****

**UNIVERSITATEA TEHNICĂ “GH. ASACHI” IAŞI**

**FACULTATEA AUTOMATICĂ ŞI CALCULATOARE**

**SPECIALIZAREA CALCULATOARE ŞI TEHNOLOGIA INFORMAŢIEI**

**DISCIPLINA**  **REȚELE DE CALCULATOARE PROIECT**

DOCUMENTAȚIE

APLICAȚIE PENTRU MONITORIZAREA RESURSELOR SO

(Client Modbus TCP)

**Coordonator,**

**Ș.l.dr.ing. Nicolae BOTEZATU**

**Studenți**

**George-Laurențiu APETRIA – 1305A**

**Alexandru ZBEREANU – 1305B**

**Gabriel BĂLĂNESCU – 1307A**

**Iaşi, 2020**

1. **Modbus TCP/IP – Introducere**

Modbus este un protocol încorporat pentru sistemele de automatizare industrială și pentru controller-ele programabile Modicon. Acesta a fost dezvoltat în anul 1979 și de atunci a devenit o metodă standard în industrie pentru transferul informațiilor digitale/analogice și stocarea datelor între controlul industrial și monitorizarea dispozitivelor. În prezent, Modbus este universal-acceptat, deschis, un protocol domeniu-public, dar care necesită o licență și care exclude obligativitatea plății unei ,,taxe de autor”.

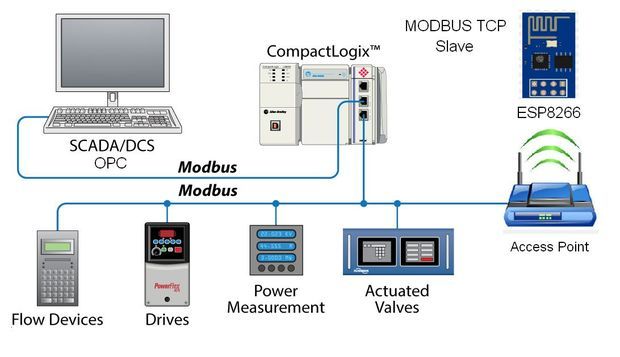


Fig. 1 **Ansamblu de dispozitive monitorizate prin intermediul protocolului Modbus**

1. **Modbus TCP/IP – Comunicarea între dispozitive**

În cadrul protocolului Modbus, dispozitivele comunică utilizând tehnica master-slave (client-server) în care doar un dispozitiv (the master/clientul) poate iniția o tranzacție numită cerere/solicitare. Celelalte dispozitive (slaves/server-ele) răspund prin livrarea datelor solicitate către master sau prin plasarea cererii într-o coadă.

Un **slave** este un dispozitiv periferic (traductor I/O, valvă, circuit de rețea sau alt dispozitiv de măsură, etc. ) care procesează informația și trimite output-ul său către master utilizând Modbus. De exemplu, modulele Acromag formează dispozitivele slave/server, în timp ce un dispozitiv master tipic este un calculator gazdă care rulează software-ul adecvat. Alte dispozitive pot funcționa atât ca masters/clienți, cât și ca slaves/servere.

**Clienții** (masters) pot adresa individual server-ele sau pot iniția mesaje de difuzie către toate server-ele. Aceste servere returnează răspunsuri pentru toate cererile adresate individual, dar nu răspund cererilor de difuzie (broadcast). Mai mult, serverele nu inițiază mesaje din proprie inițiativă, ci doar răspund cererilor master-ului.

*O* ***interogare*** *a unui* ***master*** *va consta din:*

* adresa unui slave (sau o adresă de difuzie);
* codul funcției care definește acțiunea solicitată;
* datele necesare;
* un câmp de verificare a erorii.

***Răspunsul*** *unui* ***slave*** *constă din:*

* câmpurile care confirmă acțiunea;
* datele care trebuie returnate;
* un câmp de verificare a erorii.



**Atenție!** Atât interogarea, cât și răspunsul trebuie să includă: adresa dispozitivului, codul funcției, datele aplicabile, dar și un câmp de verificare a eventualelor erori.

În cazul în care nu apare nicio eroare, răspunsul server-ului conține datele care au fost cerute. Dacă apare o eroare în cererea primită sau dacă slave-ul nu este disponibil pentru prelucrarea unei cereri adresate, acesta va returna un mesaj de excepție ca răspuns (vezi Modbus Exceptions).

Cadrul câmpului de verificare a erorii (error check field) al slave-ului îi permite master-ului să confirme dacă respectivul conținut al mesajului este valid. Mesajele Modbus obținute sunt transmise serial și verificarea parității este, de asemenea aplicată pentru fiecare caracter transmis în propriul cadru de date.

În acest moment, este important să facem distincția între faptul că Modbus însuși este un *protocol aplicație* care definește reguli de organizare și interpretare a datelor, și aspectul că acesta rămâne totuși o simplă structură de mesagerie, independentă de substratul nivelului fizic. În plus, acesta este ușor de înțeles, disponibil, gratuit și accesibil pentru oricine; este astfel furnizat și i se oferă suport pe scară largă de către o serie de producători.

1. **Ce este Modbus TCP/IP?**

Modbus TCP/IP (sau simplu Modbus-TCP) este pur și simplu protocolul Modbus RTU cu o interfață TCP care rulează pe Ethernet.

Structura de mesagerie Modbus este *protocolul aplicație* care definește regulile pentru organizarea și interpretarea datelor în mod independent de mediul de transmisie al datelor.

TCP/IP face referire la Protocolul de Control al Transmisiei și la Protocolul Internet care furnizează mediul de transmisie pentru mesageria Modbus TCP/IP.

Cu alte cuvinte, TCP/IP permite schimbul de blocuri de date binare între computere. Este, de asemenea, un standard recunoscut la nivel mondial care servește drept fundament pentru World Wide Web. Principala funcție pentru TCP este de a se asigura că toate pachetele de date sunt recepționate corect, în timp ce IP se asigără că mesajele sunt corect adresate și direcționate/rutate. Combinația TCP/IP reprezintă pur și simplu un protocol de transport și nu definește datele sau modul în care acestea trebuie interpretate (acesta este rolul *protocolului aplicație*, în cazul nostru Modbus).

Rezumând, protocolul Modbus TCP/IP folosește TCP/IP și Ethernet pentru a transporta datele în cadrul structurii de mesaje Modbus între dispozitivele compatibile. Astfel, Modbus TCP/IP combină rețeaua fizică (Ethernet) cu un standard de rețea (TCP/IP) și o metodă standard de reprezentare a datelor (Modbus ca *protocol aplicație*). Esențial este faptul că, un mesaj Modbus TCP/IP este o comunicație Modbus încapsulată într-un wrapper (,,înfășurător”) Ethernet TCP/IP.

