DT-AVR

DT-AVR *Application Note*

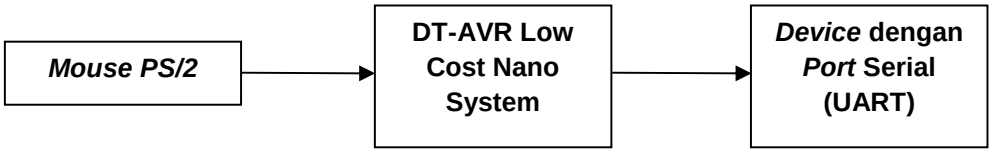
AN153 – Konversi protokol PS/2 menjadi protokol UART

Oleh: Tim IE dan Nathanael R.A.

M*ouse PS/2* merupakan sebuah *input device* yang menggunakan protokol PS/2 untuk berkomunikasi dengan *device* lain seperti PC. Artikel kali ini akan membahas cara membaca data dan mengirimkan perintah ke *mouse PS/2* dengan menggunakan emulasi protokol PS/2 dan kemudian merubah protokol PS/2 menjadi UART untuk dihubungkan dengan device lain yang hanya memiliki komunikasi UART. Aplikasi ini dikembangkan dengan menggunakan basaha pemrograman *assembly*.
Aplikasi ini memerlukan modul dan komponen sebagai berikut:

- DT-AVR Low Cost Nano System
- Konektor *Mini-DIN* 6 pin
- *Mouse PS/2*

Adapun blok diagram sistem secara keseluruhan adalah sebagai berikut:



Gambar 1
Blok Diagram AN153

Hubungan antara *Device* dengan *Port* Serial (UART) dengan DT-AVR Low Cost Nano System adalah sebagai berikut:

***Device* dengan *Port* Serial (UART)**	**DT-AVR Low Cost Nano System**
RXD	TXD (J8 Pin 4)
GND	GND (J8 Pin 1)

Tabel 1
Hubungan antara *Device* dengan *Port* Serial (UART) dengan DT-AVR Low Cost Nano System

Komunikasi *serial* antara DT-AVR Low Cost Nano System dengan *device* lain dengan *port* serial (UART) hanya bersifat satu arah saja, sehingga hanya diperlukan satu pin saja (pin TXD pada sisi pengirim, dan pin RXD pada sisi penerima).

Hubungan antara *Mouse PS/2* dengan DT-AVR Low Cost Nano System adalah sebagai berikut:

DT-AVR Low Cost Nano System	***Mouse PS/2* (Konektor *Mini-DIN* 6 pin)**
VCC (J7 Pin 2)	VCC (Pin 4)
GND (J7 Pin 1)	GND (Pin 3)
PB.0 (J7 Pin 3)	CLOCK (Pin 5)
PB.1 (J7 Pin 4)	DATA (Pin 1)

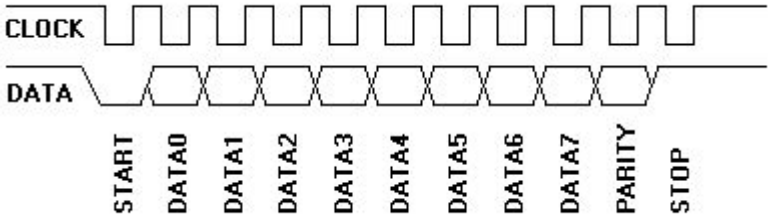
Tabel 2
Hubungan antara DT-AVR Low Cost Nano System dengan *Mouse PS/2* (Konektor *Mini-DIN* 6 pin)

Output dari mouse PS/2 bersifat *open-collector* sedangkan pada setiap *port* pada mikrokontroler AVR sudah dilengkapi dengan resistor *pull-up* internal, oleh sebab itu tidak diperlukan lagi tambahan resistor *pull-up*.

