BAB III

METODE PENELITIAN

Metode yang digunakan pada perancangan dan pembuatan perangkat keras dan perangkat lunak yaitu dengan studi kepustakaan dan eksperimen. Dengan cara ini penulis berusaha untuk mendapatkan dan mengumpulkan data-data, informasi, konsep-konsep yang bersifat teoritis dengan membaca buku-buku serta literature dan bahan-bahan kuliah yang berkaitan dengan permasalahan tersebut.

Setelah dilakukan perancangan perangkat keras yang bisa dikatakan sebagai perancangan sederhana, maka dilakukan eksperimen mengenai perangkat lunak yang ingin dirancang. Dengan melakukan berbagai percobaan dan algoritma untuk mendapatkan perangkat lunak yang lebih baik.

3.1 Perangkat Keras

Pada perancangan perangkat keras dan pembuatan perangkat keras ini, langkah pertama yang harus dilakukan adalah menentukan tujuan dari pembuatan perangkat keras. Kemudian merancang blok diagram secara umum atau skematik rangkaian.

Tujuan dari pembuatan perangkat keras pada penelitian kali ini terbagi menjadi dua. Pertama, membuat sebuah rangkaian yang menghubungkan atau mengkoneksikan dua buah PLC agar bisa saling berkomunikasi pada masing-

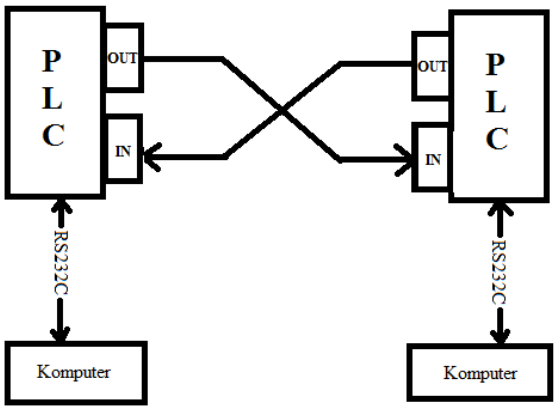
masing PLC menggunakan 1 pin *output* dan 1 pin *input*. Pembuatan perangkat keras yang pertama dimaksudkan untuk melakukan eksperimen terlebih dahulu pada perangkat lunak yang dibangun menggunakan aplikasi atau sistem yang sederhana sebelum nantinya diimplementasikan di sistem yang lebih besar yaitu MPS. Pada perancangan perangkat keras yang pertama penulis beri nama dengan "Perangkat Keras Komunikasi PLC Pada *Simple System*".

Sedangkan perancangan perangkat keras yang kedua menghubungkan dan memastikan hubungan antar PLC pada MPS agar bisa saling terhubung untuk nantinya dapat melakukan komunikasi antar PLC pada MPS. Sama halnya dengan perancangan perangkat keras yang pertama, perancangan perangkat keras yang kedua juga memerlukan masing-masing PLC menggunakan 1 pin *input* dan 1 pin *output* untuk berkomunikasi dengan sebuah PLC yang lain pada MPS *previous station* atau *subsequent station*. Maksud dari perancangan perangkat keras yang kedua adalah ingin mengamati dan mempelajari bagaimana perangkat lunak yang sudah dibangun sebelumnya di *simple system* bekerja pada sistem yang lebih besar, yaitu MPS. Dan seperti perancangan perangkat keras pertama yang diberi nama, maka perancangan perangkat keras yang kedua penulis juga memberi nama yaitu "Perangkat Keras Komunikasi Pada MPS".

3.1.1 Perangkat Keras Komunikasi Pada *Simple System*

Seperti yang dikatakan di atas, apabila tujuan dari pembuatan perangkat keras sudah ditentukan maka langkah selanjutnya adalah merancang blok diagram atau rangkaian skematik dari perangkat keras tersebut. Karena tujuan dari perancangan perangkat keras yang pertama sudah dijelaskan

sebelumnya maka pada bagian ini penulis tidak menjelaskan lagi, oleh karena itu langsung saja pada rancangan blok diagram yang dapat dilihat pada gambar 3.1.



Gambar 3.1 Komunikasi Pada *Simple System*

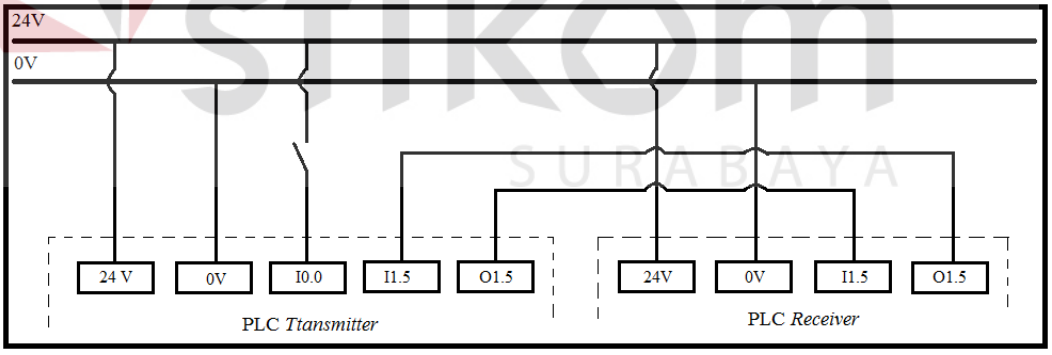
Pada gambar 3.1 dapat dilihat bahwa pada perancangan perangkat keras pertama ini membutuhkan 2 buah kabel komunikasi serial dan 2 buah PLC. PLC yang di sebelah kiri ditentukan sebagai PLC pengirim atau *transmitter*, sedangkan pada PLC yang di sebelah kanan ditentukan sebagai PLC penerima *receiver*. Di antara keduanya dapat dilihat bahwa PLC dihubungkan oleh 1 pin *output transmitter* ke 1 pin *input receiver*, dan 1 pin *output receiver* dihubungkan ke 1 pin *input transmitter*. Untuk pin yang digunakan pada masing-masing PLC adalah pin *ouput* 1.5 (O1.5) dan *input* 1.5 (I1.5). Agar lebih jelas mengenai *allocation list* PLC yang digunakan dapat dilihat pada tabel tabel 3.1 di bawah ini.

Tabel 3.1 Allocation List Simple System

PLC *Transmitter*		**PLC *Receiver***	
Operand	**Fungsi**	***Operand***	**Fungsi**
O1.5	Komunikasi *output* ke *receiver*	O1.5	Komunikasi *output* ke *transmitter*
I1.5	Komunikasi *input* dari *receiver*	I1.5	Komunikasi *input* dari *transmitter*
I0.0	Tombol *start* pengiriman data	FW14	Menyimpan jumlah bit yang

			akan diterima
FW14	Menyimpan jumlah bit yang akan dikirim	FW15	Menyimpan data yang akan diterima
FW15	Menyimpan data yang akan dikirimkan	F14.15	*Flag* tanda penerimaan selesai
F14.15	*Flag* tanda pengiriman selesai	CP15	*Counter Preselect*
CP15	*Counter Preselect*	C15	*Counter Bit Status*
C15	*Counter Bit Status*	T31	*Timer Bit Status*
T31	*Timer Bit Status*	F14.0	*Bit Flag* yang diterima
F14.0	*Bit Flag* yang dikirim	T30	*Timer Bit Status*
		CW15	*Counter Word*
		CP14	*Counter Preselect*
		C14	*Counter Bit Status*

Dalam sebuah perancangan *system* di PLC tidak pernah lepas dari yang namanya diagram rangkaian listrik, hal ini dilakukan bertujuan untuk memudahkan dalam perancangan sistem yang nantinya akan dibuat. Diagram rangkaian listrik merupakan hubungan antara catu daya, PLC, *input* dan *ouput*. Diagram ini sangat berguna sebagai panduan pemasangan peralatan seperti yang terlihat pada gambar 3.2 di bawah ini.



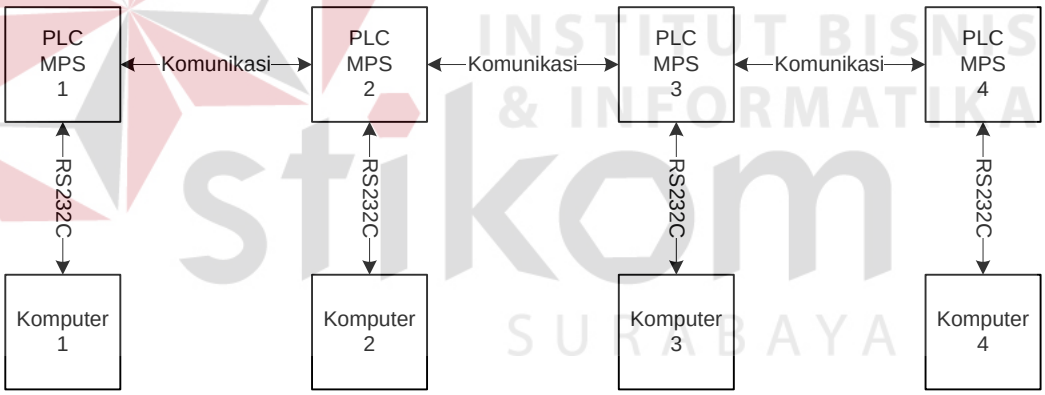
Gambar 3.2 Diagram Rangkaian Listrik *simple system*

Perangkat keras yang dibutuhkan pada perancangan pertama ini adalah 2 buah kabel komunikasi serial, 2 buah PLC yang mana keduanya sudah berada dalam satu jalur sumber tegangan 24V, 2 buah modul *input* dan *output* yang sudah tersedia di laboratorium PLC STIKOM dan dua buah modul tombol yang juga

sudah tersedia di laboratorium STIKOM. Serta beberapa kabel untuk menghubungkan *input* dan *output* dari masing-masing PLC, juga diperlukan kabel untuk menghubungkan antara PLC *transmitter* dan PLC *receiver*. Kabel komunikasi ini dipasang di *output* 1.5 PLC *transmitter* menuju *input* 1.5 PLC *receiver*, begitu juga dipasang pada *output* 1.5 PLC *receiver* menuju *input* 1.5 PLC *transmitter*.

3.1.2 Perangkat Keras Komunikasi Pada MPS

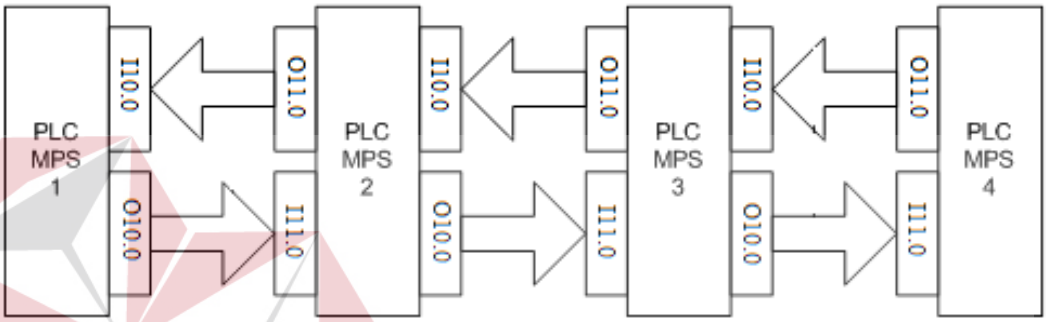
Pada perancangan perangkat keras ini diperlukan kabel komunikasi serial RS232C dan kabel komunikasi antar PLC. Gambar 3.3 menunjukkan diagram perancangan perangkat keras dari penelitian ini.