În practică, protocolul Modbus TCP încorporează un cadru de date standard Modbus într-un cadru TCP, fără sumă de control Modbus, după cum se prezintă în schema următoare:

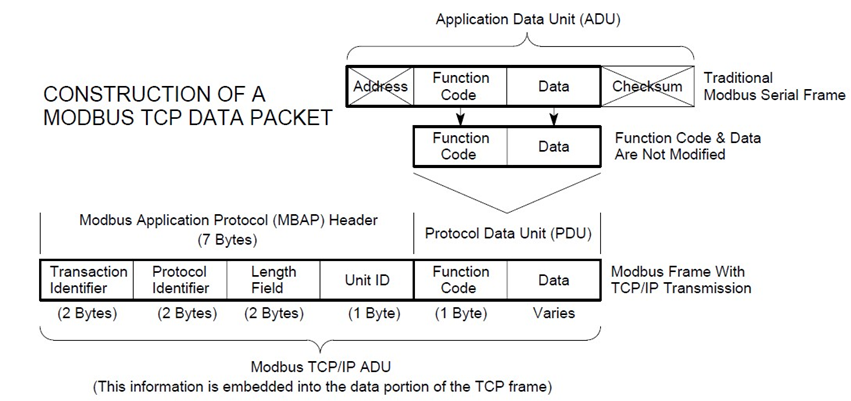


Fig. 2 **Structura unui pachet de date Modbus**

Atât comenzile Modbus, cât și datele-utilizator sunt încapsulate în ,,containerul de date” al unui mesaj TCP/IP fără a fi modificate în niciun fel. Cu toate acestea, câmpul de verificare a erorilor (checksum) nu este utilizat; în cazul Ethernet TCP/IP, metodele standard de checksum pentru *nivelul legătură* sunt utilizate în scopul garantării integrității datelor. Mai mult, cadrul câmpului de adresă Modbus este înlocuit de identificatorul unității în Modbus TCP/IP și devine parte integrantă a antetului Modbus Application Protocol (MBAP).

Din fig. 2 se observă faptul că atât codul funcției, cât și câmpurile de date sunt integrate în structura cadrului Modbus cu transmisie TCP/IP în formatul lor original. Astfel, Modbus TCP/IP **Application Data Unit (ADU)** ia forma unui **antet de 7 octeți** compus din:

* Identificator de tranzacție (Transaction Identifier);
* Identificator de protocol (Protocol Identifier);
* Câmp de lungime (Length Field);
* Identificator de unitate (Unit Identifier);

**Unitatea de date a protocolului (Protocol Data Unit - PDU)**, formată din:

* Codul funcției (Function Code);
* Date (Data);

Antetul MBAP are o lungime de 7 octeți și conține următoarele câmpuri:

1. **Identificator de tranzacție/invocare (Transaction/invocation Identifier - 2 octeți)**: acest câmp este utilizat pentru potrivirea/asocierea tranzacțiilor (transaction pairing) atunci când sunt mesaje multiple trimise în cadrul aceleiași conexiuni TCP de către un client fără a aștepta un răspuns anterior.
2. **Identificator de protocol (Protocol Identifier – 2 octeți)**: acest câmp are întotdeauna valoarea 0 pentru serviciile Modbus, iar celelalte valori sunt rezervate pentru posibilele extensii ulterioare.
3. **Câmpul de lungime (Length – 2 octeți)**: acest câmp reprezintă numărul de octeți din câmpurile rămase și include octetul identificatoului de unitate(unit identifier byte), octetul codului funcției și câmpurile de date.
4. **Identificatorul de unitate (Unit Identifier – 1 octet)**: acest câmp este utilizat pentru a identifica un server remote localizat într-o rețea non TCP/IP (pentru conectarea serială). Într-o aplicație clasică server Modbus TCP/IP, ID-ul unității este setat la 00 sau FF, fiind ignorat de server și returnat simplu în răspuns.

Application Data Unit (ADU) a protocolului Modbus TCP/IP este încorporat în manieră unitară în câmpul de date al unui cadru TCP standard și trimis prin intermediul TCP la binecunoscutul port de sistem 502, care este rezervat în mod special aplicațiilor Modbus. Clienții și serverele Modbus TCP/IP ascultă (listen) și primesc date Modbus prin portul 502.

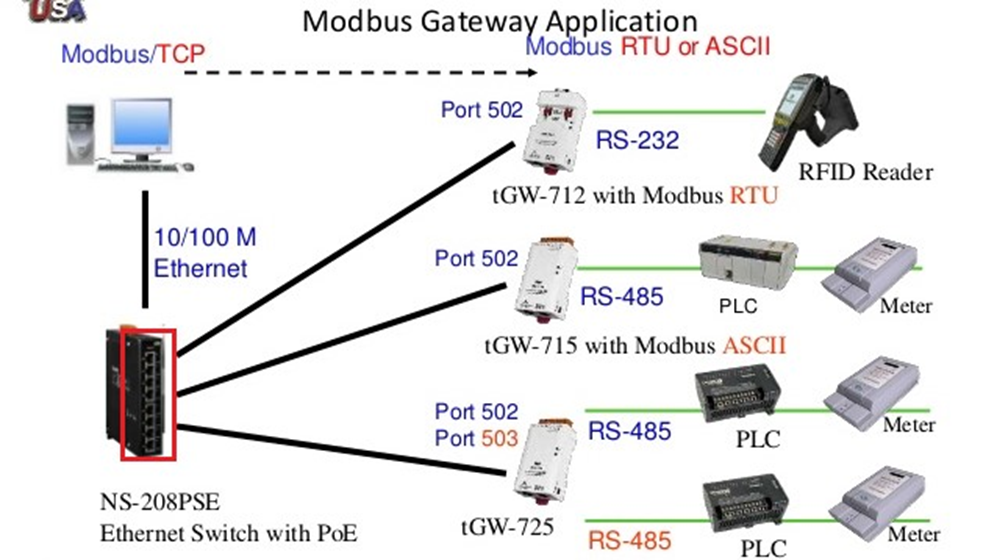


Fig. 3 **Exemplificare port TCP/IP 502**

Se poate observa că modul de funcționare al protocolului Modbus în raport cu Ethernet este aproape transparent în ceea ce privește structura registrelor/comenzilor Modbus. Astfel, o familiarizare cu funcționarea clasică a protocolului TCP poate determina automat o ușurință în înțelegerea funcționării ModbusTCP/IP.

1. **Fuziunea Modbus-Ethernet**

IEEE 802.3 Ethernet este un protocol de rețea office (office networking protocol) de lungă durată universal acceptat. De asemenea, este un standard deschis, susținut de o serie de producători, iar infrastructura sa este disponibilă și chiar instalată pe scară largă. În consecință, la nivel mondial se utilizează suita sa de protocoale TCP/IP și are rolul de bază în accesul în întreaga lume web. Deoarece multe dispozitive acceptă deja Ethernet, este normal ca utilizarea sa să fie extinsă în aplicațiile industriale.

La fel ca în cazul Ethernet, Modbus este disponibil gratuit, accesibil oricui și susținut pe scară largă de mulți producători de echipamente industriale. În plus, este ușor de înțeles și un ,,candidat” potrivit pentru utilizarea la construirea altor standarde de comunicare industriale. Cu atât de multe elemente comune, asocierea *protocolului aplicație* Modbus cu clasica transmisie IEEE 802.3 Ethernet reprezintă un puternic standard de comunicații industriale în Modbus TCP/IP. Deoarece Modbus TCP/IP are același *nivel fizic* și același *nivel legătură de date* cu tradiționalul IEEE 802.3 Ethernet și utilizează aceeași suită de protocoale TCP/IP, acesta rămâne pe deplin compatibil cu infrastructura Ethernet deja instalată (cabluri, conectori, interfața de rețea, carduri, hub-uri și switch-uri).

1. **Determinism**

În contextul actual, determinismul este un termen utilizat pentru a descrie capacitatea protocolului de comunicație de a garanta că un mesaj este trimis sau primit într-un interval finit de timp. Acest aspect al determinismului este foarte important în sensul controlului critic al aplicațiilor.