Dasar teori untuk komunikasi dengan *mouse* adalah menggunakan protokol PS/2. *Device* PS/2 (seperti *keyboard* dan *mouse*) menggunakan protokol *serial* dengan 11 bit data. Bit-bit tersebut adalah:

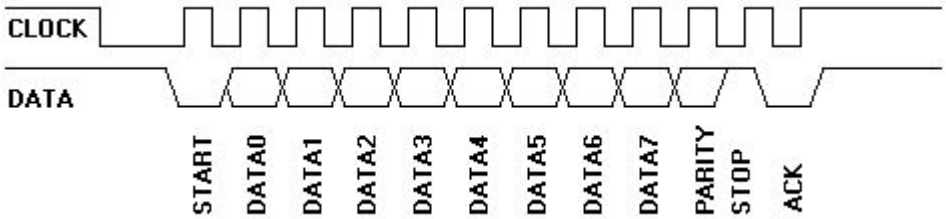
- 1 *start* bit. Bit ini selalu bernilai 0.
- 8 data bit, *Least Significant Bit* dikirim lebih dulu.
- 1 *parity* bit (*odd parity*).
- 1 *stop* bit. Bit ini selalu bernilai 1.

Untuk lebih jelasnya, gambar di bawah ini adalah *timing diagram* dari komunikasi *device* ke *host* (mikrokontroler):



Gambar 2
Komunikasi dari *Device* ke *Host*

Sedangkan komunikasi dari *host* ke *device* dijelaskan dalam gambar berikut:



Gambar 3
Komunikasi dari *Host* ke *Device*

Pada komunikasi dari *host* ke *device*, terdapat 1 bit tambahan, yaitu ACK. Bit ini dikirimkan oleh *device* yang menandakan keseluruhan data sudah diterima dengan baik.

Tiap data yang dikirimkan oleh *device* mengandung informasi tertentu yang menyatakan kondisi dari *device*. Khusus untuk *mouse* dalam mode *scroll*, data yang diterima dari *device* berupa 4 byte data secara berturut-turut, sebagai berikut:

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 1	Y overflow	X overflow	Y sign bit	X sign bit	Always 1	Middle Btn	Right Btn	Left Btn
Byte 2	X Movement							
Byte 3	Y Movement							
Byte 4	Z Movement							

Tabel 3
Data yang Diterima dari *Mouse*

Keterangan:

- Y overflow dan/atau X overflow akan diset 1 apabila *counter* melebihi batas maksimal penghitungan (*range counter*: -255 sampai +255)
- Y sign bit dan X sign bit menandakan arah pergerakan *mouse* ke kuadran negatif atau positif.
- Middle Button, Right Button, dan Left Button akan diset 1 apabila ada penekanan tombol tersebut pada *mouse*
- X Movement menandakan banyaknya pergerakan *mouse* dalam sumbu X
- Y Movement menandakan banyaknya pergerakan *mouse* dalam sumbu Y
- Z Movement menandakan banyaknya *scroll* pada *mouse*

Pada saat *mouse* pertama kali dinyalakan, *mouse* akan masuk ke *reset mode* dan melakukan *self-diagnostic*, kemudian mengatur beberapa nilai sebagai berikut:

- *Sample rate* = 100 samples/sec
- *Resolution* = 4 counts/mm
- *Scaling* = 1:1
- *Data Reporting Disabled*

Setelah *self-diagnostic* selesai, *mouse* akan mengirimkan *BAT (Basic Assurance Test) Code*, diikuti dengan *Device ID*. Apabila *BAT Code* bernilai AAh (*BAT Successful*) berarti *mouse* berada dalam kondisi baik. Tetapi apabila *BAT Code* bernilai FCh (*Error*), maka kemungkinan *mouse* mengalami kerusakan.

Mouse juga mempunyai *command set*, yang dapat digunakan untuk memberikan perintah kepada *mouse* untuk melakukan sesuatu. *Command set* yang dapat dimengerti oleh *mouse* antara lain:

- FFh (*Reset*): Perintah ini digunakan untuk me-*reset mouse*. *Mouse* akan membalas dengan "*acknowledge*" (FAh) kemudian masuk *Reset Mode*.
- FEh (*Resend*): Perintah ini digunakan oleh *host* untuk meminta *mouse* mengirimkan data yang terakhir apabila *host* menerima data yang tidak benar dari *mouse*.
- F6h (*Set Defaults*): Perintah ini digunakan untuk mengembalikan nilai *setting mouse* ke posisi *default*. *Mouse* akan membalas dengan "*acknowledge*" (FAh), kemudian mengatur nilai-nilai berikut: *Sample rate* = 100 samples/sec, *Resolution* = 4 counts/mm, *Scaling* = 1:1, *Data Reporting Disabled*. Setelah itu, *mouse* akan me-*reset counter* gerakan dan masuk ke *Stream Mode*.
- F5h (*Disable Data Reporting*): Perintah ini digunakan untuk mematikan *data reporting* pada *mouse*. *Mouse* akan membalas dengan "*acknowledge*" (FAh) kemudian mematikan *data reporting* dan me-*reset counter* gerakan.
- F4h (*Enable Data Reporting*): Perintah ini digunakan untuk mengaktifkan *data reporting* pada *mouse*, sehingga *mouse* selalu melaporkan perubahan data setiap kali ada aktivitas. Mouse akan membalas dengan "*acknowledge*" (FAh) kemudian mengaktifkan *data reporting* dan me-*reset counter* gerakan.
- F3h (*Set Sample Rate*): Perintah ini digunakan untuk menentukan *sample rate* yang akan dipakai oleh *mouse*. *Mouse* akan membalas dengan "*acknowledge*" (FAh) kemudian meminta satu byte data lagi sebagai nilai *sample rate*-nya. Setelah itu, *mouse* akan membalas lagi dengan "*acknowledge*" (FAh) kemudian *counter* gerakan akan di-*reset*. *Sample rate* yang dapat dipakai adalah: 10, 20, 40 60, 80, 100, dan 200 *samples/sec*.
- F2h (*Get Device ID*): Perintah ini digunakan untuk membaca *device ID* dari *mouse*. *Mouse* akan membalas dengan "*acknowledge*" (FAh) diikuti dengan *device ID*-nya. Apabila *device ID* bernilai 00h, maka *mouse* tersebut adalah *mouse* standar (*mouse* dengan 3 tombol tanpa *scroll*). Apabila *device ID* bernilai 03h, maka *mouse* tersebut mempunyai "*scrolling wheel*". Apabila *device ID* bernilai 04h, maka *mouse* tersebut adalah *mouse* dengan "*scrolling wheel + 5 button*".
- F0h (Set Remote Mode): Perintah ini digunakan untuk masuk ke *Remote Mode*. *Mouse* akan membalas dengan "*acknowledge*" (FAh) kemudian masuk ke *Remote Mode*.
- EEh (Set Wrap Mode): Perintah ini digunakan untuk masuk ke Wrap Mode. *Mouse* akan membalas dengan "*acknowledge*" (FAh) kemudian masuk ke *Wrap Mode*.
- ECh (Reset Wrap Mode): Perintah ini digunakan untuk keluar dari Wrap Mode. *Mouse* akan membalas dengan "*acknowledge*" (FAh) kemudian me-*reset counter* gerakan dan masuk ke *mode* terakhir sebelum *Wrap Mode*.
- EBh (Read Data): Perintah ini digunakan untuk membaca data dalam *Remote Mode*. *Mouse* akan membalas dengan "*acknowledge*" (FAh) kemudian mengirimkan data gerakan. Setelah data berhasil dikirimkan, mouse me-*reset counter* gerakan.
- EAh (Set Stream Mode): Perintah ini digunakan untuk masuk ke *Stream Mode*. *Mouse* akan membalas dengan "*acknowledge*" (FAh) kemudian masuk ke *Stream Mode*.
- E9h (Status Request): Perintah ini digunakan untuk meminta status terakhir dari mouse yang berupa 3 bit data status. *Mouse* akan membalas dengan "*acknowledge*" (FAh) kemudian mengirimkan statusnya dan me-*reset counter* gerakan.
- E8h (Set Resolution): Perintah ini digunakan untuk menentukan resolusi yang akan dipakai oleh *mouse*. *Mouse* akan membalas dengan "*acknowledge*" (FAh) kemudian menunggu 1 byte data lagi yang merupakan data resolusi yang akan dipakai, kemudian *mouse* akan membalas dengan "*acknowledge*" (FAh) lagi.
- E7h (Set Scaling 2:1): Perintah ini digunakan untuk mengatur scaling pada mouse menjadi 2:1.
- E6h (Set Scaling 1:1): Perintah ini digunakan untuk mengatur scaling pada mouse menjadi 1:1.