Gambar 3.3 Perancangan Perangkat Keras Pada MPS

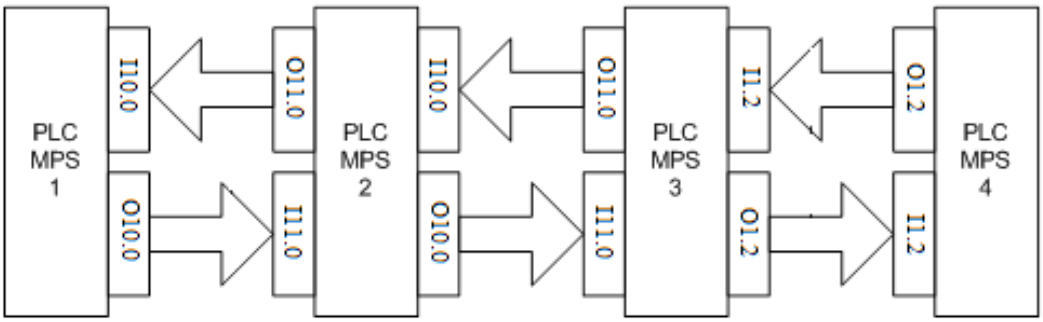
Kabel komunikasi serial RS232C dibutuhkan untuk media komunikasi antara PLC dengan komputer. Banyaknya PLC yang digunakan dalam penelitian ini sebanyak 4 unit. Agar pemantauan tiap PLC dapat fokus, diperlukan 4 unit komputer. Jadi banyaknya kabel komunikasi serial RS232C yang diperlukan sebanyak 4 unit.

Penulis berniat menggunakan jalur komunikasi MPS yang sudah ada untuk melakukan komunikasi, sehingga tidak perlu lagi memasang kabel tambahan untuk jalur komunikasi antar PLC pada MPS. Jalur komunikasi yang sudah ada pada MPS adalah jalur komunikasi paralel yaitu dari 10.0-10.7 dan 11.0-11.7, karena jalur yang dibutuhkan hanya satu maka hanya satu *output* dan satu *input* yang digunakan, seperti pada gambar 3.4 di bawah ini.



Gambar 3.4 Komunikasi MPS

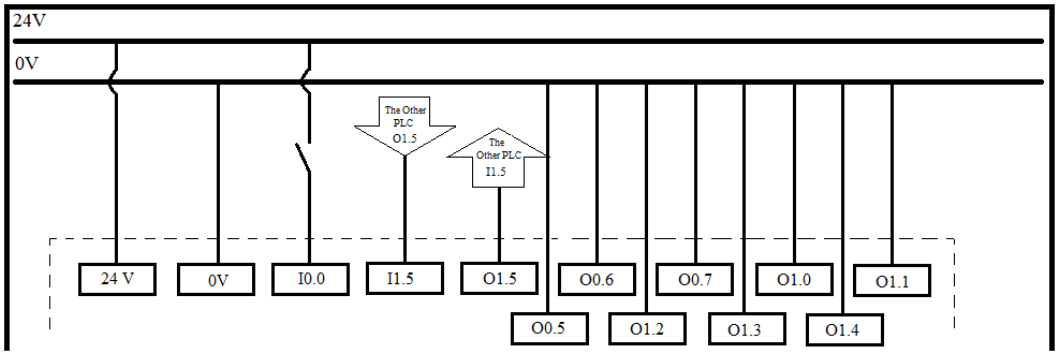
Setelah dilakukan pengecekan terhadap jalur-jalur komunikasi MPS, ternyata didapatkan bahwa jalur komunikasi antara MPS3 dan MPS4 tidak bisa digunakan lagi. Oleh karena itu dilakukan perubahan dengan menambah kabel baru dan dipasang pada pin I1.2 dan O1.2 masing-masing MPS3 dan MPS4. Jalur komunikasi dipasang pada pin tersebut karena kemudahan pemasangan kabel pada pin tersebut dan pin tersebut tidak digunakan untuk keperluan sistem MPS3 dan MPS4 secara umum. Sehingga diagram blok rangkaian menjadi seperti di bawah ini.



Gambar 3.5 Komunikasi MPS

3.1.3 Komunikasi Non-Pneumatic

Terdapat keraguan pada percobaan mengamati kecepatan kinerja dari MPS setelah dilakukan perubahan komunikasi menjadi serial. Keraguan tersebut muncul karena rangkaian *pneumatic* yang membutuhkan tekanan angin akan mengalami perubahan tekanan angin pada saat dilakukan percobaan. Hal ini akan mengakibatkan kinerja MPS pada percobaan satu dengan lainnya tidak akan sama persis. Sehingga kita tidak dapat melihat perbedaan komunikasi serial dan paralel secara valid. Untuk solusi dari masalah tersebut maka dilakukan percobaan *non-penumatic* (hanya menggunakan listrik sebagai aktuatornya) agar dapat dilihat perbedaan komunikasi paralel dan serial tanpa adanya perbedaan kinerja pada percobaan satu dengan lainnya. Pada percobaan *non-pneumatic* diberikan program dengan mengaktifkan seluruh modul yang ada pada PLC yang berjalan secara *multitasking* dengan alasan agar beban kerja dari PLC mendekati dengan beban kerja PLC pada MPS. Berikut adalah diagram rangkaian listrik komunikasi *non-pneumatic*.

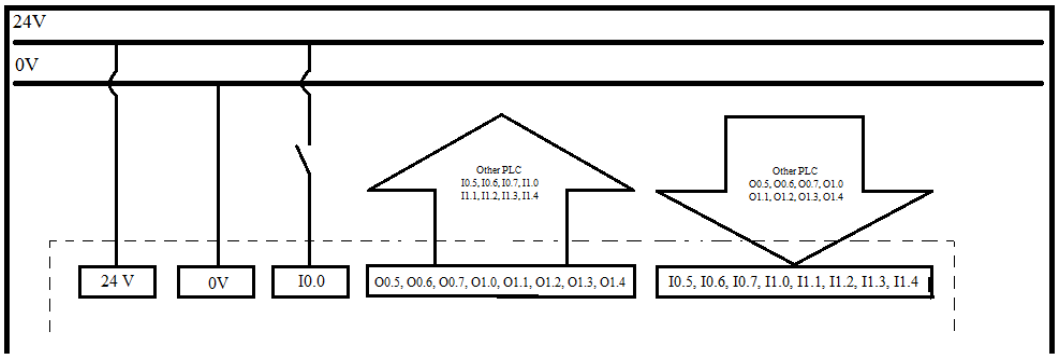


Gambar 3.6 Diagram Listrik *Non-Pneumatic* Serial

Pada gambar di atas terlihat bahwa I1.5 dan O1.5 terhubung dengan I/O dari PLC lainnya, PLC tersebut adalah PLC yang diajak berkomunikasi. Diagram listrik pada gambar di atas diterapkan pada PLC pengirim dan penerima dengan catatan masing-masing sumber tegangan harus saling terhubung juga. Yaitu 24V pengirim terhubung dengan 24V penerima, begitu pula dengan 0V.

Untuk I0.0 digunakan untuk tombol start, O0.5 sampai O1.0 digunakan sebagai output yang menandakan karakteristik sebuah benda. O1.1 digunakan sebagai penanda bahwa modul ke-1 aktif, begitu pula untuk O1.2 penanda modul ke-2 aktif, O1.3 penanda modul ke-3 aktif dan O1.4 penanda modul ke-4 aktif. untuk modul ke-5 dan modul ke-6 tidak menggunakan penanda.

Rangkaian *non-pneumatic* terdiri dari dua, yaitu rangkaian paralel dan rangkaian serial. Tujuan pembuatan dua buah rangkaian ini adalah agar nantinya dapat dilihat perbedaan antara komunikasi *non-pneumatic* paralel dan serial. Gambar 3.6 adalah rangkaian dari komunikasi *non-pneumatic* serial, sedangkan rangkaian komunikasi *non-pneumatic* paralel adalah seperti gambar berikut.



Gambar 3.7 Diagram Listrik *Non-Pneumatic* Paralel

Pada gambar 3.7 dapat dilihat bahwa untuk berkomunikasi paralel membutuhkan 8 *pin input* dan 8 *pin output*. Pada masing-masing PLC, pin O0.5,O0.6, O0.7, O1.0, O1.1, O1.2, O1.3, O1.4 dihubungkan dengan pin PLC lainnya yaitu I0.5, I0.6, I0.7, I1.0, I1.1, I1.2, I1.3,I1.4. dengan catatan 24V dan 0V dari masing-masing PLC yang berkomunikasi juga saling terhubung.

24V penerima harus terhubung dengan 24V pengirim, begitu pula dengan 0V penerima terhubung dengan 0V pengirim. Hal ini harus dierhatikan, sebab apbila sumber tegangan ini tidak saling terhubung maka komunikasi tidak akan dapat berjalan. Hal ini dikarenakan dalam rangkaian listrik semua komponen harus berada dalam satu naungan sumber tegangan.

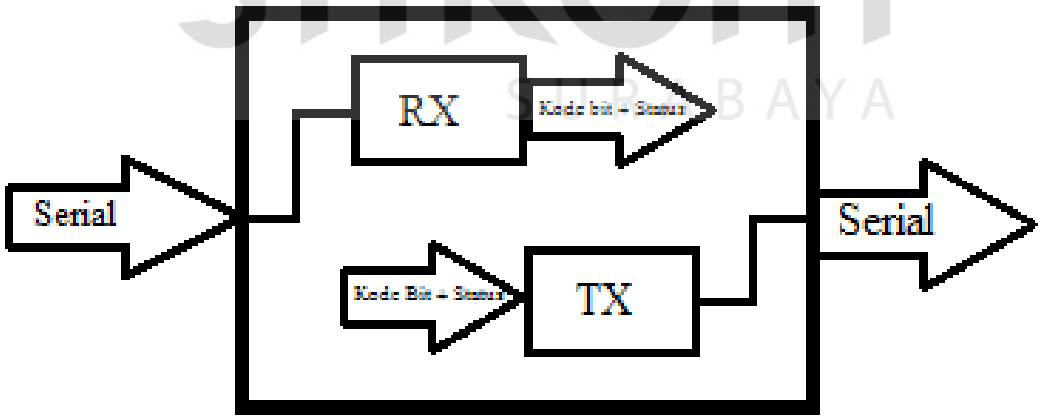
3.2 Perancangan Perangkat Lunak

Untuk perancangan perangkat lunak juga terdapat dua bagian, yaitu perangkat lunak komunikasi *simple system* dan perangkat lunak komunikasi MPS. Data-data tersebut dikodekan sehingga bisa dibaca sebagai sebuah informasi yang dapat memuat bit yang digunakan dan status dari bit tersebut. Untuk lebih jelasnya dapat dilihat pada tabel 3.2 mengenai pengkodean informasi.