Din punct de vedere istoric, Ethernet-ul tradițional nu a fost considerat un bus (fieldbus) viabil pentru controlul industrial și rețele I/O din cauza a două neajunsuri majore:

* non-determinism inerent;
* durabilitate redusă.

Cu toate acestea, aplicarea corectă a noii tehnologii a rezolvat în mare parte aceste probleme.

Inițial, echipamentele Ethernet au fost concepute pentru spațiile de birouri, nu pentru condițiile dure din mediul industrial. Deși multe instalații industriale Ethernet pot utiliza acest standard hardware fără a întâmpina probleme, noii conectori cu utilizare industrială, cablurile ecranate și switch-urile și hub-urile întărite (hardened) sunt acum disponibile pentru a ajuta la rezolvarea problemei de durabilitate.

În ceea ce privește comportamentul nedeterminist al Ethernet, acesta este în mare parte rezultatul protocolului de arbitraj pe care îl folosește pentru accesul la operatorul de transmisie în rețea (carrier transmission access) – Carrier Sense Multiple Access with Collision Detect (CSMA/CD). Întrucât orice dispozitiv de rețea poate încerca să trimită un cadru de date oricând, cu CSMA/CD activat (applied), fiecare dispozitiv va detecta mai întâi dacă linia este inactivă (idle) și disponibilă pentru utilizare. Dacă linia este disponibilă, dispozitiul va începe atunci să transmită primul său cadru. Dacă un alt dispozitiv încearcă, de asemenea, să trimită un cadru aproape în același moment, atunci se va produce o coliziune și ambele cadre vor fi înlăturate (discarded). Fiecare dispozitiv așteaptă atunci o perioadă random de timp și încearcă din nou transmisia până când respectivul cadrul este trimis cu succes. Această metodă de alocare a canalului (channel-allocation) este inerent nedeterministă deoarece un dispozitiv poate transmite numai când linia (wire) este liberă, rezultând astfel timpi de așteptare impredictibili înainte ca datele să poată fi transmise. În plus, din cauza întârzierii de transmisie a semnalului prin cablu, coliziunile sunt încă posibile odată ce dispozitivul începe să transmită datele, forțând astfel cicluri suplimentare de retransmisie/reîncercare.

Majoritatea sistemelor de control au o necesitate (requirement) de timp definită pentru transmiterea pachetelor, de regulă mai mică de 100ms. Astfel, din cauza potențialului de coliziune și a metodei de retransmisie CSMA/CD, s-a considerat că tradiționalul Ethernet are un comportament nedeterminist, motiv pentru care au apărut o serie de probleme legate de acceptarea sa în scopul utilizării în aplicațiile de control critic. Cu toate acestea, CSMA/CD este în mod normal interzis/ascuns (suppressed) într-o rețea de dispozitive care sunt interconectate prin intermediul switch-urilor Ethernet. Mai exact, un dispozitiv pentru fiecare port de comutare. Prin această construcție/alcătuire (makeup) se face cel mai adesea referire la Ethernet-ul comutat în încercarea de a se distinge de comportamentul nedeterminist al Ethernet-ului tradițional.

Ethernet dobândește un caracter mai determinist prin utilizarea switch-urilor rapide Ethernet utilizate pentru interconectarea dispozitivelor. Aceste comutatoare/switch-uri măresc lățimea de bandă a rețelelor mari prin divizarea lor într-o multitudine de rețele de dimensiuni mai reduse sau prin separarea ,,domeniilor de coliziune”. Switch-ul, de asemenea, minimizează comunicația în rețea, facilitând o conexiune directă de la un transmițător spre un receptor în așa fel încât numai un anumit receptor să primească datele, nu întreaga rețea.

Așadar, cum funcționează un switch (sau hub de comutare) pentru a crește determinismul? Fiecare port al unui switch redirecționează datele către un alt port bazându-se pe adresa MAC ce se află în pachetul/cadrul de date primit. Switch-ul de fapt învață și stochează adresele MAC ale fiecărui dispozitiv cu care este conectat, împreună cu numărul portului asociat. Acum, portul comutatorului nu necesită propria adresă MAC și în timpul retransmisiei unui pachet recepționat, portul de comutare va arăta în schimb ca dispozitivul de origine, dat fiind faptul că și-a însușit (assumed) adresa sursei. În acest fel, domeniul de coliziune Ethernet se consideră că se termină la portul comutatorului, iar

switch-ul împarte efectiv rețeaua în legături de date distincte sau domenii de coliziune, câte unul pentru fiecare port de comutare. Abilitatea switch-ului de a asigna un pachet către un anumit port, mai degrabă decât redirecționarea acestuia către toate porturile de comutare, contribuie, de asemenea, la eliminarea coliziunilor care impun Ethernet-ului un caracter non-determinist.

Deci, pe măsură ce comutatoarele au devenit mai ieftine, tendința actuală în aplicațiile de nivel critic pentru controlul industrial este conectarea unui dispozitiv Ethernet per port de comutare, tratând în mod eficient dispozitivul de comutare ca hub-ul unei rețele de tip stea.

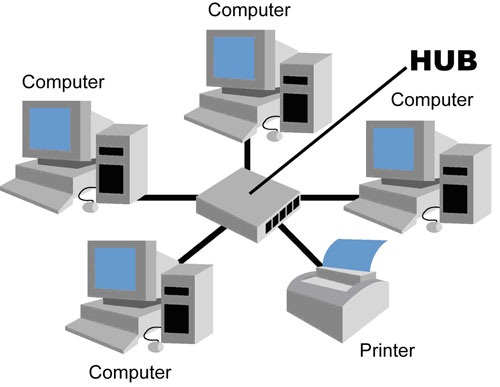
****

Fig. 4 **Rețea de tip stea**

Dat fiind faptul că la un port este conectat un singur dispozitiv, nu există șansa apariției coliziunilor. Acest lucru suprimă în mod eficient rutina CSMA/CD. În acest mod, cu un singur dispozitiv de rețea conectat per port-comutator (switch port), switch-ul poate lucra în modul full-duplex (sistemul de comunicații full-duplex este un sistem punct-la-punct compus din două sau mai multe părți sau dispozitive conectate care pot comunica între ele în ambele direcții), fără șanse de coliziune. Astfel, un switch Ethernet 10/100 rulează efectiv la 20/200 Mbps, deoarece poate transmite și recepționa simultan 10/100 Mbps în ambele direcții (full-duplex). Viteza de transfer mai mare a full-duplex-ului cuplat/activat (coupled) fără necesitatea invocării CSMA/CD produce un mod mai determinist de operare, ajutând aplicațiile de control critic să rămână previzibile și on-time.

Din păcate, traficul de broadcast pe o rețea dintr-o companie nu poate fi complet filtrat cu ajutorul switch-urilor, iar acest lucru poate provoca coliziuni suplimentare, reducând astfel determinismul unei rețele care conectează mai multe dispositive la un port de comutare. Cu toate acestea, dacă rețeaua companiei și rețeaua de control & I/O sunt în schimb separate, nu se adaugă trafic la rețeaua de control și determinismul este sporit. Mai mult, dacă se utilizează o punte de legătură (bridge) pentru a separa cele două conexiuni, atunci această legătură (bridge) poate fi configurată pentru a filtra traficul inutil.

