
PushedSP:   JNB     SPbut, Nope
            ACALL   ClrDisMem
            MOV     Operand, Buffer
            MOV     Digits, #1
            MOV     Buffer, #0
            MOV     R1, #33h
            SETB    Coun
            CLR     SPbut
Nope:       SETB    NoBack
            SETB    NoSP
            RET
;-------------------------------------------------------------------------------
;Prosedur untuk menghitung operand
CalOp:      PUSH    ACC
            MOV     A, Buffer
            MOV     B, #10
            MUL     AB              ;KALI 10
            JNB     PSW.2, NoOV     ;Cek operand overflow?
            ACALL   ClrDisMem       ;jika OV hapus display
            MOV     Digits, #0h     ;tulis lagi dari digit 1
            MOV     R1, #33h
            POP     ACC
            MOV     A, #0
            MOV     Buffer, #0
            SJMP    ba
NoOV:       MOV     Buffer, A       ;angka terakhir dengan angka sebelumnya dijumlah
            POP     ACC
            MOV     LastN, A
            ADD     A, Buffer
            JNC     NoOV2
            ACALL   ClrDisMem       ;jika OV hapus display
            MOV     Digits, #0h     ;tulis lagi dari digit 1
            MOV     R1, #33h
            MOV     A, #0
            MOV     Buffer, #0
            SJMP    ba
NoOV2:      MOV     Buffer, A       ;Simpan di Buffer
ba:         RET
;-------------------------------------------------------------------------------
;PROSEDUR-PROSEDUR ARITMATIK
Adding:     MOV     Operand2, Buffer
            MOV     A, Operand
            ADD     A, Operand2
            JNC     Much
            MOV     temh, #1
            SJMP    BC
Much:       MOV     temh, #0
BC:         RET
Subing:     MOV     Operand2, Buffer
            MOV     A, Operand
            SUBB    A, Operand2
RevO:       RET
Muling:     MOV     Operand2, Buffer
            MOV     A, Operand
            MOV     B, Operand2
            MUL     AB
            RET
Diving:     MOV     Operand2, Buffer
            MOV     A, Operand
            MOV     B, Operand2
```

```
                DIV     AB
Revl:           RET
;-------------------------------------------------------------------------------
;Prosedur Display
Display:        LCALL   BCDCONVERT

                LCALL   ClrDisMem
                MOV     A, char1
                ANL     A, #00001111B
                LCALL   WRITE7S
                MOV     A, char23
                ANL     A, #11110000B
                SWAP    A
                LCALL   WRITE7S
                MOV     A, char23
                ANL     A, #00001111B
                LCALL   WRITE7S
                MOV     A, char45
                ANL     A, #11110000B
                SWAP    A
                LCALL   WRITE7S
                MOV     A, char45
                ANL     A, #00001111B
                LCALL   WRITE7S
                RET
;-------------------------------------------------------------------------------
BCDCONVERT:
                MOV     char1, #0h
                MOV     char23, #0h
                MOV     char45, #0h

over:           DEC     teml
                MOV     a, teml
                CJNE    a, #0ffh, yoi
                DEC     temh
                MOV     a, temh
                CJNE    a, #0ffh, yoi
                RET

yoi:            MOV     a, char45
                ADD     a, #01
                DA      a
                MOV     char45, a
                JNC     lanjut
                MOV     a, char23
                ADD     a, #01
                DA      a
                MOV     char23, a
                JNC     lanjut
                MOV     a, char1
                ADD     a, #01
                MOV     char1, a
lanjut:         SJMP    over
;-------------------------------------------------------------------------------
;Prosedur mengisi register
SetReg:         MOV     Operand, #0h
                MOV     Operand2, #0h
                MOV     CekSP, #0H
                MOV     Buffer, #0h
                MOV     Digits, #0H
```

```
                MOV     R4, #0H
                MOV     R1, #33h
                RET
;-------------------------------------------------------------------------------