Data hasil pembacaan *mouse* yang dikirimkan melalui UART adalah sama dengan data yang dibaca dari *mouse* melalui protokol PS/2, dengan menghilangkan *start bit*, *parity bit*, dan *stop bit*, sehingga hanya tertinggal 8 bit data yang merepresentasikan kondisi tombol *mouse* dan pergerakan *mouse*.

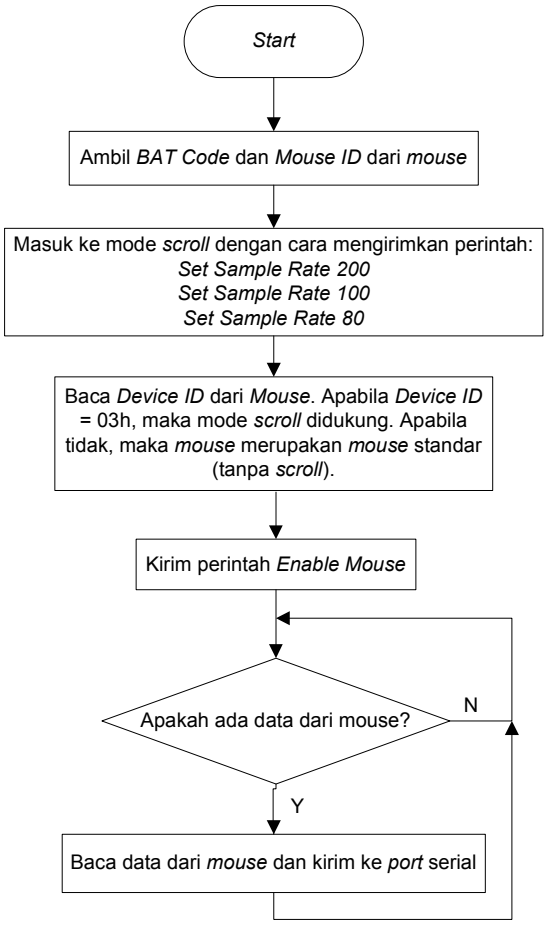
Adapun format frame dari data yang dikirimkan melalui UART tersebut adalah sebagai berikut:
Misal: Data hasil pembacaan *mouse*: 0010 1001

Data hasil pembacaan mouse		Data yang dikirimkan melalui UART	
Bit ke-0	0	Byte ke-0	00h
Bit ke-1	0	Byte ke-1	00h
Bit ke-2	1	Byte ke-2	01h
Bit ke-3	0	Byte ke-3	00h
Bit ke-4	1	Byte ke-4	01h
Bit ke-5	0	Byte ke-5	00h
Bit ke-6	0	Byte ke-6	00h
Bit ke-7	1	Byte ke-7	01h