Tabel 3.2 Pengkodean Sinyal Komunikasi

Paralel		Serial			Keterangan
Status	Bit	Status	Bit	Deretan Bit	
0	0	0	000	0000	Bit ke-0 berstatus 0
1	0	1	000	1000	Bit ke-0 berstatus 1
0	1	0	001	0001	Bit ke-1 berstatus 0
1	1	1	001	1001	Bit ke-1 berstatus 1
0	2	0	010	0010	Bit ke-2 berstatus 0
1	2	1	010	1010	Bit ke-2 berstatus 1
0	3	0	011	0011	Bit ke-3 berstatus 0
1	3	1	011	1011	Bit ke-3 berstatus 1
0	4	0	100	0100	Bit ke-4 berstatus 0
1	4	1	100	1100	Bit ke-4 berstatus 1
0	5	0	101	0101	Bit ke-5 berstatus 0
1	5	1	101	1101	Bit ke-5 berstatus 1
0	6	0	110	0110	Bit ke-6 berstatus 0
1	6	1	110	1110	Bit ke-6 berstatus 1
0	7	0	111	0111	Bit ke-7 berstatus 0
1	7	1	111	1111	Bit ke-7 berstatus 1

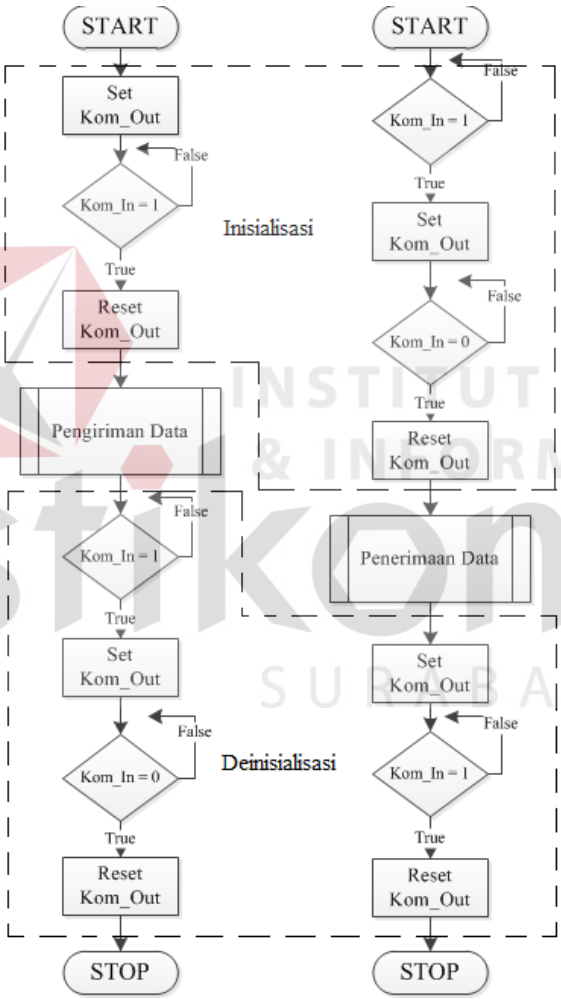
Sedangkan desain protokol komunikasi yang dibuat berupa sebuah modul yang dapat memilih mode pengiriman atau penerimaan data. Lebih jelasnya dapat dilihat pada gambar di bawah ini.



Gambar 3.8 Desain Protokol Komunikasi

Penjelasan dari blok diagram di atas yaitu pada saat modul dijalankan, modul otomatis dapat menentukan mode yang akan dijalankan. Mode terbagi

menjadi dua, mode pengiriman (TX) dan mode penerimaan (RX). Apabila mode yang dipilih adalah RX maka modul otomatis akan membaca data serial yang masuk dan menjadikan *bit-bit* yang masuk menjadi informasi yang menunjukkan kode *bit* dan status *bit* tersebut. Dan apabila mode TX yang dipilih, maka modul tomatis akan menjadikan informasi berupa kode *bit* dan status *bit* menjadi *bit-bit* yang bisa dikirimkan secara serial.

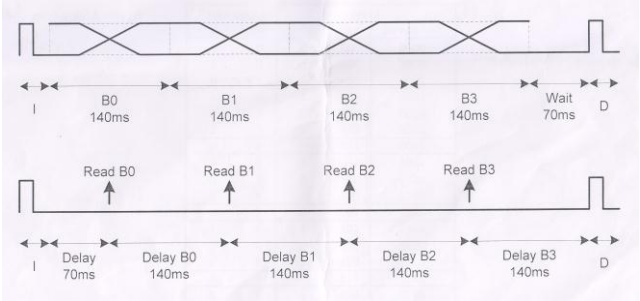


Gambar 3.9 *Flowchart* inisialisasi dan deinisialisasi TX dan RX

Pada *flowchart* TX dan RX terlihat Kom_Out dan Kom_In, maksud *variable* tersebut adalah Kom_Out berfungsi sebagai *port output* PLC *transmitter*

yang dihubungkan ke *input* PLC *receiver*. Sedangkan Kom_In berfungsi sebagai *port input* PLC *receiver* yang dihubungkan ke *output* PLC *transmitter*. Penjelasan dari *flowchart* TX dan RX dimulai dari insialisasi keduanya. Yaitu pada saat TX memberi nilai 1 pada Kom_Out yang berarti nilai Kom_In pada RX juga akan bernilai 1, maka RX dapat melanjutkan ke langkah program berikutnya yaitu memberi nilai 1 pada Kom_Out dari RX. Secara otomatis nilai Kom_In dari TX akan menjadi 1 sehingga TX bisa melanjutkan ke langkah program berikutnya yaitu memberi nilai 0 pada Kom_Out. Dan secara otomatis nilai Kom_In pada RX juga akan berubah menjadi 0 yang berarti RX bisa melanjutkan ke langkah program selanjutnya yaitu memberi nilai 0 pada Kom_Out dari RX. Kemudian RX dan TX melanjutkan ke proses pengiriman data, dan proses inisialisasi TX dan RX pun selesai. Sedangkan untuk proses deinisialisasi keduanya, merupakan kebalikan proses dari insialisasi. Yang artinya proses inisialisasi dari TX menjadi proses deinisialisasi dari RX, dan proses inisialisasi dari RX menjadi proses deinisialisasi dari TX.

Dari desain protokol ini juga ditentukan berapa lama *timer* PLC yang digunakan untuk proses mengirim atau menerima 1 *bit* data. Dalam perancangan perangkat lunak atau modul komunikasi ini penulis mendesain waktu yang dibutuhkan untuk mengirim 1 *bit* data adalah 140ms. Seperti yang ada pada gambar di bawah ini.



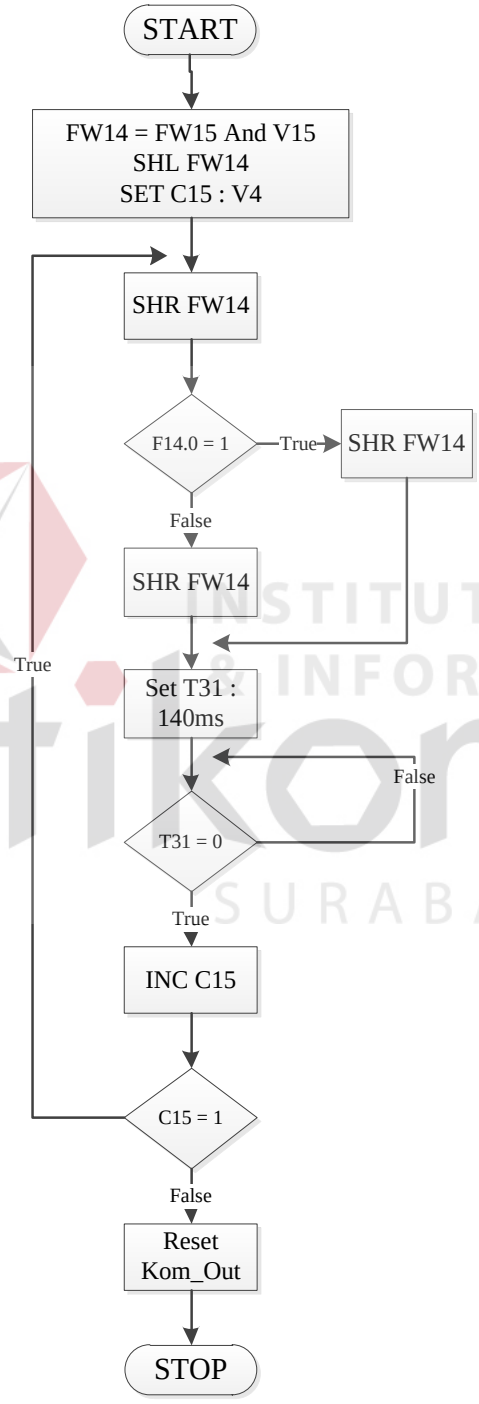
Gambar 3.10 *Timing* Diagram Komunikasi

Penggunaan 140ms/bit dikarenakan berdasar pada penelitian sebelumnya yang menggunakan 140ms/bit untuk sistem yang lebih kompleks. *System* yang lebih kompleks menuntut agar CCU bekerja lebih keras, sehingga ditakutkan akan mengganggu *timer* PLC yang juga dikerjakan oleh CCU. Dalam komunikasi serial waktu pengiriman dan pembacaan data harus tepat sehingga data dapat dibaca oleh penerima dengan benar. Sehingga apabila *timer* dari salah satu PLC yang berkomunikasi terganggu, maka otomatis juga akan mengganggu waktu pengiriman dan pembacaan data.