Astfel, putem vedea cum asocierea unui bun design de rețea cu switch-uri rapide și punți de legătură (bridges), acolo unde sunt necesare, determină creșterea nivelului determinismului unei rețele, făcând Ethernetul mai ofertant. Alte progrese înregistrate în ceea ce privește switch-urile Ethernet, cum ar fi, vitezele mai mari, protecția împotriva furtunilor de broadcast (broadcast storm), suport pentru VLAN, SNMP (Simple Network Management Protocol) și mesageria prioritară ajută în continuare la creșterea determinismului rețelelor Ethernet. Pe măsură ce Gigabit (Gbit), 10 Gbit și 100 Gbit Ethernet intră pe piață, determinismul devine o preocupare cu un interes tot mai redus.

Pentru a înțelege mai bine cum este structurat Modbus TCP/IP și care este sensul cuvintelor ,,open standard”, trebuie să revedem modelul de referință Open Systems Interconnect (OSI). Acest model a fost dezvoltat de International Standards Organization și adoptat în anul 1983 ca o referință comună pentru dezvoltarea standardelor de comunicare a datelor, cum este și Modbus TCP/IP. Menirea sa nu este de a defini o implementare reală, ci mai degrabă are rolul de ajutor/sprijin (aide) structural pentru a înțelege ,,ce trebuie făcut” (,,what must be done”) și ,,ce și unde merge” (,,what goes where”).

1. **Modelul de rețea OSI**

Modelul tradițional OSI este prezentat mai jos, împreună cu modelul simplificat 5-layer al standardului TCP/IP (layer-ele 5 și 6 au fost eliminate). În cadrul modelului OSI, funcțiile de comunicare sunt divizate în 7 (5) layere, fiecare dintre acestea având task-uri bine definite. De exemplu, nivelul 1 al acestui model este nivelul fizic și definește caracteristicile de transmitere fizică. Cel de-al 2-lea layer este nivelul legătură de date și definește protocolul de acces al magistralei. Nivelul 7 este nivelul aplicație și definește funcțiile aplicației (acesta este nivelul care definește modul de interpretare a datelor dispozitivului).

Modelul OSI reprezintă arhitectura de bază a rețelei. Fiecare nivel al acestui model folosește serviciile furnizate de nivelul situat imediat sub el. TCP/IP nu are mapări specifice pentru nivelurile 5 și 6 ale acestui model (OSI), iar aceste niveluri sunt adesea omise atunci când se face referire la stiva TCP/IP.

|  |  |  |  |
| --- | --- | --- | --- |
| **Modelul OSI cu 7 niveluri** | | | **Standardul TCP/IP** |
| 7 | Aplicație | Folosit de aplicațiile software pentru pregătirea și interpretarea datelor ce urmează a fi utilizate de către celelalte 6 niveluri OSI de dedesupt. Oferă interfața aplicației către rețea. HTTP, FTP, email, SMTP ȘI POP3, CIPTM, SNMP se găsesc la acest nivel. | Nivelul aplicație |
| 6 | Prezentare | Reprezentarea datelor, tipul de codificare și definirea caracterelor folosite. Efectuează ,,negocieri” și conversii de date și protocol pentru a se asigura că schimbul de date se poate realiza între gazde și totodată datele pot fi transportate de-a lungul rețelei. De asemenea, efectuează compresia și criptarea datelor. |
| 5 | Sesiune | Controlul apelului și sincronizarea sesiunii de conexiune. Responsabil pentru stabilirea și gestionarea sesiunilor/conexiunilor între aplicații și rețea. Windows WinSock socket API este un manager obișnuit pentru nivelul sesiune. |
| 4 | Transport | Secvențierea datelor aplicațiilor, controlează începutul/sfârșitul transmisiei, furnizează detecția erorilor, corectarea acestora, recuperarea end-to-end și deblocarea. Oferă fluxul software al controlului datelor între rețele. TCP și UDP se găgesc aici. | Nivelul transport |
| 3 | Rețea | Controlează rutarea, prioritizarea, configurarea, eliberarea conexiunilor, fluxul (de date). Stabilește/menține conexiunile într-o rețea și furnizează adresarea, rutarea și ditribuția pachetelor către gazde. IP, PPP, IPX și X.25 se găsesc aici. | Nivelul Internet, Rețea sau Internetwork |
| 2 | Legătură de date | Responsabil pentru asigurarea fiabilității distribuției către nivelurile inferioare, incluzând cadrul de date, detectarea erorilor și corecția acestora, controlul secvenței și controlul fluxului. La acest nivel sunt definite Ethernet (IEEE 802.2) și MAC. | Nivelul acces la rețea  sau  Nivelul Host-to-Network |
| 1 | Fizic | Definește atributele/elementele electrice, mecanice, funcționale și procedurale folosite pentru a accesa și pentru a trimite un flux de date binare pe un mediu fizic (definește conectorul RJ-45 și cablul CAT5 al Ethernet). |

Tabelul nr. 1

Prin intermediul modelului OSI, putem deduce că pentru a avea două dispozitive interoperabile în aceeași rețea, acestea trebuie să aibă același *nivel aplicație* în cadrul protocolului. În trecut, multe dispozitive de rețea și-au folosit propriile lor protocoale proprietare și acest lucru le-a împiedicat interoperabilitatea. Prin urmare, a apărut necesitatea adoptării unor soluții I/O de rețea deschisă care ar permite dispozitivelor provenite de la o varietate de furnizori să lucreze perfect împreună și această unitate de interoperabilitate a fost un motiv cheie pentru care a fost creat Modbus TCP/IP.

În modelul TCP/IP, Ethernet gestionează cele 2 niveluri de bază (1&2) din cele 7 niveluri ale stivei OSI, în timp ce TCP/IP se ocupă de următoarele 2 niveluri (3&4). *Nivelul aplicație* se află deasupra TCP, IP, respectiv Ethernet și reprezintă layer-ul cu informații care dau sens datelor transmise.

Spre exemplu, în cazul modulelor Acromag 9xxEN-40xx Modbus TCP/IP, nivelul aplicație este reprezentat de Modbus. Astfel, Modbus TCP/IP folosește mediile Ethenet și TCP/IP pentru a comunica, utilizând un *nivel aplicație* cu aceeași metodă de accesare a registrului folosită de Modbus RTU. Deoarece mulți producători creează suport atât pentru Modbus RTU, cât și pentru TCP/IP și pentru că Modbus este înțeles și distribuit gratuit pe scară largă, Modbus TCP/IP este perceput, de asemenea, ca un standard deschis.

Observăm astfel că Modbus TCP/IP se bazează pe familia de protocoale TCP/IP și cuprinde aceleași 4 niveluri inferioare ale modelului OSI care sunt comune tuturor dispozitivelor Ethernet. Acest lucru îl face pe deplin compatibil cu hardware-ul Ethernet existent, cum ar fi: cablurile, conectorii, plăcile de interfață de rețea, hub-urile și switch-urile.

TCP/IP se referă la Protocolul de Control al Transmisiei și Protocolul Internet care au fost utilizate pentru prima dată în 1974. TCP/IP este fundamentul pentru World Wide Web și formează protocolul cu *nivelurile transport și rețea* ale internetului care, de obicei, relizează legăturile între toate instalațiile Ethernet la nivel mondial. Cu alte cuvinte, TCP/IP permite schimbul de blocuri de date binare între computere. Principala funcție pentru TCP este de a se asigura că toate pachetele de date sunt recepționate corect, în timp ce IP se asigără că mesajele sunt corect adresate și direcționate/rutate. Combinația TCP/IP reprezintă pur și simplu un protocol de transport și nu definește datele sau modul în care acestea trebuie interpretate (acesta este rolul protocolului aplicație, în cazul nostru Modbus).

