STUDI TENTANG MODBUS PROTOKOL PADA SISTEM KONTROL

Oleh : Nurpadmi *)

SARI

Perkembangan sistem kontrol di industri tidak akan lepas dari sistem komunikasi data. Untuk mengirimkan data dari lapangan (sensor) ke kontroler, untuk komunikasi antar kontroler, untuk mengirimkan data dari kontroler ke komputer, semua membutuhkan komunikasi data. Salah satu sistem komunikasi data yang banyak digunakan pada sistem kontrol di industri adalah modbus protokol. Bahkan di dunia pendidikan juga banyak penelitian yang mengembangkan sistem kontrol menggunakan komunikasi modbus.

Artikel ini mencoba memberikan gambaran ringkas tentang bagaimana sistem komunikasi dengan Modbus Protokol.

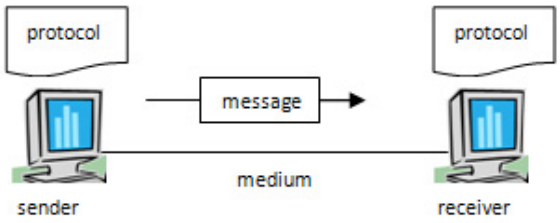
A. Pendahuluan

Perkembangan industri dewasa ini, berjalan amat pesat seiring dengan meluasnya jenis produk-produk industri, mulai dari industri hulu sampai dengan industri hilir. Kompleksitas pengolahan bahan mentah menjadi bahan baku, yang berproses baik secara fisika maupun secara kimia, telah memacu manusia untuk selalu meningkatkan dan memperbaiki unjuk kerja sistem yang mendukung proses tersebut, agar semakin produktif dan efisien. Salah satu yang menjadi perhatian utama dalam hal ini ialah penggunaan sistem pengendalian proses industri (sistem kontrol industri) [4].

Dalam era industri modern, sistem kontrol industri biasanya merujuk pada otomatisasi sistem kontrol yang digunakan. Sistem kontrol industri dimana peranan manusia yang amat dominan telah banyak digeser dan digantikan oleh sistem kontrol otomatis [4]. Dimana pada sistem kontrol otomatis ini mutlak digunakan sistem komunikasi data. Untuk mengirimkan data dari lapangan (sensor) ke kontroler, untuk komunikasi antar kontroler, untuk mengirimkan data dari kontroler ke komputer, semua membutuhkan komunikasi data.

Komunikasi data sebenarnya sudah sangat sering dilakukan dalam aktifitas sehari-hari, baik dalam lingkup global seperti internet (email, facebook), maupun dalam lingkup lokal (LAN) seperti mengakses data dari komputer lain melalui jaringan, bahkan handpone yang kita gunakan sehari – hari juga membutuhkan komunikasi data. Sebagai ilustrasi, pada saat suatu message dikirimkan dari suatu sensor dan kemudian diterima oleh kontroler, pada saat itulah terjadi komunikasi di antara kedua peralatan tersebut.

Komunikasi yang terjadi di antara kedua alat di atas, dibentuk oleh komponen-komponen yang membentuk suatu sistem yaitu sistem komunikasi data. Secara garis besar, terdapat lima jenis komponen yang saling berinteraksi membentuk sistem komunikasi data tersebut yaitu: message, sender, receiver, medium, dan protocol [5].



Gambar 1. Sistem Komunikasi Data Sederhana [5]

Dimana mesage merupakan data yang akan dikomunikasikan, sender adalah alat yang digunakan untuk mengirimkan data, receiver sebagai alat penerima data yang dikirim, medium merupakan media transmisi yang bisa dikatakan sebagai "perantara" untuk mengantarkan message dari sender ke receiver, yang biasanya berupa kabel (twisted pair, coaxial, fiber-optic), laser, atau gelombang radio. Protokol sendiri merupakan sekumpulan aturan yang harus disepakati oleh dua atau lebih alat untuk dapat saling berkomunikasi. Tanpa protokol, dua alat atau lebih mungkin saja bisa saling terhubung tetapi tidak dapat saling berkomunikasi, sehingga message yang dikirim tidak dapat diterima oleh alat yang dituju. Untuk mudahnya protokol bisa dianggap sebagai bahasa komunikasi, seseorang yang berkomunikasi menggunakan bahasa jawa tidak akan bisa dimengerti oleh orang lain yang hanya bisa berbahasa cina.