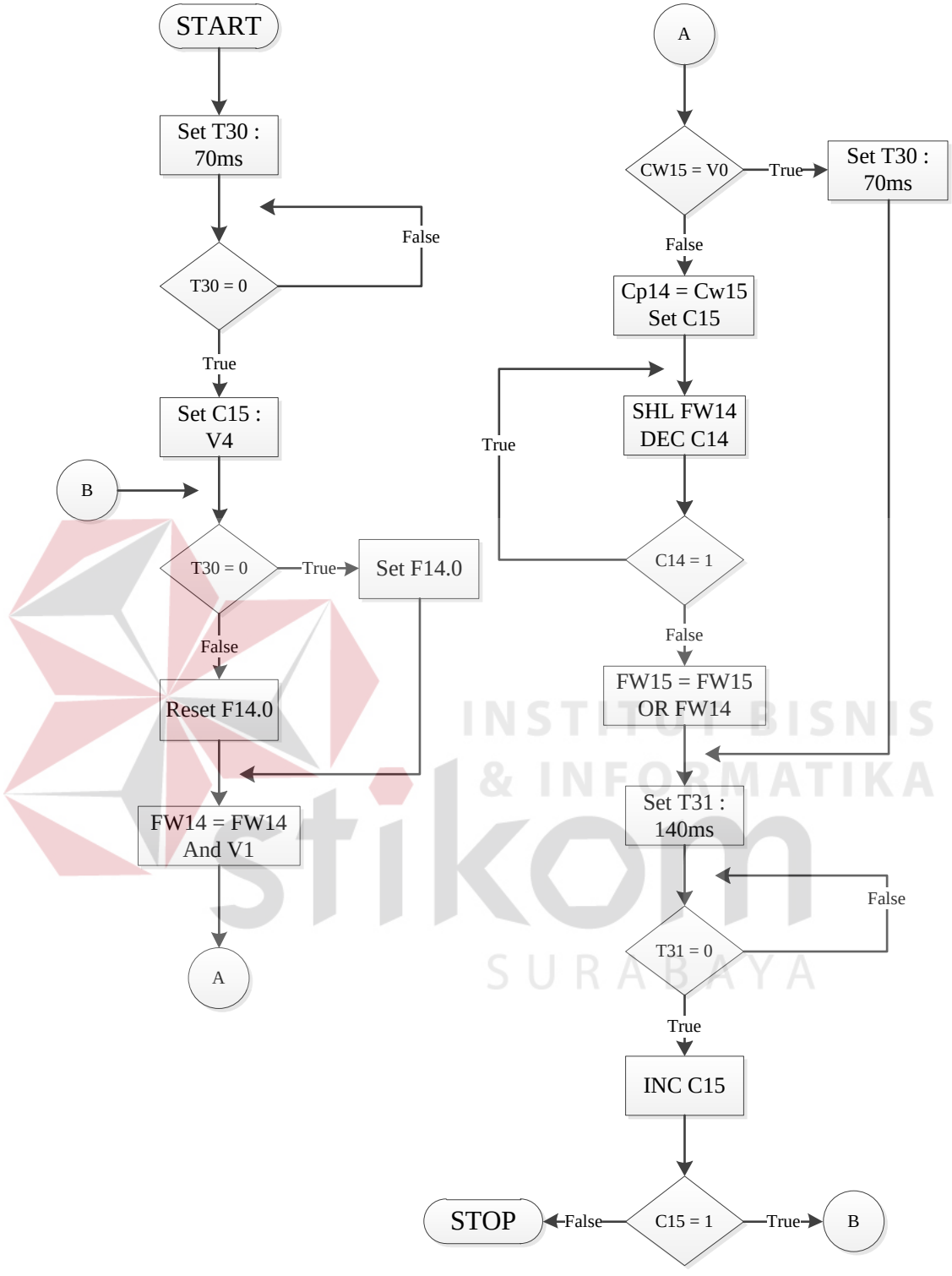
Misalnya saja *timer* dari PLC pengirim terganggu akibat CCU harus melaksanakan proses yang prioritasnya lebih tinggi daripada *timer* sehingga *timer* mengalami penundaan dalam pelaksanaan prosesnya selama beberapa saat. Hal ini akan berakibat fatal apabila waktu yang disediakan untuk pengiriman terlalu sedikit.

Oleh karena itu *timer* direntangkan hingga 140ms/bit untuk mengantisipasi adanya gangguan dari kerja CCU yang terlalu berat dan berdampak terhadap *timer* sehingga mengakibatkan kesalahan dalam proses komunikasi. Sedangkan untuk nilai 140ms didapatkan dari perkiraan saja atau dengan kata lain tanpa melalui

perhitungan-perhitungan khusus. Gambar 3.10 adalah *flowchart* pengiriman data yang cuma bertugas mengirimkan 1 *bit* data setiap 140ms sekali. Kegiatan tersebut diulang sebanyak 4 kali.



Gambar 3.11 *Flowchart* Pengiriman Data



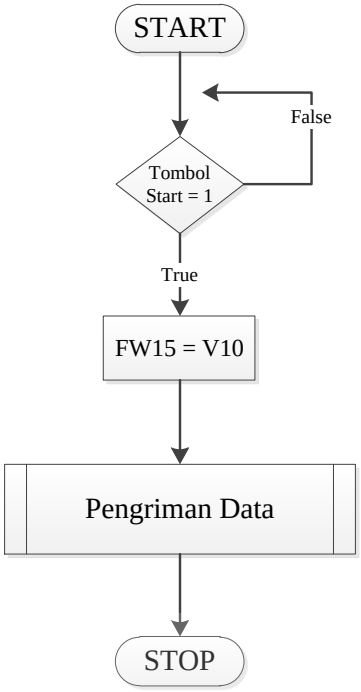
Gambar 3.12 *Flowchart* Penerimaan Data

Untuk mengetahui bahwa data yang diterima oleh penerima adalah data yang benar dapat dilakukan pengecekan manual pada *flag* tempat penyimpanan data. Karena *programmer* pasti sudah mengetahui apa data yang seharusnya diterima maka modul ini tidak dilengkapi dengan pengecekan terhadap data yang diterima.

Modul ini tidak dilengkapi dengan *error control* dengan alasan agar waktu yang digunakan untuk berkomunikasi tidak terlalu lama. Hal ini disebabkan apabila dilakukan pengecekan otomatis, maka modul harus mengirimkan lagi data yang diterima kepada pengirim, kemudian pengirim mengirimkan sinyal bahwa data tersebut valid atau tidak. Kemudian modul akan otomatis melakukan aksi sesuai dengan kondisi sinyal yang dikirim oleh pengirim, misalnya data tidak valid maka akan melakukan aksi-aksi tertentu. Apabila dilihat cara tersebut akan memerlukan waktu yang banyak untuk berkomunikasi.

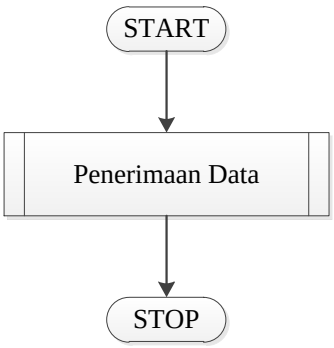
3.2.1. Komunikasi Simple System

Pada perancangan perangkat lunak komunikasi *simple system* ini mempunyai sistem yang sangat sederhana. Cara kerjanya adalah apabila tombol *start* pada PLC *transmitter* ditekan maka modul komunikasi dipanggil. Pada saat modul komunikasi dipanggil, modul akan otomatis mengecek mode yang akan dilakukan. PLC *transmitter* akan menjalankan mode pengiriman (TX) sedangkan PLC *receiver* menjalankan mode penerimaan (RX) setelah modul komunikasi diaktifkan. Untuk lebih jelasnya dapat dilihat pada gambar *flowchart simple system* di bawah ini.



Gambar 3.13 Flowchart Simple System TX

Terlihat pada *flowchart* di atas bahwa sebelum mengaktifkan modul komunikasi pengguna modul diharuskan mengisi nilai pada FW15 yang berfungsi untuk menentukan data yang akan dikirmkan. Missal pada kasus di atas diberi nilai 10 pada FW15, maka data yang dikirimkan dalam biner yaitu 1010. Sedangkan untuk program sederhana yang yang dimasukkan ke PLC *receiver* dapat dilihat di *flowchart* di bawah berikut ini.

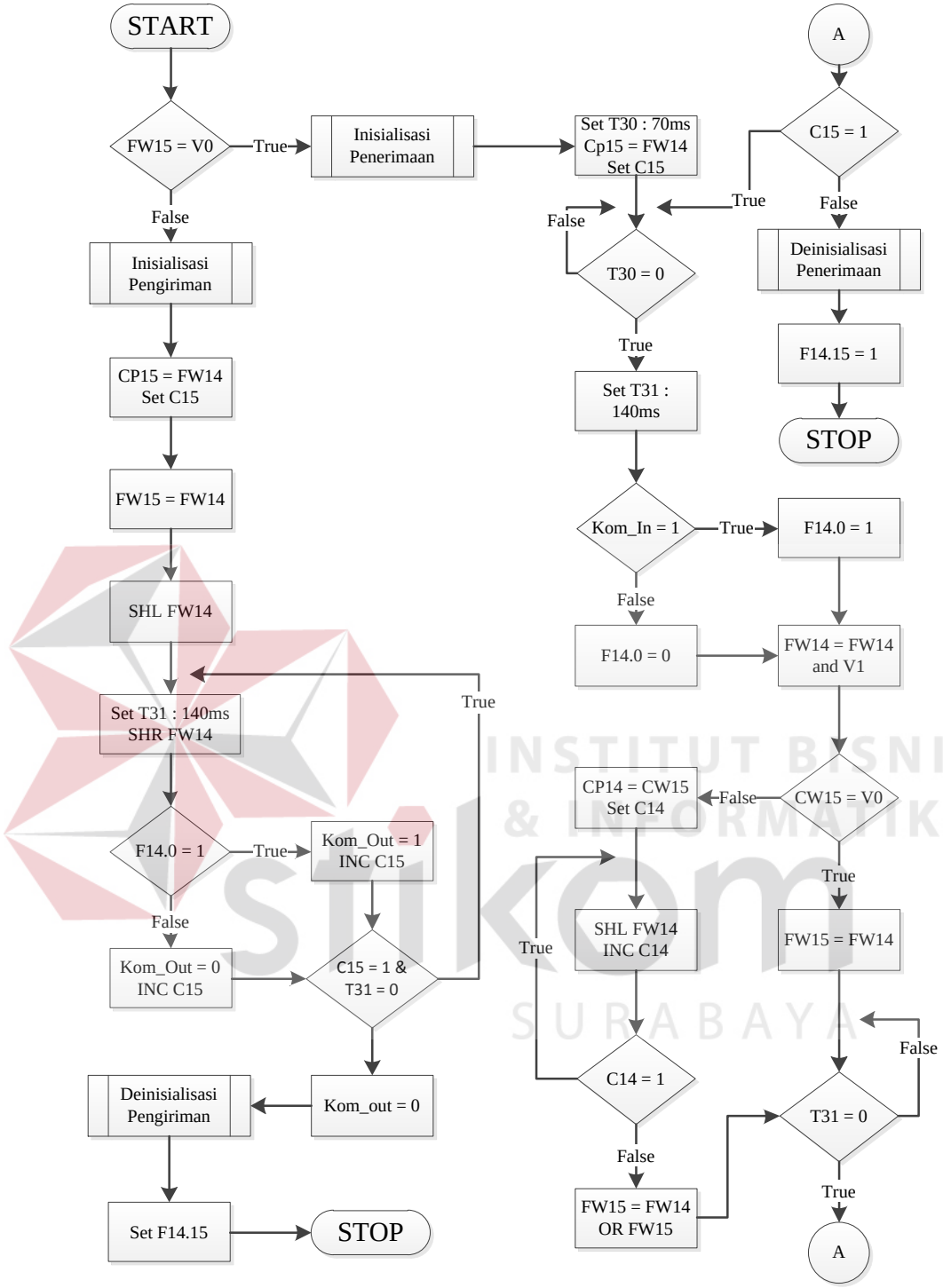


Gambar 3.14 Flowchart Symple System RX

Masalah terjadi apabila data yang akan dikirm dan diterima melebihi dari 4 bit modul komunikasi tidak bisa digunakan. Selain itu, apabila modul bisa melakukan pengiriman atau penerimaan data lebih atau kurang dari 4 *bit* akan membuat modul semakin *flexible*. *Flexible* di sini bukan tanpa memiliki batasan, batasan *bit* yang bisa diterima atau dikirim dari 1 *bit* – 15 *bit*. Hal ini dikarenakan *flag* penyimpanan data sementara (sebelum data dimasukkan ke *flag* 15 yang sudah berupa data valid, data disimpan di *flag* 14) menggunakan *flag* 14 yang hanya bisa menampung 16 bit. Sedangkan untuk *bit* terakhir pada *flag* digunakan untuk tanda komunikasi sudah selesai melakukan proses komunikasi, jadi apabila komunikasi sudah selesai dilakukan nilai *bit* F14.15 berubah menjadi 1.

Data yang dapat dikirim dan diterima maksimal hanya sebanyak 15 *bit*. Hal ini dikarenakan dalam satu word *flag* pada PLC dapat menampung 16 *bit* data.