Greet:          MOV     A, #0Ch         ;Isi ACC dengan karakter "C"
                ACALL   Write7S         ;Prosedur untuk menampilkan karakter di 7 segment
                MOV     A, #0Bh         ;Isi ACC dengan karakter "A"
                ACALL   Write7S         ;Prosedur untuk menampilkan karakter di 7 segment
                MOV     A, #14h         ;Isi ACC dengan karakter "L"
                ACALL   Write7S         ;Prosedur untuk menampilkan karakter di 7 segment
                MOV     A, #0Ch         ;Isi ACC dengan karakter "C"
                ACALL   Write7S         ;Prosedur untuk menampilkan karakter di 7 segment
                RET
;-------------------------------------------------------------------------------
WAdd:           MOV     A, #0Bh         ;Isi ACC dengan karakter "A"
                ACALL   Write7S         ;Prosedur untuk menampilkan karakter di 7 segment
                MOV     A, #0Dh         ;Isi ACC dengan karakter "d"
                ACALL   Write7S         ;Prosedur untuk menampilkan karakter di 7 segment
                MOV     A, #0Dh         ;Isi ACC dengan karakter "d"
                ACALL   Write7S         ;Prosedur untuk menampilkan karakter di 7 segment
                RET
;-------------------------------------------------------------------------------
WSub:           MOV     A, #05h         ;Isi ACC dengan karakter "S"
                ACALL   Write7S         ;Prosedur untuk menampilkan karakter di 7 segment
                MOV     A, #17h         ;Isi ACC dengan karakter "U"
                ACALL   Write7S         ;Prosedur untuk menampilkan karakter di 7 segment
                MOV     A, #08h         ;Isi ACC dengan karakter "B"
                ACALL   Write7S         ;Prosedur untuk menampilkan karakter di 7 segment
                RET
;-------------------------------------------------------------------------------
WMul:           MOV     A, #11h         ;Isi ACC dengan karakter "M"
                ACALL   Write7S         ;Prosedur untuk menampilkan karakter di 7 segment
                MOV     A, #17h         ;Isi ACC dengan karakter "U"
                ACALL   Write7S         ;Prosedur untuk menampilkan karakter di 7 segment
                MOV     A, #14h         ;Isi ACC dengan karakter "L"
                ACALL   Write7S         ;Prosedur untuk menampilkan karakter di 7 segment
                RET
;-------------------------------------------------------------------------------
WDiv:           MOV     A, #0Dh         ;Isi ACC dengan karakter "D"
                ACALL   Write7S         ;Prosedur untuk menampilkan karakter di 7 segment
                MOV     A, #12h         ;Isi ACC dengan karakter "i"
                ACALL   Write7S         ;Prosedur untuk menampilkan karakter di 7 segment
                MOV     A, #17h         ;Isi ACC dengan karakter "V"
                ACALL   Write7S         ;Prosedur untuk menampilkan karakter di 7 segment
                RET
;-------------------------------------------------------------------------------
;AWAL PROGRAM
Start:          MOV     SP, #58H        ;Menentukan posisi StackPointer
                ACALL   InitKND
                ACALL   RealDis
                ACALL   CursorON
                ACALL   ClrDisMem
                ACALL   Greet
                ACALL   LDelay
                ACALL   LDelay
                ACALL   LDelay
                ACALL   ClrDisMem
                LCALL   SetReg
;Scanning tombol
Loop:           JNB     KeyPressed, $
```

```
CLR     KeyPressed
MOV     A, KeyCode
MOV     DPTR, #JMPTABLE
DEC     A
MOV     B, #3
MUL     AB
JMP     @A+DPTR
END
```

TROUBLESHOOTING

Q: Kenapa KND saya tidak mau berfungsi sama sekali?
A: Periksa polaritas tegangan pada KND. Jangan sampai terbalik antara kutub positif dan negatif.

Q: Kenapa 7 segment-nya menyala semua saat menjalankan program?
A: Periksa kabel penghubung PORTA & PORTB apakah sudah terhubung dengan benar (lihat petunjuk pemasangan).

Q: Kenapa keypad saya tidak berfungsi?
A: Periksa kabel penghubung PORTC & PORT1 apakah sudah terhubung dengan benar (lihat petunjuk pemasangan).