În contrast cu aceste lucruri, Modbus este un *protocol aplicație*. Acesta definește regulile de organizare și interpretare ale datelor și este în esență o structură de mesagerie independentă de *nivelul fizic* subadiacent. Este disponibil gratuit și accesibil oricui, ușor de înțeles și susținut pe scară largă de mulți producători.

Protocolul Modbus TCP/IP folosește TCP/IP și Ethernet pentru a transporta datele în cadrul structurii de mesaje Modbus între dispozitive. Astfel, Modbus TCP/IP combină rețeaua fizică (Ethernet) cu un standard de rețea (TCP/IP) și o metodă standard de reprezentare a datelor (Modbus).

1. **Stiva TCP/IP**

TCP/IP este de fapt format dintr-o ,,suită” de protocoale pe care se bazează totalitatea comunicațiilor din internet. Această suită de protocoale este, de asemenea, denumită stivă de protocoale. Fiecare host sau router de pe internet trebuie să ruleze o stivă de protocoale. Utilizarea cuvântului *stivă* face referire la Modelul de Referință simplificat al ierarhiei TCP/IP sau ,,stiva” este utilizată pentru proiectarea software-ului de rețea și este prezentată după cum urmează:

|  |  |  |
| --- | --- | --- |
| 5 | Aplicație | Specifică în ce mod o aplicație utilizează o rețea. |
| 4 | Transport | Specifică modul de asigurare al transportului fiabil al datelor. |
| 3 | Internet/Rețea | Specifică formatul și rutarea pachetelor. |
| 2 | Host-to-Network | Specifică organizarea și transmiterea cadrelor. |
| 1 | Fizic | Specifică hardware-ul de rețea de bază. |

Tabelul nr. 2

Pentru a înțelege mai bine funcționarea stivei, următorul tabel ilustrează fluxul de date de la un sender la un receiver, utilizând stiva TCP/IP (*nivelul Host-to-Network* a fost redenumit astfel, *nivelul Legătură de Date*):

|  |  |  |
| --- | --- | --- |
| **SENDER** | → Virtual Connection → | **RECEIVER** |
| ↓ Aplicație | ← Mesaj Echivalent → | Aplicație ↑ |
| ↓ Transport | ← Mesaj Echivalent → | Transport ↑ |
| ↓ Internet/Rețea | ← Mesaj Echivalent → | Internet/Rețea ↑ |
| ↓ Legătură de Date | ← Mesaj Echivalent → | Legătură de Date ↑ |
| → Hardware Fizic | →→→→→→→→→→→ | Hardware Fizic ↑ |

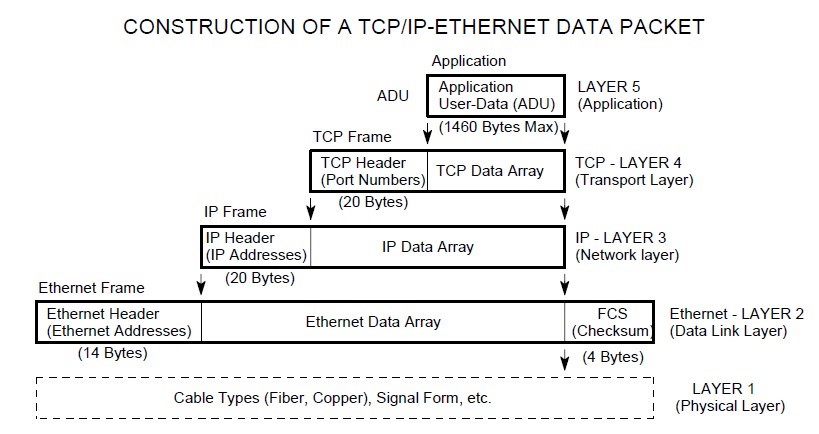
Tabelul nr. 3

Fiecare nivel al stivei de trimitere comunică cu nivelul corespunzător al stivei de recepție prin intermediul informațiilor stocate în anteturi. Pe măsură ce datele sunt mutate în partea inferioară a stivei sender-ului, fiecare nivel din stivă adaugă propriul său antet în partea din față a mesajului pe care îl primește de la nivelul superior. Astfel sunt încapsulate straturile superioare de către straturile inferioare. În momentul în care mesajul ajunge la receptor, aceste informații, din fiecare antet corespunzător, sunt eliminate. În acest fel, anteturile sunt eliminate pe măsură ce pachetul de date are o mișcare ascendentă în stiva aplicației de recepție.

|  |  |  |  |
| --- | --- | --- | --- |
| **STIVA DE COMUNICAȚIE TCP/IP** | | | |
| # | **MODEL** | Protocoale importante | Referințe |
| 7 | Aplicație | Modbus |  |
| 6 | Prezentare |  |
| 5 | Sesiune |  |
| 4 | Transport | TCP |  |
| 3 | Rețea | IP, ARP, RARP |  |
| 2 | Legătură de Date | Ethernet, CSMA/CD, MAC | IEEE 802.3  Ethernet |
| 1 | Fizic | Nivelul Fizic Ethernet |

Tabelul nr. 4

Fig. 5 ilustrează construcția unui pachet TCP/IP-Ethernet pentru transmisie. Pentru Modbus TCP/IP, nivelul aplicație este Modbus, iar Modbus Application Data Unit este încorporat într-un array de date TCP. Când o aplicație își transmite datele prin rețea, datele sunt transmise în jos prin intermediul fiecărui nivel. În figura de mai jos este reprezentat modul de împachetare (încapsulare) a informațiilor care provin de pe nivelul superior în octeții de date ai nivelurilor inferioare care urmează. Fiecare strat ulterior are o funcție desemnată (designated) și atașează propriul antet de protocol în partea din față a pachetului său. Nivelul de jos este responsabil de trimiterea efectivă a datelor. Această întreagă procedură de înfășurare este mai apoi efectuată în sens invers la recepționarea datelor (datele primite sunt despachetate/desășurate la fiecare nivel și sunt transmise ascendent către *nivelul aplicație* al receptorului).

Fig. 5 **Construcția unui pachet de date TCP/IP-Ethernet**

Pentru a ilustra, prin intermediul Modbus TCP/IP, aplicația gazdă (master/client) formează cererea, iar apoi transmite datele în jos, spre nivelurile inferioare, care la rândul lor își adaugă propriile informații de control în cadrul pachetului sub forma unor anteturi de protocol și uneori sub forma unor note de subsol. În cele din urmă, pachetul ajunge la nivelul fizic de unde va fi transmis electronic la destinație (slave/server). Atunci pachetul traversează diferite niveluri ale destinației (în ordine ascendentă); trecerea printr-un nivel reprezintă decodarea porțiunii corespunzătoare din mesaj și eliminarea antetului aferent, respectiv a subsolului, toate acestea fiind atașate de către straturile corespunzătoare ale computerului client expeditor. În final, pachetul ajunge la *nivelul aplicație* al destinației. Deși un strat poate comunica doar cu stratul imediat superior sau imediat inferior, întregul proces poate fi privit ca un singur nivel la un capăt care poate comunica cu nivelul de la celălalt capăt.