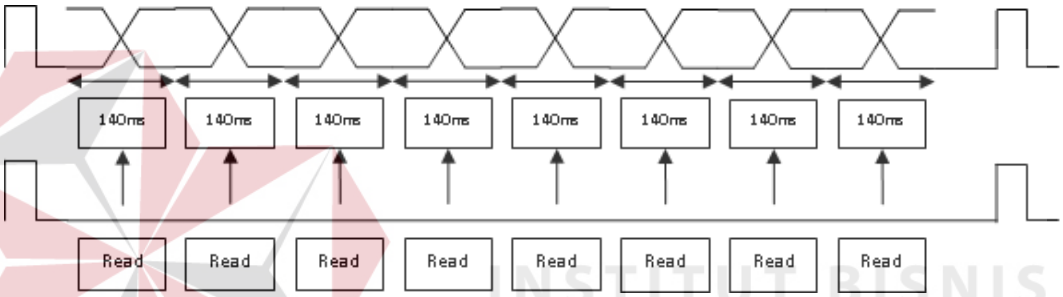
Maka dilakukan perbaikan pada *flowchart* pengiriman dan penerimaan data. Sehingga yang harus dilakukan adalah menyimpan jumlah bit yang diterima atau dikirim di sebuah *flag*, agar modul komunikasi dapat mengenali berapa bit yang akan diterima atau dikirim. Setelah dilakukan perubahan dan percobaan desain perangkat lunak, masalah kembalai didapati. Yaitu modul hanya bisa menerima data dengan valid maksimal 8 *bit* data. Untuk menyelesaikan masalah tersebut algoritma modul komunikasi coba dirubah, setelah dilakukan berbagai perubahan dan percobaan, maka didapatkan algoritma yang dapat menerima data valid maksimal mecapai 15 *bit*. Dan *timing* yang dipakai untuk pengiriman dan penerimaan data untuk 1 *bit* data masih memerlukan 140ms. Berikut adalah gambar *flowchart* dari algoritma modul komunikasi tersebut.



Gambar 3.15 *Flowchart* Modul Komunikasi Modifikasi Jumlah *Bit*

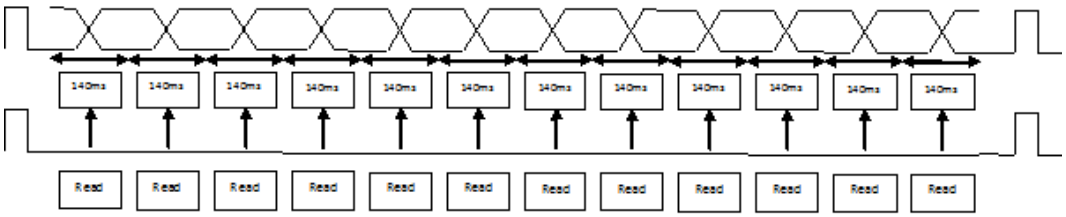
Dalam modul ini bisa mengirimkan data 4 bit, 8 bit, dan 12 bit. Sedangkan 3 bit lainnya digunakan untuk bit cadangan apabila *programmer* memerlukan bit tambahan.

Untuk gambar *timing diagram* 8 bit dapat dilihat pada gambar di bawah ini. Untuk gambar ini terdapat dua buah data yang mana satu buah data berisi 4 bit. Pada awal dan akhir pengiriman dan pembacaan data terdapat inisial dan deinisial dari komunikasi serial.



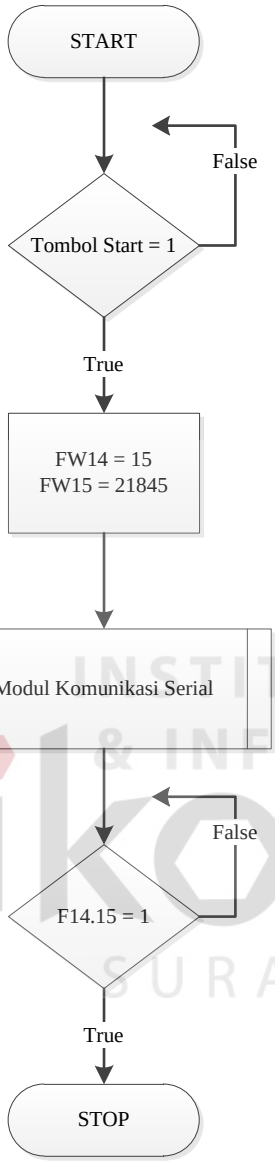
Gambar 3.16 Timing Diagram 8 bit

Sedangkan untuk gambar *timing diagram* 12 bit dapat dilihat pada gambar di bawah ini. untuk gambar ini terdapat 3 buah data yang mana satu buah data berisi 4 bit. Pada wal dan akhir pengiriman dan pembacaan data terdapat sinyal yang berarti inisial dan deinisial dari komunikasi serial ini.



Gambar 3.17 Timing Diagram 12 bit

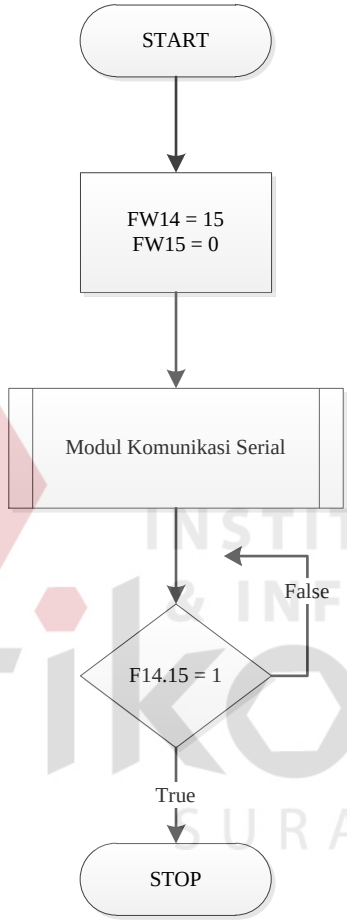
Maka untuk *flowchart simple system* pada PLC *transmitter* dapat dilihat pada gambar 3.18 di bawah ini.



Gambar 3.18 Flowchart Simple System PLC Transmitter

Pada *flowchart simple system* di atas dapat dilihat bahwa sebelum mengaktifkan modul komunikasi, user diharuskan untuk memberi nilai pada FW14 dan FW15. FW14 berfungsi untuk menyimpan jumlah *bit* yang akan dikirim, sedangkan FW15 berfungsi untuk menyimpan data yang akan dikirim.

Nilai dari FW14 adalah 15, maka jumlah *bit* yang nanti akan dikirim sebanyak 15 *bit*. FW15 mempunyai nilai 21845 yang apabila dijadikan biner, data yang akan dikirimkan adalah 101010101010101. Sedangkan untuk *flowchart* pada PLC *receiver* adalah sebagai berikut.



Gambar 3.19 Flowchart Simple System PLC Receiver

Sama seperti PLC *transmitter*, PLC *receiver* juga memberikan sebuah nilai pada FW14 dan FW15 sebelum mengaktifkan modul komunikasi serial. Fungsi dari FW14 dan FW15 juga sama, yang berbeda adalah nilai pada FW15. Apabila modul akan dipakai untuk menerima data, nilai FW15 harus dijadikan 0 (nol).

Dari semua percobaan dan perubahan pada desain modul komunikasi serial dapat disimpulkan prosedur pemasangan modul di *project* PLC dan prosedur yang harus dilakukan sebelum mengaktifkan modul komunikasi serial.

- Prosedur pemasangan modul pada *project* PLC adalah sebagai berikut :

    a. Copy file yang mempunyai format BAK, AWL, INT, LOG yang ada di SERIAL.zip ke folder *project*.

    b. Ganti nama file sesuai *module* ke berapa yang akan dijadikan *module* komunikasi. Misal, *module* 02 yang akan dijadikan sebagai *module* komunikasi :

        1) *Rename file* IZ0B00V1.BAK menjadi IZ0B02V1.BAK
        2) *Rename file* IZ0B00V1.INT menjadi IZ0B02V1.INT
        3) *Rename file* IZ0B00V1.AWL menjadi IZ0B02V1.AWL
        4) *Rename file* IZ0B00V1.LOG menjadi IZ0B02V1.LOG

    c. Buka aplikasi FST100.

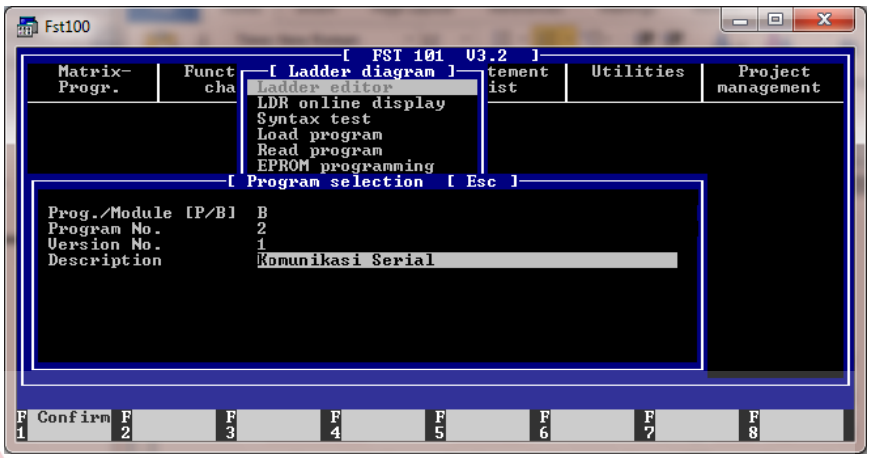
    d. Tekan F4 (pemilihan bahasa *StatementList*).

    e. Tekan F1 (pemilihan *new program*).

    f. Ubah “Prog./Module [P/B]” menjadi B, seperti pada gambar di bawah ini.

    g. Ubah “Program No.” menjadi *module* ke berapa yang akan dijadikan sebagai *module* komunikasi, kemudian ubah *description*

menjadi “Komunikasi Serial”. Misalnya *module* ke-2 yang akan digunakan untuk *module* komunikasi, perubahannya dapat dilihat pada gambar 3.20.