Didalam sistem kontrol, protokol dapat dikatakan sebagai sebuah standar bahasa yang digunakan oleh sejumlah peralatan mikrokomputer seperti PLC, meter, DCS, *protective relays*, dll untuk dapat saling berkomunikasi. Salah satu jenis protokol yang sangat "akrab" digunakan di lapangan dan didalam sistem kontrol adalah "Modbus". Modbus merupakan suatu protokol komunikasi serial, dimana komunikasi dapat terjadi dengan adanya sistem modbus master dan modbus slave.

B. Rumusan Masalah

Dewasa ini, penggunaan sistem komunikasi modbus di dunia otomasi industri menjadi sangat marak. Banyak produsen instrumen maupun peralatan industri yang mengembangkan sistem dengan komunikasi modbus. Sistem komunikasi modbus sendiri ada bermacan-macam, ada modbus RTU (serial), modbus ASCII (serial), modbus TCP/IP (melalui jaringan LAN), modbus plus, dll. Modbus yang paling banyak digunakan saat ini adalah modbus RTU. Pada tulisan ini, lebih difokuskan pada modbus serial (RTU dan ASCII). Dan beberapa masalah yang dirumuskan adalah sebagai berikut :

1. Bagaimana perbedaan sistem komunikasi serial (RTU dan ASCII)?
2. Bagaimanakah sistem komunikasi modbus serial tersebut bekerja ?
3. Bagaimana bentuk frame data pada modbus RTU dan ASCII ?

C. Modbus Protokol

Modbus adalah salah satu protokol untuk komunikasi serial yang di publikasikan oleh Modicon pada tahun 1979 untuk di gunakan pada PLC Modicon (PLC pertama di dunia yang di kembangkan oleh Schneider). Secara sederhana, modbus merupakan metode yang digunakan untuk mengirimkan data/informasi melalui koneksi serial antar perangkat elektronik. Perangkat yang meminta informasi disebut Modbus Master dan perangkat penyediaan informasi disebut Modbus Slave. Pada jaringan Modbus standar, terdapat sebuah master dan slave sampai dengan 247, masing-masing mempunyai Alamat Slave yang berbeda mulai dari 1 sampai 247. Master juga dapat menulis informasi kepada Slave.

Modbus merupakan sebuah open protokol, yang berarti bahwa dapat digunakan dalam peralatan tanpa harus

membayar royalti. Modbus telah menjadi protokol komunikasi standar dalam industri, dan sekarang paling banyak dipakai untuk menghubungkan perangkat elektronik industri. Modbus digunakan secara luas oleh banyak produsen di banyak industri.

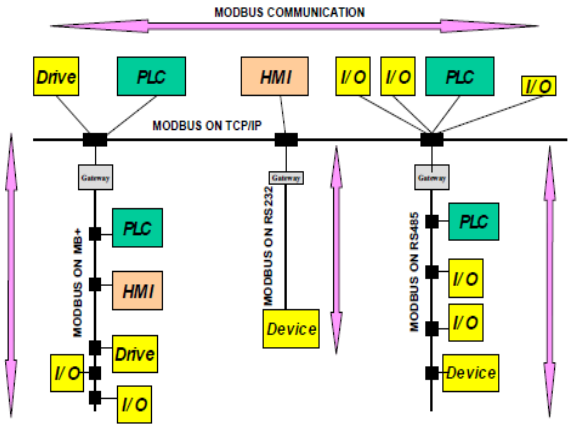
Protokol ini menjadi standard komunikasi dalam industri dan menjadi yang paling banyak dipakai untuk komunikasi antar peralatan elektronik pada industri. Alasan utama mengapa Modbus Protokol banyak di gunakan adalah [7]:

1. Di publikasikan secara terbuka tanpa royalty fee untuk penggunaannya.
2. Relatif mudah untuk di aplikasikan pada industrial network.
3. Modbus mempunyai struktur bit tanpa memiliki banyak larangan bagi vendor lain untuk mengaksesnya.