Tabel 4
***Frame* Pengiriman Data Hasil Pembacaan Mouse ke UART**

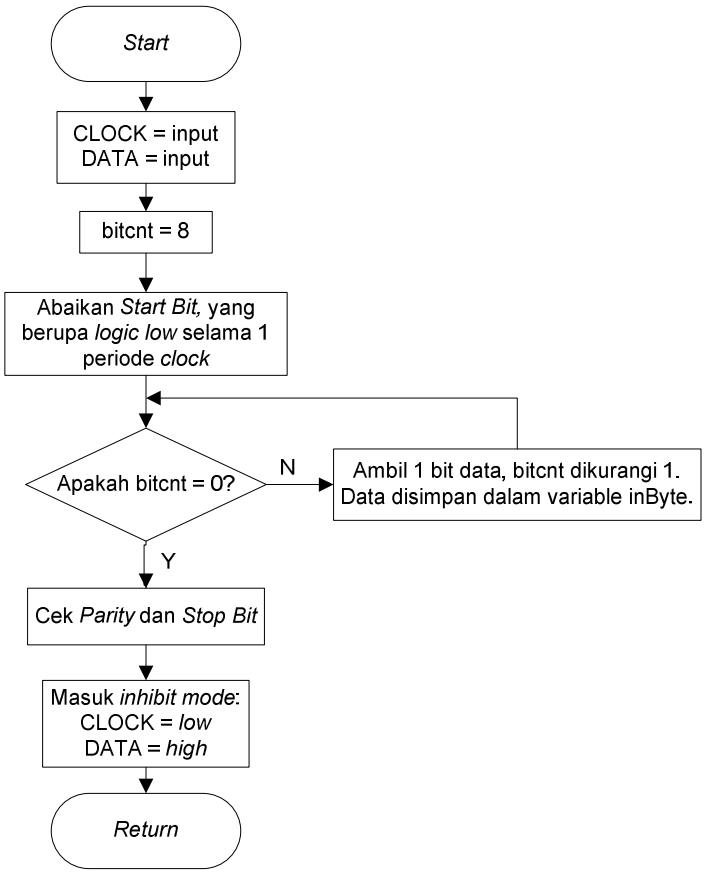
Sehingga untuk 8 bit data hasil pembacaan dari *mouse*, akan ada 8 byte data yang dikirimkan secara berurutan melalui UART.

F*lowchart* dari program utama PS2_asm.asm adalah sebagai berikut:

