

# DT-51

## DT-51 *Application Note*

### AN41 – Electronic Puzzle

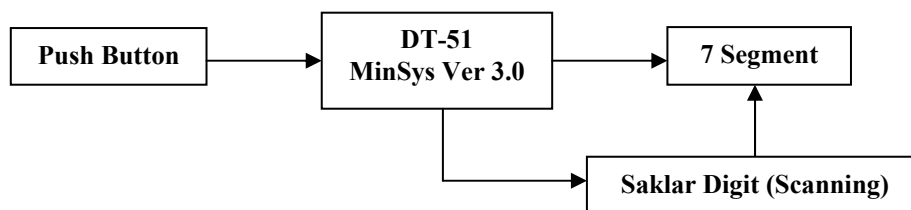
Oleh: Time IE & Gatut Eko Daryani  
(Universitas Katholik Widya Mandala)

Aplikasi ini dirancang sebagai permainan puzzle elektronik 3 x 3. Sistem ini menggunakan modul DT51 *MinSys* Ver. 3.0, *Pushbutton* dan *Seven Segment*. Metode Random yang digunakan pada aplikasi ini adalah metode *linear congruential pseudo-random number generator*.

Modul yang digunakan adalah:

- 9 Seven Segment
- 9 Transistor 2N3906
- 9 Resistor 2K2 Ohm
- 7 Resistor 180 Ohm
- 5 Tactile Switch (Push button)
- 1 Resistor 1K Ohm
- 1 modul DT-51 *MinSys* Ver. 3.0.

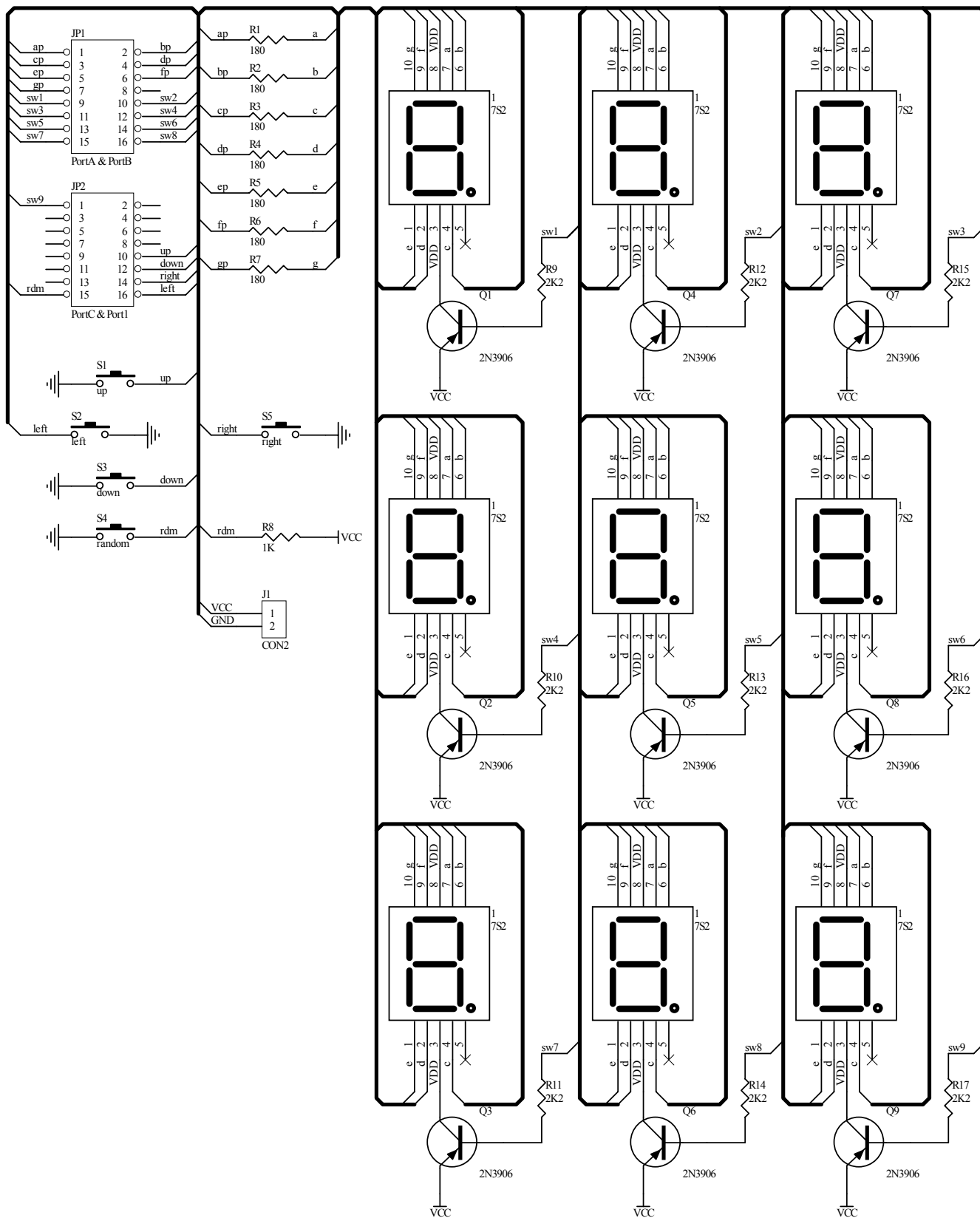
Adapun blok diagram sistem secara keseluruhan adalah sebagai berikut:



**Gambar 1**  
**Blok Diagram AN41**

Adapun Saklar Digit berfungsi dalam teknik scanning untuk tampilan 7 segment.