1. **Cuvinte cheie & Terminologie**

Pentru o bună înțelegere a protocolului Modbus TCP/IP și a funcționării operațiilor asupra stivei, reamintim următoarele cuvinte cheie, respectiv următorii termeni:

* Toate protocoalele de rețea sunt structurate ca un model stratificat.
* Există unul sau mai multe protocoale (entitate nivel) în cadrul fiecărui nivel.
* Entitățile omoloage (peer) se referă la 2 sau mai multe protocoale de pe același nivel (inclusiv protocoalele de pe același nivel, dar de pe noduri diferite).
* Regulile de funcționare între entitățile peer sunt denumite proceduri.
* Protocolul se referă la regulile de funcționare urmate de entitățile peer. Protocolul definește formatul PDU (Protocol Data Units) și regulile lor de funcționare.
* Această stratificare a entităților de protocol este denumită stivă de protocol.
* Nivelul n comunică cu alte entități de nivel n (alte protocoale de pe același layer), utilizând stratul n al Protocol Data Units (PDU’s).
* Nivelul n utilizează serviciile stratului n-1 și furnizează un serviciu pentru nivelul n+1.
* Interfața dintre un nivel și nivelul de deasupra acestuia este denumită ca Service Access Point (SAP). Datele interfeței dintre niveluri reprezintă Service Data Unit (SDU).
* Protocoalele pot fi orientate pe conexiuni sau connectionless. O conexiune implică următorul aspect: comunicația necesită sincronizarea tuturor părților înainte de a se produce transferul datelor de aplicație. Modbus TCP/IP este un protocol orientat pe conexiuni.
* Modbus TCP/IP urmează modelul Client-Server. În cadrul protocolului Modbus, master-ul este reprezentat de client, iar server-ele reprezintă entitățile slave.
* Un client (orice master) este orice dispozitiv de rețea care trimite cereri de date către servere (slaves).
* Un server (slave) este orice program care așteaptă ca spre el să fie emise cereri de date. Serverele nu inițiază cererile cu clienții, ci doar răspund la acestea. O serie de dispozitive funcționează atât în rolul de client și în cel de server.
* Un port este o adresă care este utilizată local la *nivelul transport* (pe un nod) și identifică sursa și destinația pachetului în interiorul aceluiași nod. Numerele de porturi sunt împărțite între binecunoscutele numere de port (0-1023), numere de port utilizator înregistrate (1024-49151) și numerele de port private/dinamice (49152-65535). Porturile permit protocolului TCP/IP să multiplexeze și să demultiplexeze o secvență de datagrame IP care trebuie să ajungă la multe alte (simultan) procese de aplicație. Modbus TCP/IP folosește binecunoscutul port 502 pentru a asculta (listen) și recepționa mesajele Modbus prin Ethernet.
* Un socket este o adresă a nivelului aplicație care este formată din combinația unei adrese IP și a unui număr de port (exprimat ca <Adresa IP a Gazdei>:<Număr Port>:<Nume Gazdă>:<Număr Port>) și este folosit ca adresă generală de identificare a unui proces de aplicație. Protocoalele de aplicație îl utilizează pentru a urmări numărul portului asignat către fiecare instanță a unei aplicații atunci când se utilizează TCP.

1. **Nivelul Aplicație**

Nivelul superior al modelelor de referință TCP/IP și OSI este *nivelul aplicație*. Există mai multe protocoale de nivel aplicație; dintre ele amintim: FTP, Telnet, HTTP, SMTP, DNS, NNTP ș.a. Deși fiecare dintre aceste protocoale are propriul scop specific, pentru Modbus TCP/IP, principalul protocol de interes de *nivel aplicație* este Modbus.

1. **Funcții și registre Modbus**

Suita de protocoale TCP/IP (stiva de protocoale independente) furnizează toate resursele necesare pentru ca 2 dispozitive să comunice între ele prin intermediul unui Ethernet LAN sau WAN. Dar TCP/IP garantează doar faptul că mesajele aplicației vor fi transferate între aceste dispozitive și NU garantează că aceste dispozitive vor înțelege efectiv mesajele sau vor interopera unul cu celălalt. Pentru Modbus TCP/IP, această funcție este asigurată de protocolul de *nivel aplicație* Modbus.

Modbus este un protocol de aplicație sau o structură de mesagerie care definește regulile pentru organizarea și interpretarea datelor în mod independent de mediul de transmisie al datelor. Clasicul Modbus serial este un protocol bazat pe registru care definește modul de tranzacționare a mesajelor care apar între clienți și servere. Dispozitivele slave ascultă (listen) în vederea comunicației inițiate de către master și doar răspund când acest lucru le este solicitat. Master-ul controlează întotdeauna comunicație și poate comunica direct cu un slave sau cu toate server-ele cu care este conectat, dar serverele nu pot comunica direct între ele.

Așadar, observăm că Modbus funcționează în mod obișnuit după principiul client/server (master/slave). Astfel, clientul (master-ul) trimite un mesaj de solicitare (service request) către server (slave), iar server-ul răspunde cu un mesaj de răspuns. În cazul în care server-ul nu poate să proceseze cererea, el va returna în schimb codul de eroare al unei funcții (exception response) care reprezintă codul funcției originale la care se adaugă 80H (de exemplu, cel mai semnificativ bit este setat pe 1).

Funcțiile Modbus operează pe registre de memorie pentru configurarea, monitorizarea și controlarea dispozitivelor de I/O. De obicei, dispozitivele Modbus includ o hartă a registrelor (Register Map). Este necesar să ne referim (să utilizăm) la harta de regiștri în cazul fiecărui dispozitiv folosit pentru o bună înțelegere a operațiunii efectuate de către acesta. De asemenea, un alt avantaj este reprezentat de interpretarea hărții regiștrilor ca punct de referință în momentul examinării funcțiilor Modbus (descrise ulterior în acest document).

Modelul de date Modbus are o structură simplă care se diferențiază doar între patru tipuri de date de bază:

1. Discrete Inputs
2. Coils (Outputs)
3. Input Registers (Input Data)
4. Holding Registers (Output Data)

Cererea de servire (Modbus Protocol Data Unit) este alcătuită din codul funcției și un număr de octeți de date suplimentare, acest lucru depinzând de tipul funcției. În majoritatea cazurilor, datele suplimentare sunt de obicei o referință variabilă, cum ar fi o adresă de registru, deoarece majoritatea funcțiilor Modbus funcționează pe registre.

Registrele Modbus ale unui dispozitiv sunt organizate pe baza celor patru date de referință de bază mai sus menționate, iar acest tip de dată va fi identificat în continuare prin numărul principal al adresei de referință după cum urmează:

|  |  |
| --- | --- |
| **Reference** | **Description** |
| 0xxxx | Read/Write Discrete Outputs or Coils. O adresă de referință 0x este utilizată pentru a transmite datele de ieșire spre un canal de ieșire digital. |
| 1xxxx | Read Discrete Inputs. Starea ON/OFF a unei adrese de referință 1x este controlată de canalul digital de input corespunzător. |
| 3xxxx | Read Input Registers. Un registru de referință 3x conține un număr de 16 biți primit de la o sursă externă, cum ar fi un semnal analogic. |
| 4xxxx | Read/Write Output or Holding Registers. Un registru 4x este utilizat pentru a stoca 16 biți de date numerice (în format binar sau zecimal), sau pentru a trimite datele de la CPU la un canal de ieșire. |

Tabelul nr. 5

Caracterele ,,x” care urmează după prima valoare reprezintă locația adresei formată din 4 biți în segmentul de date utilizator.