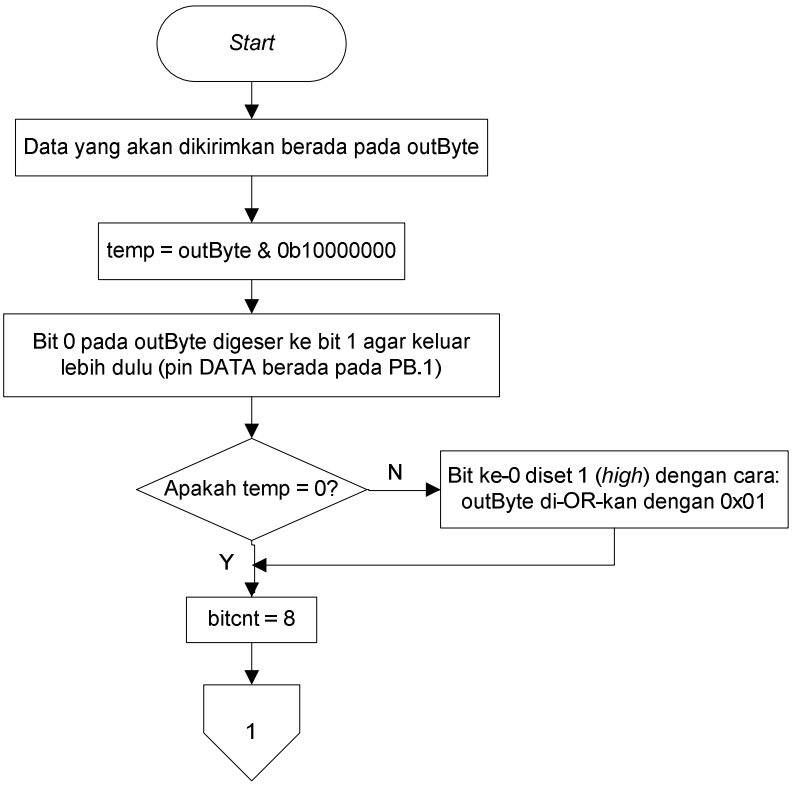


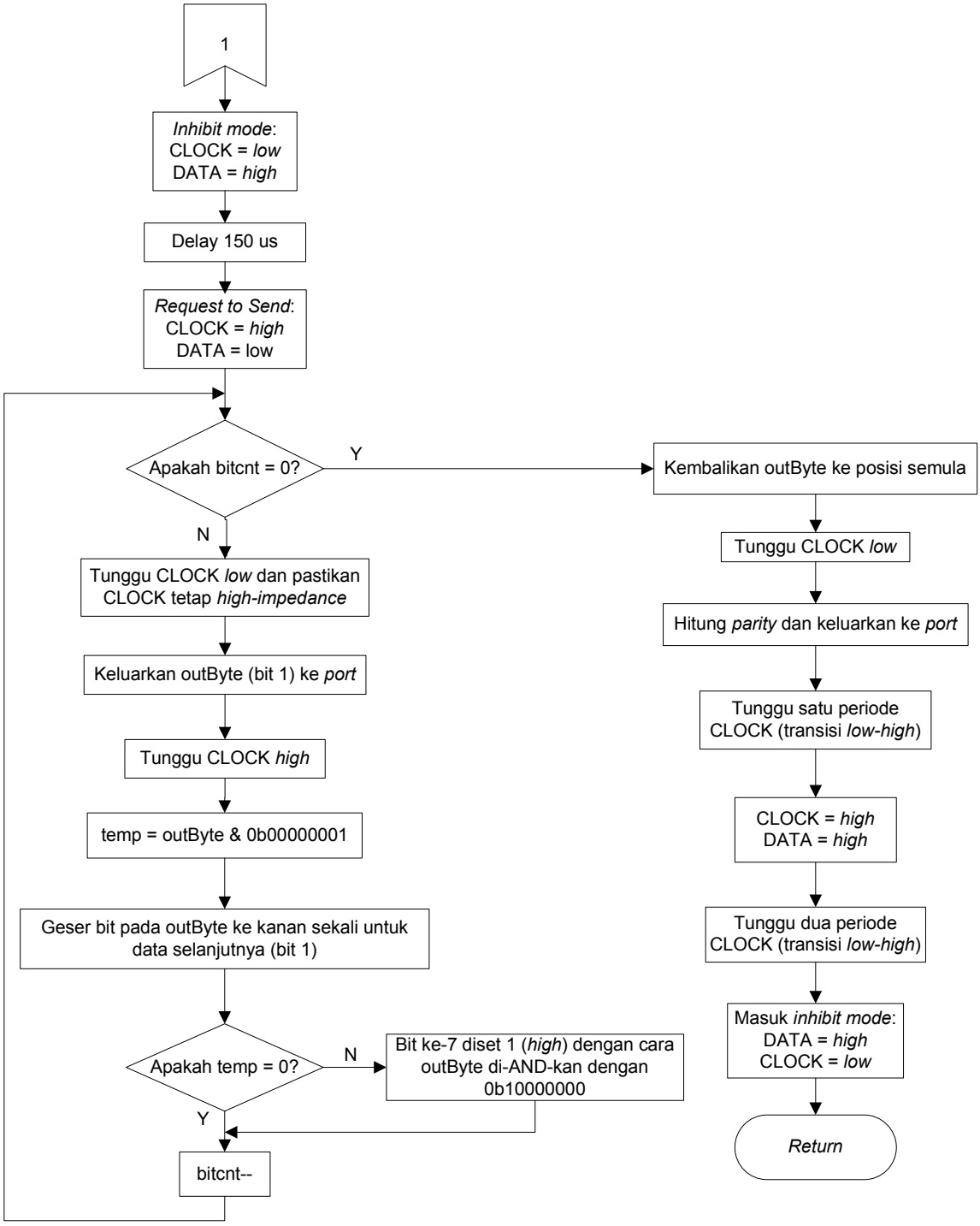
Gambar 4
***Flowchart* Program Utama PS2_asm.asm**

F*lowchart* dari rutin pembacaan dan penulisan data ke *mouse* adalah sebagai berikut:



Gambar 5
***Flowchart* subrutin geByte PS2_asm.asm**





Gambar 6
Flowchart subrutin putByte PS2_asm.asm

Cara kerja program secara garis besar adalah sebagai berikut:

Program utama:

1. Pertama-tama, program akan mengambil dua byte data dari *mouse*, yaitu *BAT Code* dan *Mouse ID*. *BAT Code* adalah suatu kode dari *mouse* yang menyatakan bahwa *hardware mouse* dalam kondisi baik (AAh) atau rusak (FCh). *Mouse ID* adalah suatu kode identitas *mouse* yang menyatakan kapabilitas *mouse* (Standard mouse = 00h).
2. Setelah itu, program akan mencoba untuk masuk ke mode *scroll* dengan mengirimkan perintah *Set Sample Rate* diikuti dengan nilai 200, 100, dan 80 secara berurutan sebanyak tiga kali. Apabila *mouse* mendukung mode *scroll*, maka pada waktu program membaca *Device ID*, *mouse* akan membalas dengan 03h. Apabila *mouse* tidak membalas dengan 03h, berarti *mouse* tersebut merupakan *mouse* standar (tanpa *scroll*).