Q: Kenapa saya tidak bisa menggunakan rutin-rutin yang disediakan seperti initKND, Write7S, dll?
A: Periksa apakah Anda sudah memasukkan file KNDINT.ASM atau KNDEXT.ASM pada program Anda (lihat contoh program)

Q: Kenapa tampilan 7 segment saya pertama-tama stabil tapi lama kelamaan kacau?
A: Apabila Anda menggunakan memori eksternal (file KNDEXT.ASM) maka delay-nya kurang lama untuk mengisi EEPROM

Q: Kenapa program saya tiba-tiba membeku/hang?
A: Cek dulu apakah anda telah menetapkan alamat stack pointer untuk menentukan tingkat stack

Q: Kenapa keypad saya tidak mau berfungsi sebagai interrupt setelah program saya berjalan?
A: Periksa apakah interrupt keypad telah diaktifkan dengan menggunakan rutin EnbKeyInt.

Q: Kenapa program saya berhenti setelah terjadi interrupt?
A: Periksa apakah anda sudah memberi perintah untuk kembali ke program utama dengan memberi perintah RETI pada akhir prosedur interrupt.

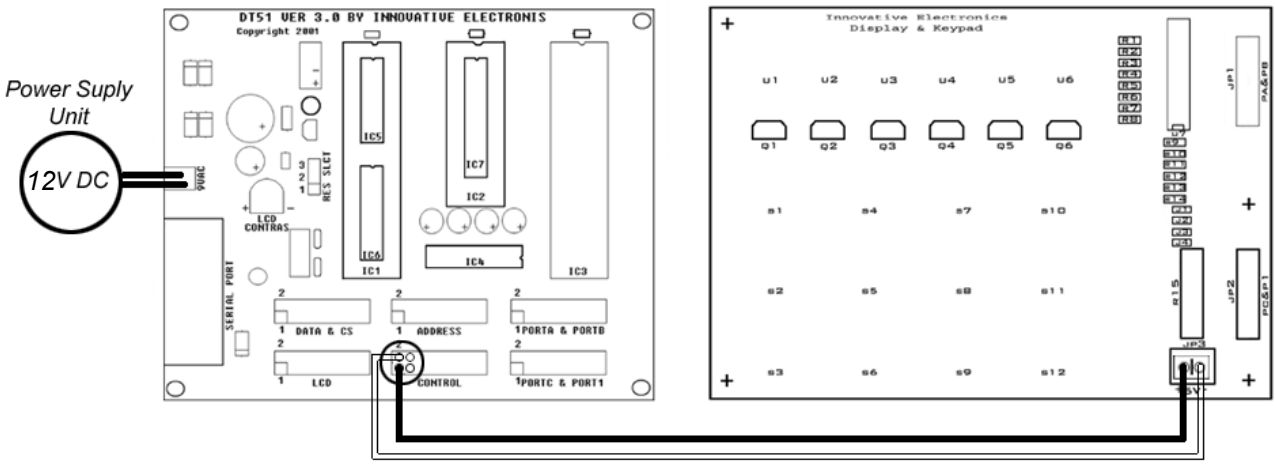
Q: Mengapa salah satu 7 segment atau salah satu segment dari 7 segment tidak menyala?
A: Ada kemungkinan salah satu kabel putus atau hubungan kabel salah. Coba jalankan program TESKND.EXE (terdapat pada disket DT-51 KND) sambil memeriksa hubungan kabel dan kabelnya

Q: Mengapa display 7 segment tidak menyala sama sekali?
A: Ada kemungkinan salah satu kabel putus atau hubungan kabel salah. Coba jalankan program TESKND.EXE sambil memeriksa hubungan kabel dan kabelnya.