Primul caracter este în general furnizat de codul funcției și omis din specificatorul de adresă pentru o funcție dată. De asemenea, acest caracter identifică și tipul de dată I/O.

**IMPORTANT!** Adresele de referință notate în harta de memorie nu sunt adrese de memorie hard-codate explicit. Intern, toate dispozitivele Modbus utilizează un offset de memorie bazat pe 0 (zero-based), calculat de la adresa de referință. Cu toate acestea, interfața de sistem a sistemelor Modbus (software) va varia în această privință și va putea solicita introducerea adresei de referință efective, renunțarea la primul (principalul) număr sau introducerea unui offset (decalaj) de memorie de la 1 sau un offset al unei adrese de memorie de la 0. Acest lucru este dependent de sistem și reprezintă totodată principala sursă a erorilor de programare. Aveți grijă la acest aspect atunci când scrieți aplicații high-level pentru accesarea acestor registre.

NU toate funcțiile Modbus funcționează pe baza unui registru din harta de registre. Toate adresele de date din mesajele Modbus sunt referite (trimise) la 0, cu prima apariție a unui element de date adresat având numărul de articol 0. Mai mult, câmpul care conține codul funcției specifică deja pe ce grup de registre funcționează (spre exemplu: adresele de referință 0x, 1x, 3x sau 4x). De exemplu, holding register 40001 este adresat ca fiind registrul 0000 în câmpul de adresă de date al mesajului. Codul funcției care operează pe acest registru specifică o operație ,,holding register” și grupul de referință ,,4xxxx” este implicat. Astfel, holding register 40108 este de fapt adresat ca fiind registrul 006BH (107 în zecimal).

Câmpul codului funcției al mesajului (PDU) va conține un octet care îi indică slave-ului ce fel de acțiune trebuie să execute. Codurile valide pentru funcții sunt cuprinse între 1 și 255, dar nu toate codurile se vor aplica unui modul, iar unele coduri sunt rezevate pentru utilizări ulterioare. Mai mult, specificațiile Modbus realizează alocarea codurilor funcțiilor cuprinse între 67-72 și 100-110 pentru serviciile definite de utilizator.

Tabelul nr. 6 evidențiază un subset de funcții standard Modbus suportate (acceptate) de către modulele Acromag (adresele registrelor de referință aupra cărora acționează funcțiile sunt de asemenea indicate). Funcțiile de mai jos sunt utilizate pentru a accesa registrele prezentate în harta de registre a modulului pentru trimiterea și recepționarea datelor. Comanda Report Slave ID nu operează pe un registru din harta de registre.

|  |  |  |
| --- | --- | --- |
| **CODE** | **FUNCTION** | **REFERENCE** |
| 01 (01H) | Read Coil (Output) Status | 0xxxx |
| 03 (03H) | Read Holding Registers | 4xxxx |
| 04 (04H) | Read Input Registers | 3xxxx |
| 05 (05H) | Force Single Coil (Output) | 0xxxx |
| 06 (06H) | Preset Single Register | 4xxxx |
| 15 (0FH) | Force Multiple Coils (Outputs) | 0xxxx |
| 16 (10H) | Preset Multiple Registers | 4xxxx |
| 17 (11H) | Report Slave ID | Hidden |

Tabelul nr. 6

Câmpul de request de date al clientului furnizează slave-ul (server-ul) și câteva informații adiționale cerute de către server pentru a finaliza acțiunea specificată de codul funcției din cererea clientului. Câmpul de date include de obicei adrese de registre, valori de numărare (count values) și datele scrise (written data). Pentru unele mesaje este posibil ca acest câmp să nu existe (adică este de lungime 0), deoarece nu toate mesajele vor necesita date.

Când dispozitivul slave îi răspunde master-ului, acesta utilizează câmpul de cod al funcției pentru a indica fie un răspuns normal (fără erori), fie un răspuns de acest tip: ,,a apărut o eroare” (,,error has occured”), adică un mesaj de excepție. Un răspuns normal pur și simplu repetă (afișează din nou) codul funcției originale al interogării, în timp ce un răspuns de excepție returnează un cod care este echivalent cu codul funcției originale care are bit-ul cel mai semnificativ (MSB) setat pe 1 logic.

De exemplu, comanda Read Holding Registers are codul funcției 0000 0011 (03H). Dacă dispozitivul slave efectuează acțiunea solicitată fără eroare, returnează același cod în răspunsul său. Cu toate acestea, dacă apare o excepție, se returnează codul 1000 0011 (83H) în câmpul codului funcției și se adaugă un cod unic în câmpul de date al mesajului de răspuns care ,,îi spune” master-ului ce tip de eroare a apărut sau motivul excepției (vezi excepții Modbus).

Programul de aplicație client trebuie să gestioneze răspunsul de excepție. El poate să aleagă să posteze reîncercările urterioare ale mesajului original, poate să încerce să trimită o interogare de diagnoză sau poate simplu să notifice operatorul în privința erorii apărute.

Următoarele paragrafe descriu câteva dintre funcțiile Modbus, în mod obișnuit suportate de către modulele Acromag 9xxEN. Aceste exemple sunt descrise din perspectiva TCP/IP (ID-ul unității înlocuiește adresa salve-ului, iar clasicul câmp de verificare serială CRC/LRC este eliminat). Doar primul exemplu va include informațiile antetului MBAP. Pentru o descriere completă a funcțiilor Modbus, se recomandă verificarea (consultarea) specificațiilor Modbus. Pentru o mai bună înțelegere a protocolului Modbus, se recomandă consultarea hărții de registre a propriului modul concomitent cu parcurgerea acestui material.

În momentul examinării acestor exemple prin comparație cu clasicele comenzi seriale Modbus, trebuie să se țină cont de faptul că adresa slave-ului este înlocuită de identificatorul unității în Modbus TCP/IP (setat în mod normal la 00H sau FFH). În plus, câmpul de verificare a erorilor (CRC/LRC) este eliminat, deoarece TCP/IP aplică deja propriul mecanism de verificare a erorilor. Pentru comenzile care acceptă transmisie broadcast, acesta se aplică doar în cazul protocolului Modbus serial, deoarece Modbus TCP/IP este unicast (cu excepția cazului în care se utilizează un bridge Ethernet-to-serial).

Diferitele câmpuri ale ADU Modbus TCP/IP sunt codate în formatul Big-Endian. Asta înseamnă că cel mai semnificativ octet din secvență este stocat la cea mai mică adresă de stocare (adică este prima).

Următorul exemplu va include informațiile impuse de formatul antetului MBAP, dar informațiile header-ului nu vor fi repetate în exemplele următoare.

***Read Coil Status (01)***

Această comandă va citi statusul ON/OFF pentru **discrete outputs or coils** (adresa de referință 0x) din slave/server. Pentru modulele Acromag, răspunsul este echivalent cu citirea statusului ON/OFF pentru releele cu output solid-state sau switch-uri. Transmisia broadcast nu este suportată.