3. Kirim perintah *Enable Mouse* agar *mouse* melaporkan data setiap ada perubahan kursor *mouse* maupun penekanan tombol. Saat pertama kali dinyalakan, secara *default mouse* berada pada *Stream Mode*, yaitu *mouse* akan mengirim data setiap kali ada perubahan posisi kursor maupun penekanan tombol, tetapi masih perlu dilengkapi dengan perintah *Enable Mouse* agar data dapat keluar ke *port* PS/2.
4. Program akan menunggu datangnya data dari *mouse*. Saat ada data masuk, data akan disimpan di variabel inByte dan selanjutnya dikirimkan melalui *port* serial. Bagian ini dijalankan berulang-ulang (*looping*) agar data dari *mouse* dapat dibaca dan dikirimkan ke *port* serial secara terus-menerus.

Prosedur pembacaan data dari *mouse* (*getByte*):

1. Pertama-tama, pin CLOCK dan DATA diatur sebagai *input* dan variabel bitcnt yang berfungsi sebagai *counter* data diisi dengan 8.
2. Setelah itu, program akan mengabaikan *start bit*, yaitu perubahan *logic* pertama yang berupa *logic low* selama satu periode *clock*.
3. Program akan mengecek nilai bitcnt, apakah sudah nol. Apabila belum nol, maka program akan menerima data sebanyak 1 bit dari *mouse* dan variabel bitcnt dikurangi 1.
4. Langkah no. 3 diulangi kembali sampai bitcnt bernilai nol. Saat kondisi ini tercapai berarti data sudah diterima secara keseluruhan. Data yang diterima dimasukkan dalam variabel inByte dengan cara meng-OR-kan inByte dengan bitmask untuk mendapat satu bit data (sebagai contoh, bila bitmask=0b00000001, maka didapat bit ke-0, dst.). Bitmask ini digeser ke kiri satu kali setiap *looping*, sehingga pada akhirnya kedelapan bit data pada variabel inByte terisi.
5. Selanjutnya, program akan mengambil *parity bit*. Data *parity bit* ini akan disimpan dalam variabel parity. Dalam program ini, pemeriksaan terhadap kebenaran *parity bit* belum diimplementasikan, sehingga program hanya menerima *parity bit* saja tanpa memeriksa kebenarannya. Setelah itu, program juga akan memeriksa *stop bit* yang merupakan akhir dari komunikasi *serial*.
6. Terakhir, program akan masuk ke *Inhibit Mode* dengan mengatur *port* DATA menjadi *high* dan CLOCK menjadi *low*, agar *mouse* tidak mengirimkan data sebelum mikrokontroler siap kembali. Apabila program telah siap untuk menerima data kembali dari *mouse*, maka pin CLOCK harus diatur menjadi HIGH, yang menandakan jalur data saat itu sedang *idle*.

Prosedur pengiriman perintah ke *mouse* (*putByte*):