Modbus memungkinkan adanya komunikasi dua-jalur antar perangkat yang terhubung ke jaringan yang sama, misalnya suatu sistem yang mengukur suhu, tekanan, kelembaban dsb, kemudian mengkomunikasikan hasilnya ke komputer (HMI/Human Machine Interface). Modbus sering digunakan untuk menghubungkan sopervisory computer dengan remote terminal unit (RTU), supervisory control dan sistem akuisisi data (SCADA).

Produsen atau suplier besar maupun kecil, system integrator, end-user, pengembang open source, dosen dan pihak yang berkepentingan lainnya dapat menjadi anggota Modbus. Beberapa anggota yang menonjol adalah SoftDEL Systems, Precision Digital Corporation, Motor Protection Electronics, FieldServer Technologies dan masih banyak lagi [7].

Berdasarkan media transfernya, Modbus dikategorikan ke dalam Modbus serial (RS232/485) dan Modbus Ethernet (TCP/IP). Jika dirujuk dari bentuk datanya, Modbus dibagi ke dalam Modbus RTU (serial) dan Modbus ASCII. Pada Modbus serial digunakan istilah Master/Slave sedangkan Modbus Ethernet biasanya memakai terminologi Server/Client.



Gambar 2 : Contoh Arsitektur Modbus [3]

Protokol Modbus memungkinkan komunikasi yang mudah di semua jenis arsitektur jaringan. Setiap jenis perangkat (PLC, HMI, Kontrol Panel, Driver, kontrol Motion, I / O Device dll) dapat menggunakan protokol Modbus untuk operasi remote. Komunikasi yang sama dapat dilakukan juga pada serial line seperti pada Ethernet TCP / IP. Gateway memungkinkan komunikasi antara beberapa jenis bus atau jaringan dengan menggunakan protokol Modbus.[3].

D. Prinsip Modbus Protokol (Serial)

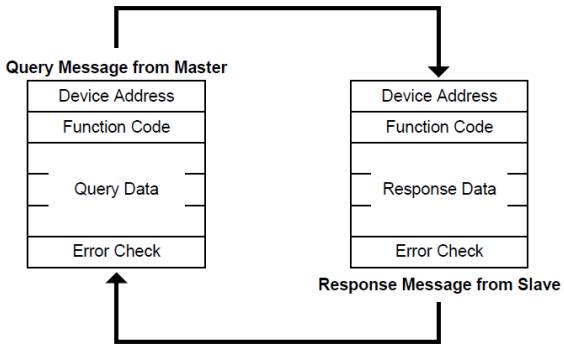
Perangkat Modbus berkomunikasi menggunakan teknik master-slave, dimana hanya satu perangkat (master) yang dapat melakukan transaksi atau melakukan perintah atau permintaan yang disebut ‘queries’. Perangkat lain (slave) merespon dengan menyediakan data yang diminta untuk master , atau dengan melakukan aksi sesuai dengan yang diminta dalam query.

Master dapat mengirimkan request ke satu slave secara individu, atau dapat mengirim pesan broadcast ke semua slave. Slave memberikan respon untuk pertanyaan

yang ditujukan kepada mereka secara individu. Sedangkan query broadcast dari master tidak akan diberikan respon oleh slave.