Gambar 3.20 Program No.

h. Tekan F1 (Confirm). Maka otomatis *module* akan terisi dengan program komunikasi serial.

i. Masukkan di *allocation list* sebagai berikut :

1) F14.0

2) F14.15

3) FW14

4) FW15

5) T30

6) T31

7) C14

8) C15

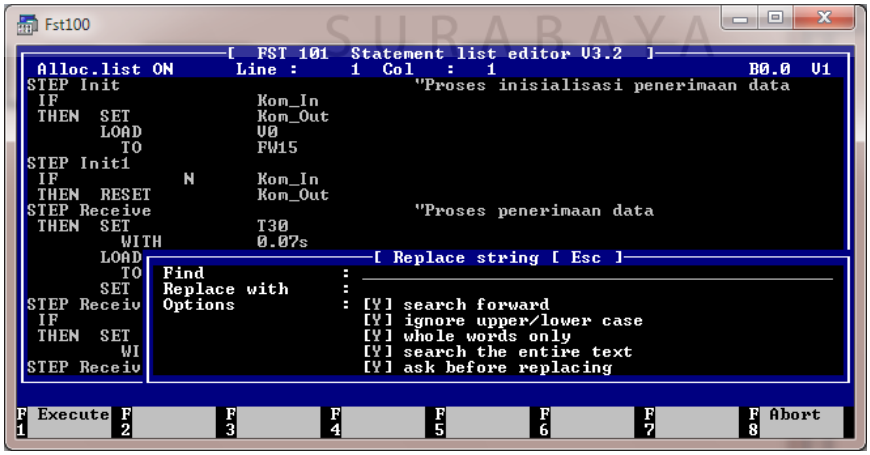
9) CP14

10) CP15

11) CW15

j. Masih di *allocation list*, user harus memasukkan I/O yang akan digunakan untuk komunikasi. Misalkan O1.5 dan I1.5 yang akan digunakan, maka *symbol operand* O1.5 diisi dengan Kom_Out dan *symbol operand* I1.5 diisi dengan Kom_In. (Catatan : Apabila *variable* Kom_Out/Kom_In ingin dirubah, perhatikan langkah k.)

k. Apabila *symbol operand* ingin diganti dengan variabel yang lain maka masuk ke *module* Komunikasi Serial, tekan F5 (Edit Commands), tekan F1 (Find/replace), tekan F2 (Replace string), maka akan muncul tampilan seperti gambar di bawah ini.



Gambar 3.21 Replace variable

Isikan *variable* yang kita ingin ganti, yaitu Kom_Out/Kom_In pada kolom “Find :”. Dan isikan *variable* yang kita inginkan untuk menggantikan *variable* yang lama pada kolom “Replace with :”. Kemudian tekan F1 dan apabila *variable* yang sedang ditunjuk *pointer* ingin diganti dengan *variable* yang sudah ditentukan tadi, tekan F1 lagi untuk menyetujui perubahan, tekan F2 apabila perubahan tidak diinginkan pada *variable* yang sedang ditunjuk *pointer*. (Catatan : Perhatikan *allocation list* dari *variable* yang ingin diganti, jangan sampai terjadi kesalahan.)

- Prosedur pemakaian *module* Komunikasi Serial

a. Beri nilai pada FW14 dengan jumlah *bit* yang akan digunakan.

b. Beri nilai pada FW15 dengan data biner yang dirubah menjadi desimal. Apabila *module* ingin menggunakan mode TX (*transmitter*) nilai FW15 tidak boleh bernilai 0 (nol), sedangkan apabila *module* ingin menggunakan mode RX (*receiver*) nilai FW15 harus bernilai 0 (nol).

c. Aktifkan *module* Komunikasi Serial

d. Apabila nilai status F14.15 bernilai 1, maka proses komunikasi sudah selesai dilakukan.

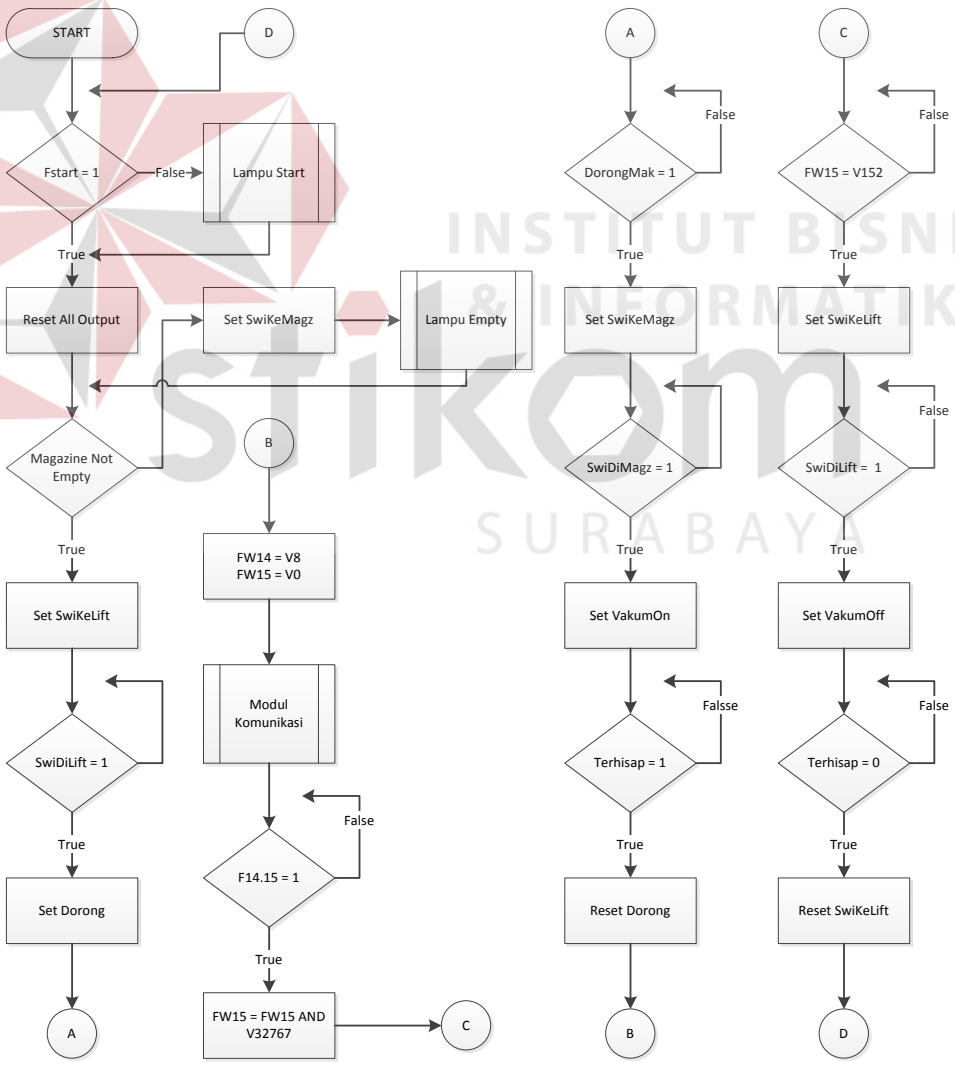
e. Apabila modul digunakan untuk penerimaan data maka diharuskan untuk melakukan operasi logika yaitu FW15 = FW15 AND

V32767. Hal ini untuk mematikan F14.15 dan menjadikan data di FW15 menjadi data yang sebenarnya atau data yang *valid*.

3.2.2. Komunikasi MPS

Perancangan perangkat lunak pada MPS dilakukan dengan cara menyelipkan *module* komunikasi serial yang sudah dibuat pada *simple system* sebelumnya di tahap-tahap tertentu dari sistem kerja MPS.

A. *Distributing Sation*



Gambar 3.22 Sistem Kerja *Distributing Station*

Tugas dari MPS *distributing station* dalam komunikasi serial ini adalah hanya menunggu MPS *testing station* mengirimkan tanda siap berkomunikasi (S_RDY) dan siap menerima benda (P_RDY). Jadi apabila tanda yang dikirimkan MPS *testing station* sudah diterima oleh MPS *distributing station*, maka MPS *distributing station* bertugas mengirimkan benda ke *testing station*.

Dapat dilihat sebelum memanggil modul komunikasi FW14 diberi nilai V8 dan FW15 diberi nilai 0 (nol), hal ini menunjukkan data yang digunakan sebanyak 8 *bit* dan modul dimaksudkan untuk penerimaan data. Sesudah pemanggilan modul komunikasi, F14.15 dicek. Apabila bernilai 1 maka komunikasi sudah selesai dilakukan. Dan kemudian FW15 yang berisi data penerimaan di AND dengan nilai 32767, hal ini dilakukan untuk mematikan F14.15 dan merubah data menjadi data yang sebenarnya. Setelah itu barulah data di FW15 bisa dipakai untuk keperluan sistem kerja *distributing station*. Contoh program dapat dilihat pada lampiran program.

B. *Testing Station*

Tugas dari MPS *testing station* pada komunikasi serial ini adalah mengirim dan menerima. Stasiun ini berhubungan dengan *distributin station* dan *processing station*. *Testing station* harus mengirimkan sinyal S_RDY dan P_RDY kepada *distribution station*. *Testing station* juga harus menunggu *processing station* mengirimkan sinyal S_RDY, P_RDY dan D_REQ. Apabila sinyal sudah diterima maka stasiun ini harus mengirimkan benda dan mengirimkan informasi karakteristik benda.

Untuk *flowchart* dari kerja sistem *testing station* dapat dilihat di gambar 3.24 di bawah ini.

Data *biner* yang dikirimkan *testing station* kepada *distribution station* adalah 10011000. Apabila data biner tersebut dipisahkan menjadi masing-masing 4 *bit* menjadi 1001 dan 1000. Data biner 1001 berarti *bit* ke-1 berstatus 1, *bit* ke-1 adalah *bit* yang menunjukkan P_RDY. Data biner 1000 berartii *bit* ke-0 yang berarti S_RDY berstatus 1.

Data *biner* yang dikirimkan *testing station* kepada *processing station* adalah 01111111 atau 11110111 atau 11111111 sesuai kondisi karakter benda yang dikirimkan. Terdapat kesamaan *bit* yang digunakan pada data-data tersebut. Yaitu menggunakan *bit* ke-7 yang dibedakan menjadi *bit* ke-7 pertama dan kedua.