1. Pertama-tama, data yang akan dikirim diisikan ke variabel outByte. Lalu outByte di-AND-kan dengan bilangan biner 1000 0000 dan dimasukkan ke variabel temp. Tujuannya adalah untuk mengambil bit ke-7 dari outByte.
2. Setelah itu, bit-bit pada outByte digeser ke kiri sekali agar bit pertama (bit 0) berpindah ke bit 1. Hal ini dilakukan karena pin DATA berada pada bit 1 dari *port* mikrokontroler.
3. Lalu program akan mengecek apakah variabel temp bernilai nol. Apabila tidak, maka outByte di-OR-kan dengan bilangan 0x01, dengan maksud mengatur bit ke-0 menjadi bernilai 1 (hal ini sama dengan prinsip *rotate bit left*). Setelah selesai, program akan diteruskan ke perintah berikutnya, yaitu variabel bitcnt, yang merupakan *counter* data, diisi dengan 8.
4. Pada tahap ini, program akan memulai pengiriman data ke *mouse PS/2*. Pertama-tama, program masuk ke *Inhibit Mode* dengan cara mengeluarkan logika *low* ke pin CLOCK dan *high* ke pin DATA. Setelah itu, program akan menunggu selama 150 µs. Kemudian program masuk ke mode *Request To Send* dengan mengeluarkan logika *high* ke pin CLOCK dan *low* ke pin DATA.
5. Lalu program akan memeriksa, apakah variabel bitcnt bernilai nol. Apabila tidak, maka program akan mengirimkan bit pertama (yang terletak pada bit 1 dari variabel outByte) dengan cara menunggu pin CLOCK berlogika *low*, kemudian mengeluarkan isi variabel outByte (bit ke-1) ke *port*, dan memastikan pin CLOCK tetap berada pada kondisi *high impedance*. Setelah itu, program akan menunggu pin CLOCK berlogika *high* kembali.
6. Setelah itu, outByte akan di-AND-kan dengan bilangan biner 0000 0001 dan hasilnya disimpan di variabel temp. Hal ini dilakukan untuk mendapatkan nilai bit ke-0 dari data yang akan dikirim. Kemudian, isi variabel outByte akan digeser ke kanan satu kali untuk mempersiapkan data selanjutnya. Lalu program akan mengecek apakah variabel temp bernilai nol. Jika tidak, maka outByte akan di-OR-kan dengan bilangan biner 1000 0000 dengan maksud untuk mengatur nilai bit ke-7 dari outByte menjadi berlogika 1 (hal ini sama dengan prinsip *rotate bit right*). Terakhir, variabel bitcnt akan dikurangi 1.
7. Program akan kembali ke poin 5 sampai variabel bitcnt bernilai nol.
8. Setelah pengiriman data selesai, yang ditandai dengan variabel bitcnt bernilai 0, variabel outByte dikembalikan ke posisi semula (sebelum digeser) untuk perhitungan *parity*. Program akan mengirim perhitungan *parity* dengan cara menunggu pin CLOCK berlogika *low*, mengirimkan *parity*, dan menunggu pin CLOCK berlogika *high* kembali.
9. Pada tahap terakhir, program akan mengirimkan *stop bit* dengan cara menunggu pin CLOCK berlogika *low*, mengatur pin CLOCK dan DATA menjadi *high impedance*, dan menunggu CLOCK berlogika *high* kembali. Lalu program akan menunggu *Acknowledgement* (ACK) dari *mouse* dengan menunggu pin CLOCK berlogika *low* kemudian *high* satu kali lagi.
7. Terakhir, program akan masuk ke *Inhibit Mode* dengan mengatur *port* DATA menjadi *high* dan CLOCK menjadi *low*, agar *mouse* tidak mengirimkan data sebelum mikrokontroler siap kembali. Apabila program

telah siap untuk menerima data kembali dari *mouse*, maka pin CLOCK harus diatur menjadi HIGH, yang menandakan jalur data saat itu sedang *idle*.

Keseluruhan program tersebut diimplementasikan dalam mikrokontroler AT90S2313 yang terdapat pada board DT-AVR Low Cost Nano System. Tetapi, menurut informasi yang terdapat pada *website* Atmel, AT90S2313 tidak direkomendasikan lagi untuk dipakai pada sistem baru. Sebagai gantinya, Atmel merekomendasikan pengguna untuk memakai ATtiny2313.

Agar program ini dapat berjalan pada mikrokontroler ATtiny2313, maka perlu dilakukan sedikit perubahan pada program, yaitu:

.include “2313def.inc” diganti menjadi .include “tn2313def.inc”

Perintah lain yang terdapat dalam program tidak perlu diganti, karena dalam *file* “tn2313def.inc” (untuk ATtiny2313) sudah terdapat definisi untuk kompatibilitas dengan AT90S2313. *AVR Assembler 2* yang terdapat dalam *AVR Studio 4* secara otomatis akan menyesuaikan nama *register* pada AT90S2313 menjadi nama *register* pada ATtiny2313.

L*isting* program PS2_asm.asm terdapat pada AN153.ZIP

Selamat berinovasi!