Protokol Modbus mempunyai format tertentu untuk setiap query dari master yang berisi alamat dari perangkat (slave) yang dituju, kode fungsi yang mendefinisikan aksi yang diminta, data yang akan dikirim, dan pemeriksaan kesalahan. Pesan respon slave juga mempunyai format tertentu dalam protokol Modbus. Format ini berisi tentang konfirmasi tindakan yang dilakukan, data yang akan dikirim, dan bidang pemeriksaan kesalahan.

Proses Query-Response pada protokol modbus mempunyai format seperti gambar 3.



Gambar 3 : Contoh Arsitektur Modbus [3]

Format untuk query master, device address merupakan alamat slave yang akan diambil datanya, *Function Code* merupakan kode fungsi yang mendefinisikan aksi yang diminta, dimana beberapa contoh kode fungsi tersebut adalah sebagai berikut:

01 : *read* DO *(Digital Output)*

02 : *read* DI *(Digital Input)*

03 : *read* AO *(Analog Output)*

04 : *read* AI *(Analog Input)*

05 : *write single* DO *(Digital Output*

06 : *write single* AO *(Analog Output)*

15 : *write multiple* DO *(Digital Output)*

16 : *write multiple* AO *(Analog Output).*

Query data merupakan blok data informasi dan *Error Check* merupakan pemeriksaan kesalahan (cek data dari kesalahan komunikasi). Respon slave pada protokol Modbus juga mempunyai format yang sama, hanya pada function code berisi konfirmasi tindakan yang dilakukan, Respond merupakan data yang akan dikembalikan, dan error check sebagai bidang pemeriksaan kesalahan.

Sistem komunikasi pada jaringan standar Modbus mempunyai dua mode transmisi: ASCII (American Standard Code for Informasi Interchange) atau RTU (Remote Terminal Unit). Pada satu jaringan modbus, mode transmisi untuk semua perangkat/device yang terhubung harus sama.

Dalam mode ASCII, setiap byte 8-bit dalam pesan yang dikirim sebagai dua karakter ASCII. Dalam modus RTU, setiap byte 8-bit dalam pesan berisi dua buah 4-bit karakter heksadesimal. Modus RTU, dengan kepadatan yang lebih besar karakter nya, memungkinkan throughput data yang lebih baik dari ASCII untuk baud rate yang sama.

Algoritma yang digunakan dalam memeriksa kesalahan, tergantung pada metode transmisi yang digunakan; LRC (Longitudinal Redundancy Check) dalam mode ASCII; CRC (Redundancy Check Siklis) dalam mode RTU.

E. Format Frame / Bingkai Data Pada Modbus

Pada tiap jenis Modbus menggunakan format frame/bingkai data yang berbeda.

Modbus RTU , digunakan dalam komunikasi serial & menggunakan representasi nilai data biner yang

dipadatkan sebagai protokol komunikasi. Format RTU mengikuti request perintah / transfer data, dengan CRC checksum sebagai mekanisme pemeriksaan kesalahan (error-check) untuk memastikan keandalan data. Modbus RTU adalah implementasi yang paling umum dari semua versi Modbus yang ada. Sebuah pesan Modbus RTU harus dikirimkan secara terus menerus tanpa jeda antar-karakter. Setiap pesan Modbus dibingkai atau dipisahkan oleh periode-periode saat idle (tanpa komunikasi atau Port komunikasi ditutup atau OFF). Komunikasi via Modbus RTU sering dipakai dalam sistem monitoring skala kecil dimana sensor-sensor dan komputer HMI letaknya tidak sangat jauh [7]. Adapun gambaran bentuk format frame pada modbus RTU tersebut dapat dilihat pada gambar 4.

START	ADDRESS	FUNCTION	DATA	CRC CHECK	END
T1–T2–T3–T4*	8 BITS	8 BITS	*n* x 8 BITS	16 BITS	T1–T2–T3–T4*

*For T1–T2–T3–T4, 3.5 character times at no communication.