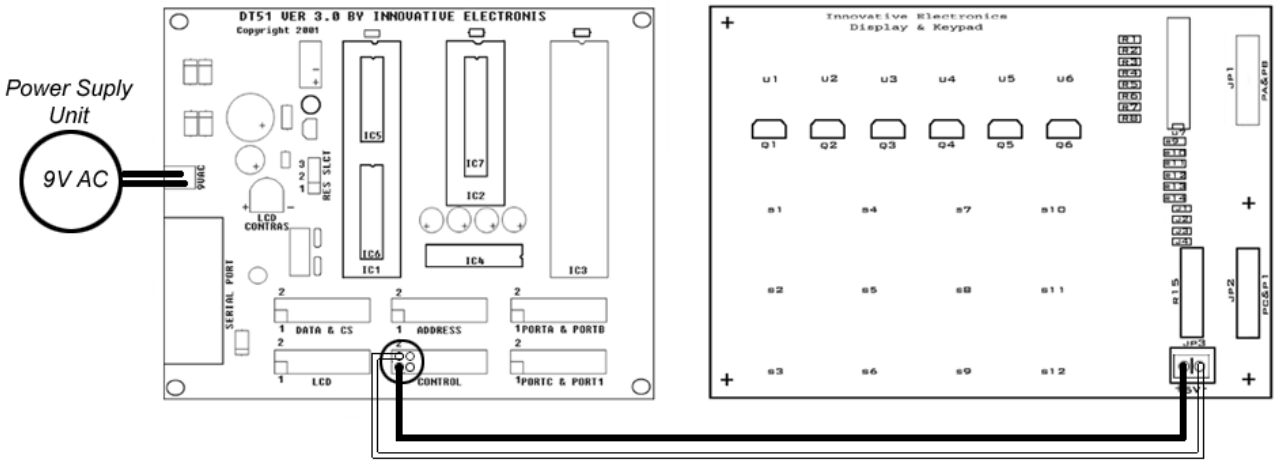
Q: Mengapa keypadnya tidak berfungsi?
A: Ada kemungkinan salah satu kabel putus atau hubungan kabel salah. Coba jalankan program TESKND.EXE sambil memeriksa hubungan kabel dan kabelnya.

Q: Mengapa display 7 segment-nya membayang?
A: Hal itu disebabkan ground antara DT-51 MinSys Ver 3.0 dan DT-51 KND tidak terhubung (referensi ground tidak sama).

Perhatikan cara menghubungkan yang benar. Lihat contoh pada gambar 9 dan 10.

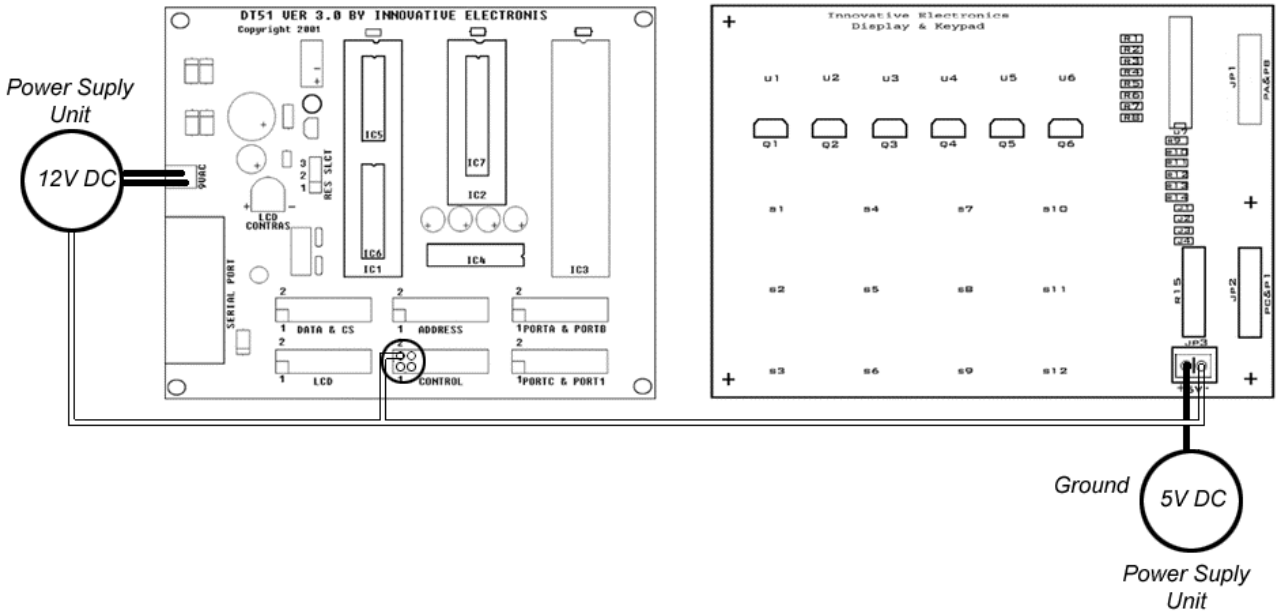


Gambar 9. Menghubungkan DT-51 KND dan DT-51 MinSys ver. 3.0 (melalui Port Control) dengan sumber tegangan 12 V DC