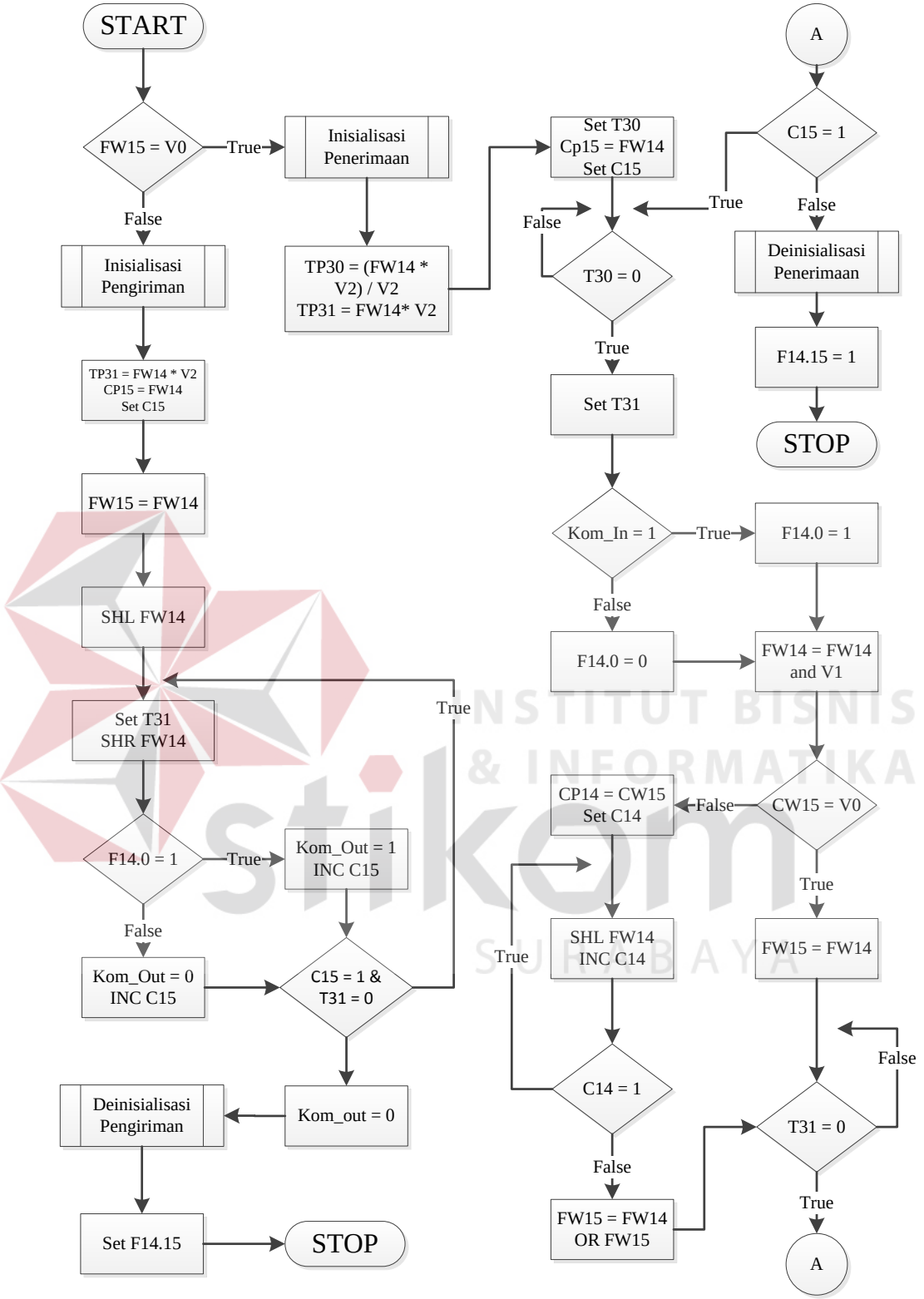
Testing station akan menunggu sinyal P_RDY, S_RDY dan D_REQ dari *processing station*. Sehingga jumlah bit yang dikirimkan adalah 12 bit karena masing-masing sinyal direpresentasikan dengan 4 *bit* data. Data tersebut adalah 101010011000. Setelah dilakukan komunikasi, ternyata data valid yang bisa diterima hanya 10 bit. Disimpulkan bahwa *error* tersebut disebabkan *timer* yang digunakan untuk mengirim data terlalu cepat, sehingga waktu *timer* perlu diperpanjang.

Maka desain pengiriman dan penerimaan data dirubah menjadi lebih *fleksibel* dalam penggunaan *timer*. Maksudnya adalah penggunaan *timer* ditentukan dari jumlah *bit* yang digunakan, sehingga semakin banyak jumlah *bit* maka semakin panjang waktu *timer* yang digunakan.

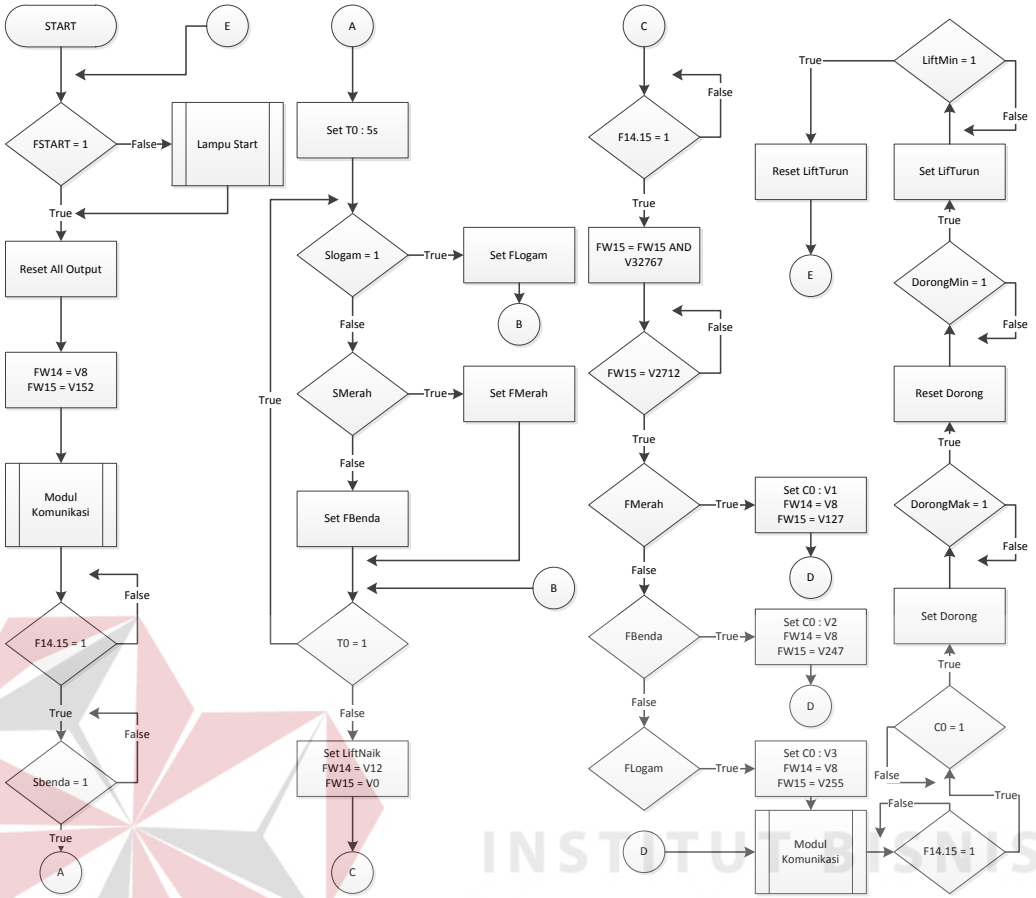
Dalam desain ini ditentukan untuk 1 *bit* membutuhkan waktu sebesar jumlah *bit* dikali 20ms. Penggunaan 20ms karena pada saat dilakukan percobaan dengan jumlah bit dikali 10ms, ternyata data yang diterima belum valid. Karena PLC tidak bisa mengalikan waktu timer dengan menggunakan bilangan pecahan maka nilai yang bisa digunakan adalah 20ms. Setelah menggunakan nilai 20ms ternyata data bisa diterima dengan valid. Gambar *flowchart* modul komunikasi yang sudah dilakukan perubahan dapat dilihat pada gambar 3.23.

Memang terdapat perbedaan pada hasil komunikasi dengan menggunakan modul komunikasi ini pada *simple system* dan MPS. MPS yang lebih banyak mempunyai *input* dan *output*, serta lebih banyak memiliki proses dapat mengakibatkan proses modul komunikasi *error*. Berbeda pada *simple system* yang tidak memiliki banyak *input*, *output* dan proses dapat mengirim dan menerima data dengan baik.

Maka disimpulkan bahwa semakin banyak proses dalam sebuah PLC akan mengganggu kerja *timer* pada PLC tersebut.