Gambar 4, format frame data pada modbus RTU [2]

Dalam mode RTU, pesan dimulai dengan interval minimal 3,5 kali karakter, dan diakhiri dengan interval yang sama minimal 3,5 kali karakter. Hal ini paling mudah diimplementasikan sebagai beberapa kali karakter pada baud rate yang sedang digunakan pada jaringan (ditampilkan sebagai T1 - T2 - T3 - T4 pada gambar di atas). Semua bagian frame lain terdiri dari 8-bit data. Untuk lebih jelas dapat dilihat pada ilustrasi format frame berikut.

Format Frame / Bingkai Data Modbus RTU (Serial) - untuk satu kali transfer data		
Nama	Panjang Data	Fungsi Layer / Lapisan Data Biner
Start	3.5c idle	Setidaknya 3.5 kali karakter antar frame untuk waktu jeda (keheningan/silence) setiap proses transfer data
Address	8 bits 8 bit	Station Address atau Alamat Perangkat/Slave yang Dituju
Function Code	8 bits 8 bit	Indikasi Function Codes seperti Read Coil, Read Register, Write Register dsb.
Data	n * 8 bits n * 8 bit	Data + Length, akan terisi bergantung pada jenis pesan yang dikirim/diterima
CRC Check	16 bits 16 bit	Error checks atau Checksum kesalahan transfer data
End	3.5c idle	Setidaknya 3.5 kali karakter antar frame untuk waktu jeda (keheningan/silence) setiap proses transfer data

Gambar 5. ilustrasi frame data pada modbus RTU

Modbus ASCII, Protokol Modbus jenis ini digunakan pada komunikasi serial dan memanfaatkan karakter ASCII dalam berkomunikasi di dalam satu protokol. Format data ASCII menggunakan sebuah LRC checksum di dalam memeriksa kesalahan transfer data. Dalam mode ASCII, pesan dimulai dengan ‘colon’ (:) karakter (ASCII 0x3A), dan diakhiri dengan 'carriage return - line feed' (CRLF) pasangan (ASCII 0x0D dan 0x0A). Karakter yang diijinkan dikirim adalah heksadesimal 0 - 9, A - F. Komunikasi antar perangkat elektronik dengan komputer melalui Protokol Modbus ASCII umumnya digunakan dalam jaringan telepon atau pun Modem GSM apabila tidak tersedia infrastruktur jaringan yang memadai seperti LAN atau jaringan Fiber-Optic.

START	ADDRESS	FUNCTION	DATA	LRC CHECK	END
1 CHAR :	2 CHARS	2 CHARS	*n* CHARS	2 CHARS	2 CHARS CRLF

Gambar 6. format frame data pada modbus ASCII [2]

Format Frame Modbus ASCII		
Nama	Panjang Data	Fungsi Layer
Start	1 char	Dimulai dengan "titik dua" atau colon (:), nilai ASCII colon dalam hexadecimal adalah 3A)
Address	2 chars	Station Address atau Alamat Perangkat/Slave yang Dituju
Function Code	2 chars	Indikasi Function Codes seperti Read Coil, Read Register, Write Register dsb.
Data	n chars	Data + Length, akan terisi bergantung pada jenis pesan yang dikirim/diterima
LRC Check	2 chars	Error checks atau Checksum kesalahan transfer data
End	2 chars	Carriage Return – Line Feed (ditulis: CRLF) pair/sepasang, nilai karakter ASCII ini jika ditulis dalam bilangan hexadecimal adalah 0D & 0A hex

Gambar 7. ilustrasi frame data pada modbus ASCII [2]

F. Jenis Fuction Code Pada Modbus

Function Code / tipe data pada Modbus meliputi beberapa jenis berikut ini. Dan yang paling sering digunakan dalam aplikasi industri ditandai dengan *huruf miring*.