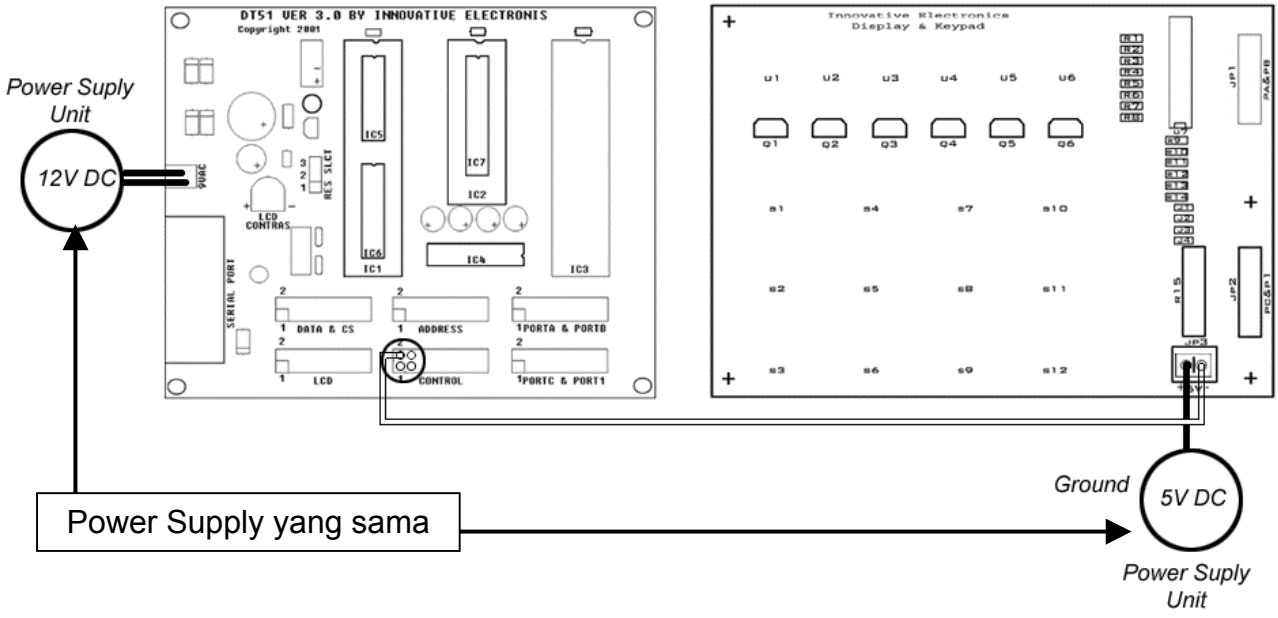
Gambar 10. Menghubungkan DT-51 KND dan DT-51 MinSys ver. 3.0 (melalui Port Control) dengan sumber tegangan 9 V AC

Tetapi jangan dihubungkan seperti gambar 11.



Gambar 11. DT-51 KND dan DT-51 MinSys ver. 3.0 dengan sumber tegangan 12 V DC dan 5 V DC dari power supply yang sama/berbeda

Hati-hati jika menghubungkannya seperti pada gambar 12.



Gambar 12. DT-51 KND dan DT-51 MinSys ver. 3.0 dengan sumber tegangan 12 V DC dan 5 V DC dari power supply yang sama

Listing semua program dapat dilihat pada **AN14.ZIP**.