Skema rangkaian adalah sebagai berikut:



**Gambar 2**  
**Skema Puzzle**

Hubungan antara DT-51 MinSys Ver 3.0, 7 Segment, dan Push Button adalah sebagai berikut:

7 Segment	DT-51 MinSys Ver 3.0
Data	Port A
Saklar 1 - 8	Port B
Saklar 9	Port C (PC.0)

**Tabel 1**  
Hubungan DT-51 MinSys dengan 7 Segment

Push Button	DT-51 MinSys Ver 3.0
Random	P1.6
Kanan	P1.5
Kiri	P1.7
Atas	P1.1
Bawah	P1.3

**Tabel 2**  
Hubungan DT-51 MinSys dengan Push Button

Setelah menghubungkan rangkaian dan menghubungkan supply tegangan yang tepat, *download*-lah program RANDOM2.HEX ke DT-51 MinSys Ver. 3.0.

## Metode Random

Karena begitu banyak metode random number generator, maka kita perlu mencari sebuah metode sederhana yang bisa diterjemahkan ke dalam bahasa assembly. Maka setelah mempelajari beberapa metode, penulis memutuskan untuk menggunakan metode Linear Congruential Pseudo-Random Number Generator. Adapun fungsinya adalah seperti dibawah ini:

$$G(x) = (Cx + D) \bmod M$$

dimana  $C, D$  dan  $M$  adalah konstan.

dengan memilih sebuah angka awal  $X_0$ , kita bisa menghasilkan sebuah deret angka  $X_0, X_1, X_2, \dots$  dengan memperhatikan hal sebagai berikut:

$$X_{n+1} = G(X_n)$$

Contoh:  $M = 9, C = 1, D = 8, X_0 = 3$ . Kemudian,

$$G(x) = (x + 8) \bmod 9$$

dengan menggunakan rumus diatas kita peroleh:

$$X_1 = [(3) + 8] \bmod 9 = 2$$

$$X_2 = [(2) + 8] \bmod 9 = 1$$

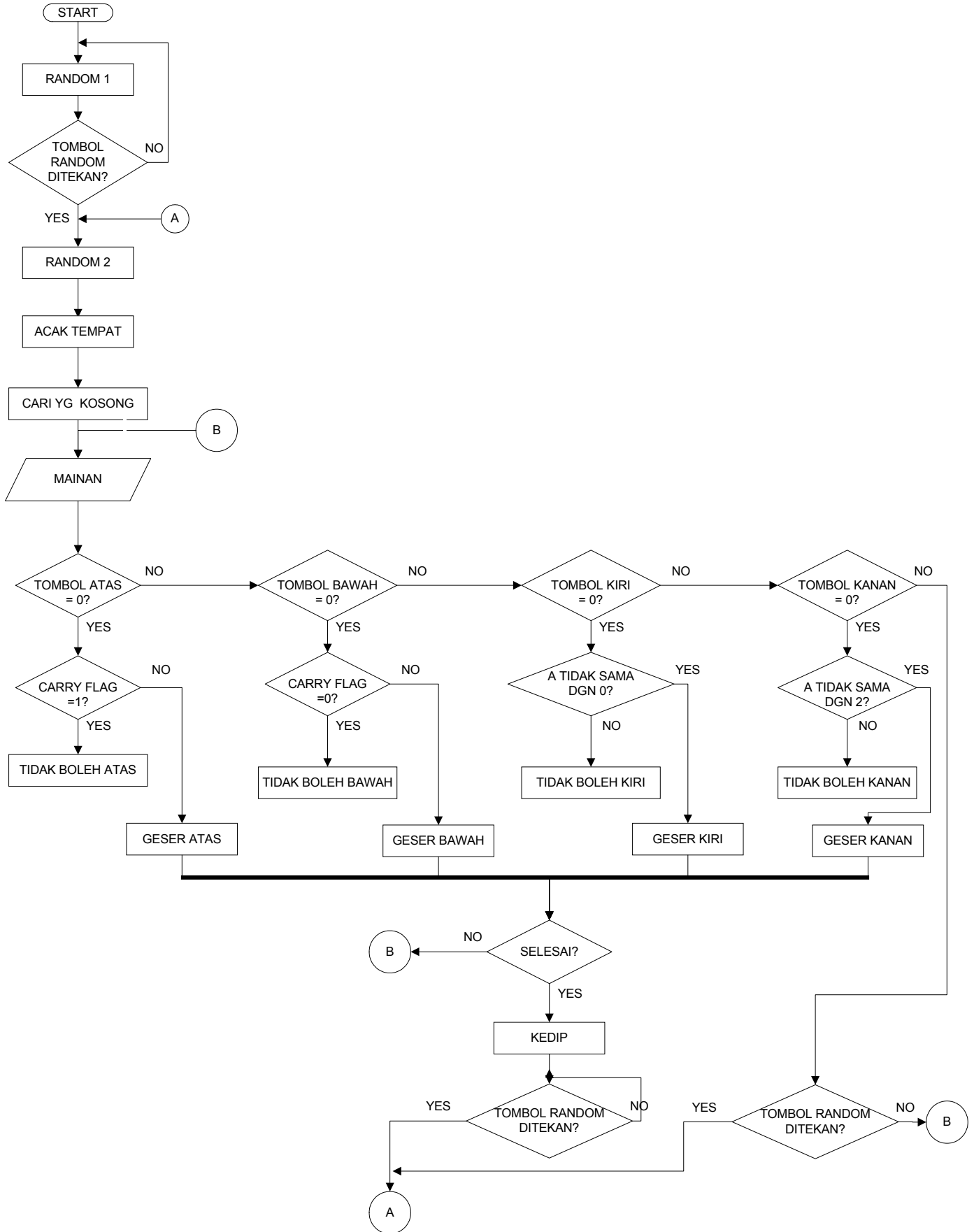
$$X_3 = [(1) + 8] \bmod 9 = 0$$

$$X_4 = [(0) + 8] \bmod 9 = 8$$

Apabila kita melanjutkan perhitungan diatas kita akan mendapatkan  $X_5 = 7, X_6 = 6, X_7 = 5, X_8 = 4, X_9 = 3$ . Pada  $X_9 = X_0$ , deret akan kembali memulai dari awal dan mengulangi 9 nilai sama seterusnya.

Satu hal yang perlu diberi catatan untuk contoh ini bahwa masing-masing nilai dari  $\{ 0..8 \}$  terjadi sebelum deret mulai berulang. Untuk memastikan hal ini, maka nilai  $C, D, M$  harus dipilih secara benar.

Flowchart dari sistem ini adalah sebagai berikut:



Gambar 3  
Flowchart Program

**P**rogram akan diproses dengan urutan sebagai berikut:

1. Pada awal program akan dilakukan inisialisasi terhadap PPI.
2. Kemudian program akan melakukan proses random pertama.
3. Berikutnya program akan memeriksa penekanan tombol "random". Jika ada penekanan tombol tersebut, program akan melakukan satu kali random sebelum menampilkan hasil random pada 7 segment.
4. Program akan melakukan pemeriksaan semua tombol.

Karena masing-masing 7 segment memiliki alamat, maka saat sebuah tombol ditekan, posisi kotak kosong diperiksa. Dengan memanfaatkan carry flag, program akan memeriksa apakah kotak kosong boleh digeser. Jika posisi kotak kosong sudah di baris atas, tentunya tidak mungkin jika kotak kosong tersebut digeser lagi ke atas (dengan menekan tombol bawah). Hal ini berlaku untuk penekanan tombol atas, bawah, kiri, dan kanan. Jika tombol random yang ditekan, otomatis program akan melakukan random lagi dan memulai permainan dari awal.

3	4	6
	2	8
5	1	7

(a)

	4	6
3	2	8
5	1	7

(b)

3	4	6
2		8
5	1	7

(c)

**Gambar 4**

**Berbagai Respon Tampilan 7 Segment Saat Penekanan Tombol**

a) Posisi awal

b) Penekanan tombol bawah

c) Penekanan tombol kiri

5. Setelah itu, program akan melakukan pemeriksaan kondisi permainan, apakah angkanya sudahurut (permainan diselesaikan). Jika belum, program maka akan kembali ke langkah 4. Jika permainan sudah selesai, tampilan akan berkedip sejenak lalu

Kesimpulan dan saran:

1. Alat ini masih kurang sempurna karena masih memiliki *bouncing*.
2. Variasi yang dimiliki oleh permainan ini hanya 9 variasi. Dan ada beberapa variasi yang tidak bisa diselesaikan, hal ini disebabkan adanya penukaran angka pada suatu alamat yang tidak tepat.
3. Alat ini bisa dikembangkan menjadi permainan yang lebih kompleks dengan meningkatkan menjadi 4 x 4 , 8 x 8 , dan lain-lain. Dengan menambahkan IC *Latch*.
4. Selain itu pula, kita juga bisa menggunakan metode *random* yang lain. Yang memiliki hasil *random* yang lebih banyak & lebih tidak terduga.

**L**isting program **RANDOM2.ASM** terdapat pada **AN41.Zip**.

**S**elamat berinovasi!