- *01 Read Coil Status atau Baca Status Coil*
- *02 Read Input Status* atau *Baca Status Input*
- *03 Read Holding Registers* atau *Baca Register Holding*
- *04 Read Input Registers* atau *Baca Register Input*
- *05 Force Single Coil* atau *Rubah Status Single Coil (0 atau 1)*
- *06 Preset Single Register* atau *Mengisi/Menulis Data pada Single Register*
- 07 Read Exception Status
- 08 Diagnostics
- 09 Program 484
- 10 Poll 484
- 11 Fetch Communication Event Counter (Ambil Penghitung Event Komunikasi)
- 12 Fetch Communication Event Log (Ambil Event Log Komunikasi)
- 13 Program Controller
- 14 Poll Controller
- *15 Force Multiple Coils* atau *Merubah Status Banyak Coils*
- *16 Preset Multiple Registers* atau *Menulis/Mengisi Data Lebih dari Satu Register*
- 17 Report Slave ID
- 18 Program 884/M84
- 19 Reset Comm. Link
- 20 Read General Reference
- 21 Write General Reference
- 22 Mask Write 4X Register
- 23 Read/Write 4X Registers
- 24 Read FIFO (First In Fisrt Out) Queue

G. Aplikasi Modbus Protokol

Pada penjelasan berikut dapat diberikan contoh Proses Query-Response untuk membaca register holding pada protokol modbus.

Contoh:

[MASTER] dimisalkan kita akan me-request ke slave dengan id/address 07, mengambil holding register di slave dengan register 40201 sampai dengan 40204. maka bentuk pesannya dalam Hexadesimal:

[07] [03] [00] [C8] [00][04] [CRC1] [CRC2]

[Address]	[Function Code]	[2*Panjang data]	[data]	[data]	[data]	[data]	[CRC1][CRC2]
8 Bits	8 Bits	8 Bits	2 x 8 Bits	2 x 8 Bits	2 x 8 Bits	2 x 8 Bits	16 Bits
[07]	[03]	[08]	[00] [07]	[00] [06]	[00] [05]	[00] [04]	[CR1][CR2]

Gambar 7. frame data pada query membaca register holding

Seperti halnya pada bentuk frame pada Modbus Protokol, terutama pada mode RTU, pesan dari master ini dimulai dengan interval minimal 3,5 kali karakter, dan diakhiri dengan interval yang sama minimal 3,5 kali karakter. Address diatas merupakan alamat slave yang dituju. Fuction code merupakan jenis-jenis fungsi yang dapat digunakan/diminta oleh master. Data disini terdiri dari [reg] dan [lenght] dimana [reg] merupakan register yang ingin diambil nilainya, frame ini menggunakan 2 byte data (2 x 8bits). Untuk [lenght] sendiri merupakan jumlah register yang ingin direquest, menggunakan 2 byte frame seperti halnya register. [CRC] adalah *Cyclic*

Redundancy Check yaitu sebuah metode untuk pengecekan error, menggunakan 2 byte frame juga.

Pada contoh diatas, bentuk pesan permintaan untuk mengambil holding register di slave 07 tersebut yang dituliskan dalam bentuk **[07] [03] [00] [C8] [00][04] [CRC1] [CRC2]**, dapat dijelaskan sbb:

[07] : karena slave ID nya adalah 7

[03] : karena me-request holding register (sesuai table function code)

[00] [C8] : karena 40201 adalah register pertama maka alamat 40201 - 40001 = 200 = C8 hex)..

[00] [04] : karena ada 4 data yang akan di request 40201 – 40204

[CRC1] [CRC2] : Hasil CRC

Kemudian Slave akan memberikan balasan, dimisalkan register 40201 bernilai 7, 40202 bernilai 6, 40203 bernilai 5, dan 40204 bernilai 4. maka bentuk pesan dari respon slave tersebut adalah:

[07] [03] [08] [00] [07][00] [06][00] [05] [00] [04] [CRC1] [CRC2]