Gambar 3.23 *Flowchart* Modul Komunikasi Modifikasi *Timer*

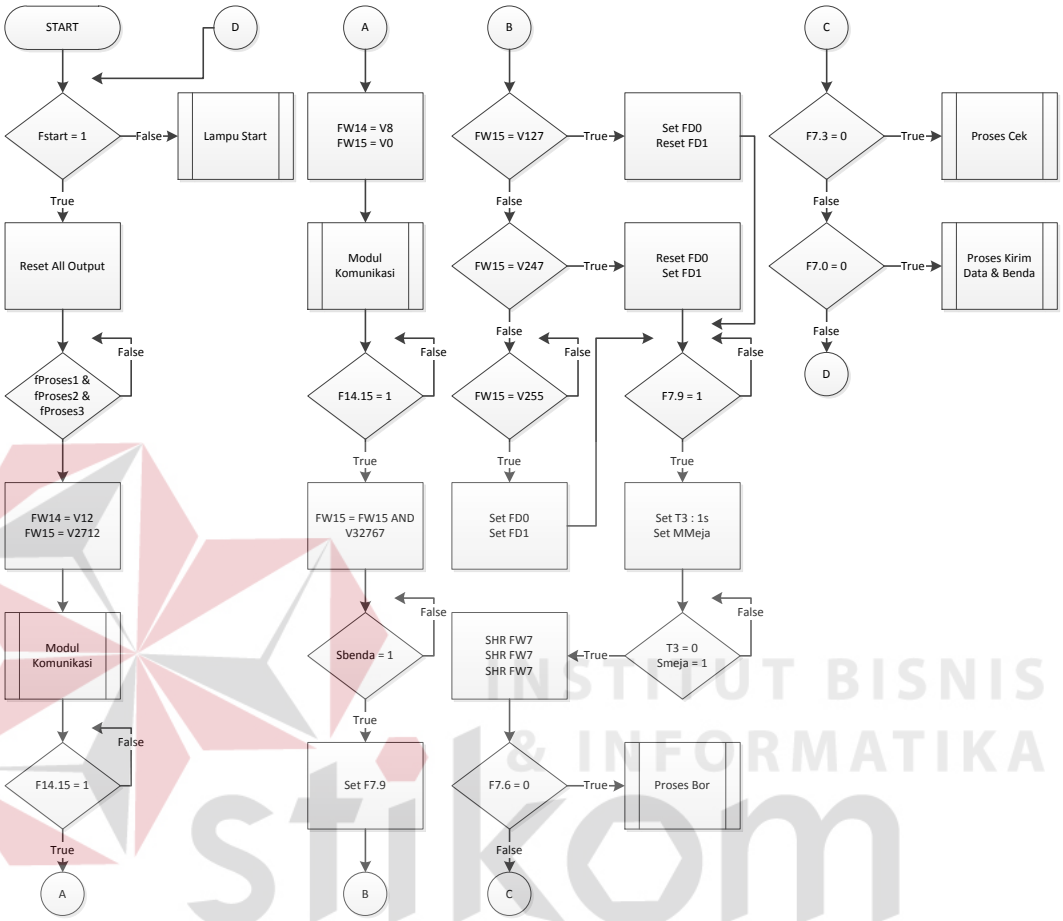


Gambar 3.24 Sistem Kerja *Testing Station*

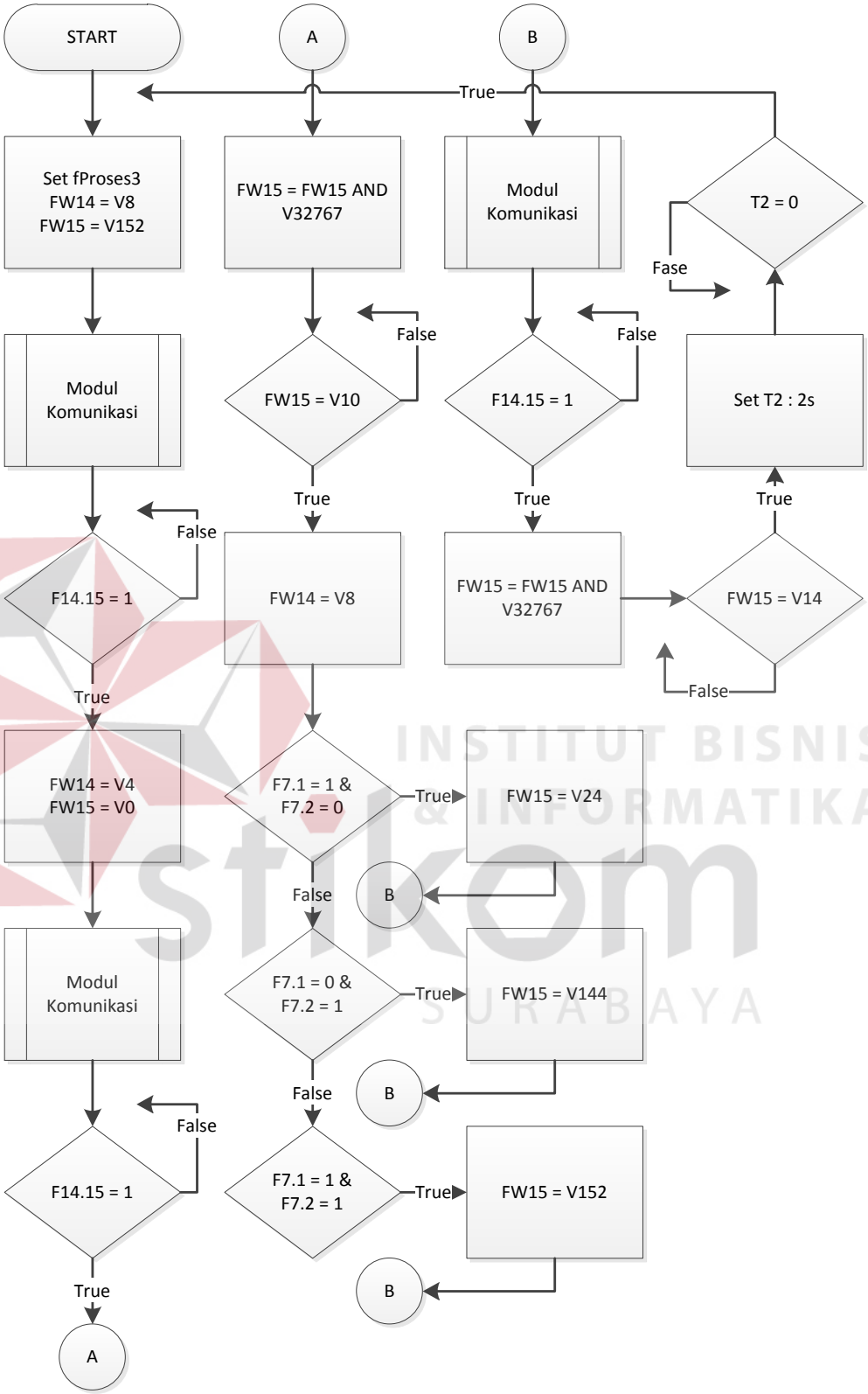
C. *Processing Station*

Tugas dari *processing station* adalah mengirimkan data S_RDY, P_RDY dan D_REQ kepada *testing station*. Selain itu stasiun ini juga harus menyimpan data-data yang masuk dan nantinya akan dikirimkan ke *handling station*. Data-data tersebut adalah data karakteristik benda yang akan dikirimkan ke *handling station*. Data-data tersebut menunjukkan karakteristik benda. Jadi data yang dikirimkan dalam biner adalah 00011000 yang berarti *bit* ke-0 berstatus 1 dan *bit* ke-1 berstatus 0 (nol), 10010000 yang berarti *bit* ke-0 berstatus 0 (nol) dan *bit* ke-1 berstatus 1, 10011000 yang berarti *bit* ke-0 dan *bit* ke-1 berstatus 1. Gambar 3.25

adalah *flowchart* sebagian *system* kerja *processing station* dan gambar 3.26 adalah *flowchart* pengiriman data dan benda ke *handling station*.



Gambar 3.25 Sistem Kerja *Processing Station*



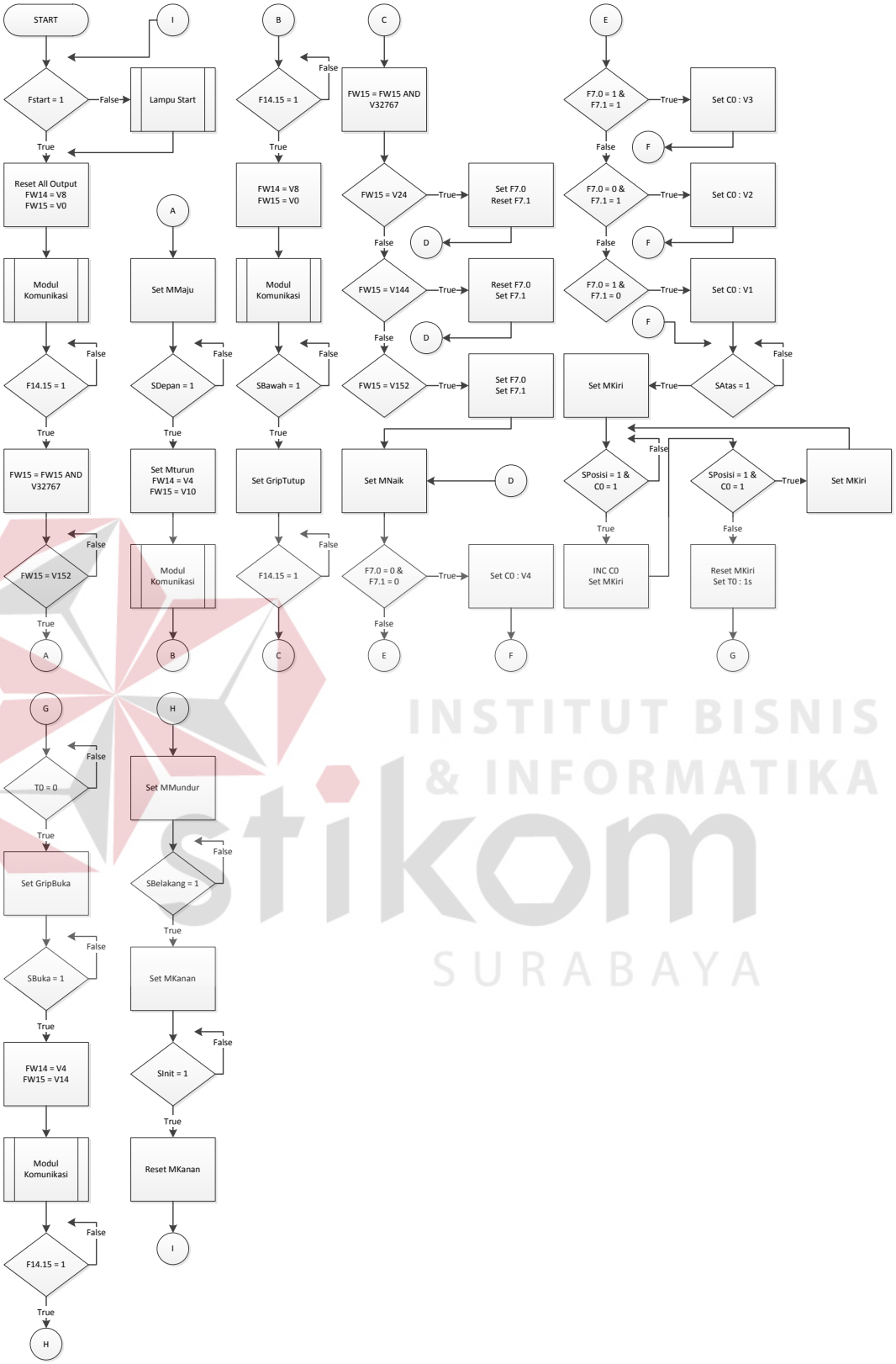
Gambar 3.26 Pengiriman Benda dan Data Ke *Handling Station*

D. *Handling Station*

Tugas dari *handling station* adalah mengambil benda dan menerima data dari *processing station*. *Handling station* akan menunggu sinyal S_RDY dan P_RDY dari *processing station*. Apabila sinyal sudah didapat, stasiun ini harus mengambil benda yang sudah disediakan *processing station*. Setelah benda diambil stasiun ini meminta data berupa karakteristik benda. Apabila data sudah diterima maka *handling station* harus siap berkomunikasi lagi. Ternyata hal ini menyebabkan *error* pada MPS *handling station*.

Berbagai percobaan dan perubahan dilakukan pada *handling station*. Maka didapatkan algoritma yang bisa membuat program dari stasiun ini berjalan lancar. *Flowchart* sebagian kerja dari *handling station* dapat dilihat pada gambar 3.27. Pada algoritma dilakukan perubahan pada alur pelaksanaan komunikasi. Yaitu apabila sebelumnya stasiun ini harus sudah siap melakukan komunikasi lagi setelah data dari karakteristik benda didapat dan benda tersebut sudah diambil, setelah dilakukan modifikasi algoritmanya menjadi apabila data dari benda didapat dan benda sudah diambil, stasiun belum boleh melakukan komunikasi dulu sampai benda sudah diletakkan ke tempatnya dan *hand* dari stasiun ini menuju posisi awal.

Analisis sementara adalah apabila komunikasi harus segera dilakukan dan pada waktu bersamaan PLC masih melakukan pekerjaan yang banyak, maka akan mengganggu komunikasi.



Gambar 3.27 Sistem Kerja *Handling Station*

3.2.3. Komunikasi Antar PLC Pada Sistem *Non-Pneumatic*

Perancangan perangkat lunak komunikasi pada sistem *non-pneumatic* dilakukan dengan menjalankan semua modul secara *multitasking*, sehingga akan membuat CCU pada PLC akan berkerja cukup keras . Modul PLC yang berjumlah 7 buah modul digunakan secara bersamaan atau dengan kata lain aktif dalam satu waktu bersamaan. Pada modul ke-0 digunakan untuk modul komunikasi serial sedangkan modul ke-1 sampai modul ke-6 digunakan untuk menyalakan dan mematikan lampu dengan desain nyala lampu yang sudah ditentukan.

Modul ke-1 digunakan untuk menyalakan lampu pada O0.0, O0.1, O0.2 secara bergantian dimulai dari O0.0 sampai O0.2. Sedangkan modul ke-2 juga digunakan untuk menyalakan lampu pada O0.4, O0.5, O0.6 secara bergantian dimulai dari O0.6 sampai O0.4. Seluruh modul menggunakan *interval* waktu 0,5 detik untuk masing-masing lampu menyala.

Modul ke-3 menyalakan lampu O0.3 dan O0.7 secara bergantian dimulai dari O0.3 kemudian O0.7. Sedangkan modul ke-4 menyalakan lampu O1.0 dan O1.3 secara bergantian dimulai dari O1.3 kemudian O1.0. Dan untuk modul ke-5 menyalakan lampu O1.1 dan O1.2 secara bergantian, tetapi modul ini bekerja sama dengan modul ke-6 yang menyalakan lampu O1.3. Sehingga O1.1, O1.2 dan O1.3 menyala bergantian tetapi dikendalikan oleh dua modul sekaligus.

Apabila kita amati semua modul sudah dijalankan secara multitasking, sehingga membuat PLC bekerja cukup keras. Disamping PLC menjalankan semua modul yang mengendalikan lampu-lampu tersebut, PLC *transmitter* juga melakukan komunikasi dengan PLC *receiver* secara terus-menerus. Jadi apabila

ada sebuah benda maka PLC *transmitter* akan mengecek karakteristik benda tersebut dan mengirimkan data karakteristik tersebut ke PLC *receiver*.