Start	[Address]	[Function Code]	[Reg][Reg]	[Lenght][Lenght]	[CRC1][CRC2]	End
3.5 caracter	8 Bits	8 Bits	2 x 8 Bits	2 x 8 Bits	16 Bits	3.5 caracter
3.5 caracter	[07]	[03]	[00] [C8]	[00] [04]	[CR1][CR2]	3.5 caracter

Gambar 8. frame data pada response membaca register holding

Bentuk frame pada rensponse membaca register holding hampir sama dengan query, pesan dari master ini dimulai dengan interval minimal 3,5 kali karakter, dan diakhiri dengan interval yang sama minimal 3,5 kali karakter, namun untuk mempersingkat, tidak digambarkan dalam gambar 8. Address diatas merupakan alamat slave yang memberi response. Fuction code merupakan jenis-jenis fungsi yang dapat digunakan/diminta oleh master. 2*panjang data merupakan banyaknya data yang ditampilkan dikalikan 2, sehingga 2*4 = 8 [08]. [data][data][data][data] merupakan data register dari register pertama sampai keempat, karena ada 4 register yang dibaca dan masing-masing bernilai 2 byte. [CRC] adalah *Cyclic Redundancy Check* yaitu metode untuk pengecekan error, menggunakan 2 byte frame juga.

Penjelasan:

[07] : Slave ID/address 07

[03] : Function (Read Holding Register)

[08] : 2 x jumlah data (2 x 4)

[00] [07] : nilai data di register pertama yang di request

[00] [06] : nilai data di register kedua yang di request

[00] [05] : nilai data di register ketiga yang di request

[00] [04] : nilai data di register keempat yang di request

[CRC1] [CRC2]: Hasil CRC nya

Contoh berikutnya, dimisalkan master meminta slave dengan ID/ address 07 untuk membaca register *input* dari alamat 30301 sampai 30303, dengan total 3 register. Maka bentuk pesannya adalah sbb:

[07] [04] [01] [2C] [00][03] [CRC1] [CRC2]

Start	[Address]	[Function Code]	[Reg][Reg]	[Lenght][Lenght]	[CRC1][CRC2]	End
3.5 caracter	8 Bits	8 Bits	2 x 8 Bits	2 x 8 Bits	16 Bits	3.5 caracter
3.5 caracter	[07]	[04]	[01] [2C]	[00] [03]	[CR1][CR2]	3.5 caracter

Gambar 9, frame data pada query membaca register input

[07] : karena slave ID/address nya adalah 07 (07 = 07 hex)

[04] : *function code* (kode fungsi membaca register *input*)

[01] [2C] :alamat *Register input* pertama dibaca (alamat 30301 - 30001 = 300 = 12C hex).

[00] [03] : jumlah alamat *Register input* yang dibaca (alamat 30301 sampai 30303 = 3 = 03 hex)

[CRC1] [CRC2] : Hasil CRC

Slave akan memberikan respon terhadap query dari master tersebut, dimisalkan register 30301 bernilai 1000 (3E8 hex), 30302 bernilai 275 (1F4 hex), dan 30303 bernilai 10 (0A hex). maka bentuk pesan dari respon slave tersebut adalah:

[07] [04] [06] [03] [E8][01] [F4][00] [0A] [CRC1] [CRC2]

Start	[Address]	[Function Code]	[2*Panjang data]	[data]	[data]	[data]	[CRC1][CRC2]	End
3.5 caracter	8 Bits	8 Bits	8 Bits	2 x 8 Bits	2 x 8 Bits	2 x 8 Bits	16 Bits	3.5 caracter
3.5 caracter	[07]	[04]	[06]	[03] [E8]	[01] [F4]	[00] [0A]	[CR1][CR2]	3.5 caracter

Gambar 10. frame data pada response membaca register input

Penjelasan:

[07] : Slave ID/address 07

[04] : Function (Read Input Register)

[06] : 2 x jumlah data (2 x 3)

[03] [E8] : nilai data di register pertama yang di request

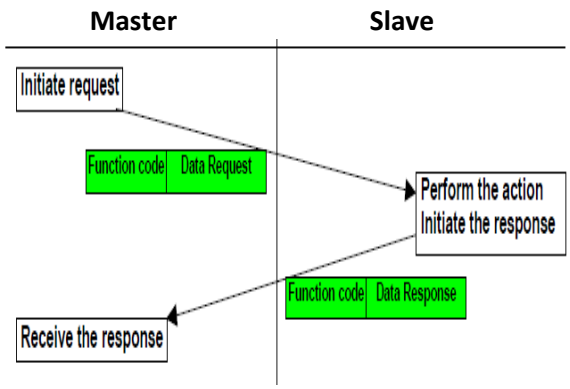
[01] [F4] : nilai data di register kedua yang di request

[00] [0A] : nilai data di register ketiga yang di request

[CRC1] [CRC2]: Hasil CRC nya

Pada contoh diatas menunjukkan bentuk frame data pada saat master meminta query dan slave memberikan respone terhadap query tersebut. Namum bagaimana bentuk sistem komunikasinya pada saat query response tersebut berlangsung? Gambar di bawah memberikan gambaran bagaimana dua sistem yang berbeda (Master & Slave) berkomunikasi menggunakan Modbus RTU, misal pada PLC A dan B dengan brand yang berbeda. Master (sistem utama, PLC A) melakukan query ke slave (PLC B) untuk mengirimkan data dengan parameter tertentu (slave ID, function code, data size, address, dll). Setelah query diterima, PLC slave akan merespon dengan mengirimkan reply berupa data yang diinginkan oleh Master.

Pada saat awal komunikasi, master mengirimkan query berupa initiate request yang dilengkapi dengan function code dan data request (dalam bentuk format data query seperti contoh diatas). Apabila query tersebut telah diterima oleh slave, format tersebut akan dibaca dan dilaksanakan oleh slave. Apabila telah selesai dan tidak ada error, slave akan mengirimkan initiate response dilengkapi dengan function code dan data response (dalam bentuk format data response seperti contoh diatas). Selanjutnya, master akan menerima response yang dikirimkan oleh slave tersebut.



Gambar 11. Komunikasi query/response pada modbus

H. Penutup

Protokol modbus merupakan salah satu sistem komunikasi yang banyak digunakan pada sistem kontrol di industri dan menjadi standar komunikasi antar PLC, RTU, SCADA dll. Modbus RTU merupakan aplikasi yang paling banyak digunakan dibanding modbus yang lain. Modbus RTU merupakan komunikasi serial dan menggunakan representasi nilai data biner.

Pada saat query/response, setiap protokol modbus mempunyai format data yang berbeda-beda dalam bentuk frame.

Pada modbus RTU, setiap frame baik dari master maupun slave selalu dimulai dengan interval minimal 3,5 kali karakter, dan diakhiri dengan interval yang sama minimal 3,5 kali karakter. Dan mekanisme untuk pemeriksaan kesalahan (error-check) digunakan CRC checksum.

Daftar Pustaka :

Huda Miftahul, *ProtokolKomunikasi Modbus RTU pada Sistem Otomasi Industri*, LPP Kampus Yogjakarta

Minamitsumori, Nishinari-ku, *Modbus Protocol Reference Guide*, *EM-5650 Rev.10*, M-System Co., LTD.

http://www.modbus.org/MODBUS Application Protocol Specification V1.1.pdf

http://elektroindonesia.com/elektro/instrum11.html

http://cangkruk.com/index.php?option=com_content&view=article&id=186&Itemid=243

http://munzir888.wordpress.com/2009/05/31/sepintas-tentang-modbus/

http://www.ari-sty.cz.cc/2011/04/latar-belakang-alur-data-aplikasi.html

http://www.simplymodbus.ca/FAQ.htm

http://www.rifqion.com/menulis/Modbus Protokol dan Serial Standard.htm

*) *Penulis adalah CPNS 2010 Pusdiklat Migas Cepu.*