Interogarea Read Coil Status specifică punctul de începere al înfășurătorului (canalul de output) și cantitatea de date (coils) care trebuie citită (quantity of coils to be read). Coil-urile corespund releelor discrete solid-state ale acestui dispozitiv și sunt adresate începând de la (până la 4 coil-uri adresate între 0 și 3 la acest model). Read Coil Status din mesajul de răspuns este împachetat ca un coil sau canal pe bit din câmpul de date. Pentru modulele Acromag, statusul de output indică 1 pentru ON (,,conduce curentul”) și 0 pentru OFF (,,NU conduce curentul”). LSB-ul primului octet de date corespunde statusului coil-ului adresat în interogare. Celelalte coil-uri urmează secvențial, prin deplasare spre capătul de ordin superior al octetului. Din moment ce acest exemplu are doar 4 output-uri, biții rămași din octetul de date vor fi setați pe 0 în direcția capătului superior neutilizat al octetului.

|  |  |
| --- | --- |
| **MBAP Header Fields** | **Example Decimal (Hexadecimal)** |
| Transaction ID High Order | 0 (00) Client sets, unique value. |
| Transaction ID Low Order | 1 (01) Client sets, unique value. |
| Protocol Identifier High Order | 0 (00) Specifies Modbus service. |
| Protocol Identifier Low Order | 0 (00) Specifies Modbus service. |
| Length High Order | 0 (00) Client calculates. |
| Length Low Order | 6 (06) Client calculates. |
| Unit Identifier | 255 (FF) or 0 (00) Do not bridge. |

Tabelul nr. 7 – **Modbus Request ADU Example Header**

Identificatorul de tranzacție este utilizat pentru realizarea potrivirii dintre răspuns și interogare atunci când clientul trimite mai multe interogări, fără a aștepta un răspuns anterior. De regulă, acesta este un număr cuprins între 1 și 16, dar numărul maxim de tranzacții client și numărul maxim de tranzacții per server vor varia conform dispozitivului. Identificatorul de protocol este întotdeauna 0 pentru Modbus. Lungimea este reprezentată de o valoare compusă din numărul de octeți conținuți în date, codul funcției (un octet) și identificatorul unității (un octet).

Identificatorul unității este 00H sau FFH, deoarece acest modul este Modbus TCP/IP. Dacă acest modul a fost unul de tip Modbus serial clasic (fără port Ethernet) și a fost adresat prin intermediul unui bridge sau gateway de la un client Ethernet (un Ethernet-to-serial bridge), atunci identificatorul unității este echivalent cu adresa clasică a unui dispozitiv Modbus slave serial (1-247). Folosind 00H sau FFH după cum se prezintă aici, se va cauza, de către orice dispozitiv serial (bridge/gateway), blocarea trecerea mesajului client prin intermediului acelui bridge. Acesta este motivul pentru care un text afișează identificatorul unității ca parte integrantă a interogării, din moment ce înlocuiește clasica adresă a slave-ului (lungimea include octetul de idetificare a unității), în timp ce ceilalți sunt prezentați ca parte a antetului MBAP după cum se expune și aici.

|  |  |
| --- | --- |
| **Field Name** | **Example Decimal (Hexadecimal)** |
| Function Code | 1 (01) |
| Starting Adress High Order | 0 (00) |
| Starting Adress Low Order | 0 (00) |
| Number Of Points High Order | 0 (00) |
| Number Of Points Low Order | 4 (04) |

Tabelul nr. 8 – **Modbus Request ADU Example – Read Coil Status Query**

Acest exemplu citește statusul canalului de output al coil-urilor 0-3.

De reținut este faptul că primul caracter al adresei de referință 0x este implicat de codul funcției și este omis din adresa specificată. În acest exemplu, prima adresă este 00001, la care se face referire prin 0000H și corespunde coil-ului 0.

|  |  |
| --- | --- |
| **MBAP Header Fields** | **Example Decimal (Hexadecimal)** |
| Transaction ID High Order | 0 (00) Echoed back, no change. |
| Transaction ID Low Order | 1 (01) Echoed back, no change. |
| Protocol Identifier High Order | 0 (00) Echoed back, no change. |
| Protocol Identifier Low Order | 0 (00) Echoed back, no change. |
| Length High Order | 0 (00) Server calculates. |
| Length Low Order | 4 (04) Server calculates. |
| Unit Identifier | 255 (FF) or 0 (00) No change. |

Tabelul nr. 9 – **Modbus Response ADU Example Header**

|  |  |
| --- | --- |
| **Field Name** | **Example Decimal (Hexadecimal)** |
| Function Code | 1 (01) |
| Byte Count | 1 (01) |
| Data (Coils 3-0) | 10 (0A) |

Tabelul nr. 10 – **Modbus Response ADU Example – Read Coil Status Response**

**Notă**: codul funcției de răspuns este același cu al funcției de solicitare. Identificatorul tranzacției este păstrat de către server și apoi returnat. Identificatorul de protocol rămâne 0 pentru Modbus. Lungimea răspunsului este calculată de către server și este reprezentată de dimensiunea PDU a serverului Modbus la care se adaugă identificatorul unității (un octet). Identificatorul de unitate este același cu ceea ce s-a recepționat de la client.

Dacă a apărut o eroare, codul funcției de răspuns este modificat și setat ca fiind egal cu codul funcției de request la care se adaugă 80H. ID-ul tranzacției, ID-ul protocolului și identificatorul unității rămân aceiași. Lungimea devine 0002H (2 octeți). PDU devine apoi valoarea codului de excepție în sine (un octet). Pentru a obține informații legate de codurile de excepție trebuie să verificăm Modbus Exception.

Pentru a rezuma, statusul coil-urilor 3-0 este afișat ca valoare a octetului 0A în hexa sau 00001010 în binar. Al 3-lea coil este al 5-lea bit din stânga acestui octet, iar coil-ul 0 este LSB. Cei 4 biți rămași (spre capătul de rang superior/înalt) sunt 0. Citind de la stânga la dreapta, statusul coil-urilor 3...00 sunt ON-OFF-ON-OFF.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bin** | **0** | **0** | **0** | **0** | **1** | **0** | **1** | **0** |
| **Hex** | **0** | | | | **A** | | | |
| **Coil** | **NA** | **NA** | **NA** | **NA** | **3** | **2** | **1** | **0** |

Tabelul nr. 11

Următoarele exemple nu repetă informațiile conținute în header-ele MBAP de cerere sau răspuns, fiind furnizate numai informațiile legate de Modbus PDU. Pentru formatul antetului MBAP se consultă exemplul precedent.

***Read Holding Registers (03)***

Citește conținutul binar al holding registers (adresele de referință 4x) în dispozitivul slave. Transmisia prin difuzie nu este suportată. Interogarea Read Holding Registers specifică registrul de pornire și numărul de registre care trebuie citite.

**Notă**: registrele sunt adresate începând de la 0 (resistrele 1-16 sunt adresate ca 0-15). Răspunsul la mesaje în cazul Read Holding Registers este împachetat astfel: 2 octeți per registru, cu conținutul binar aliniat la dreapta (right-justified) în fiecare octet. Pentru fiecare registru, primul octet conține biții de ordin înalt și al 2-lea octet conține biții de ordin inferior.

Următorul exemplu realizează citirea pentru holding registers 40006...40008 (valoarea limită superioară pentru canalul 0, valoarea limită inferioară și valoarea de deadband).

|  |  |
| --- | --- |
| **Field Name** | **Example Decimal (Hexadecimal)** |
| Function Code | 3 (03) |
| Starting Adress High Order | 0 (00) |
| Starting Adress Low Order | 5 (05) |
| Number of Points High Order | 0 (00) |
| Number of Points Low Order | 3 (03) |

Tabelul nr. 12 – **Modbus PDU Example – Read Holding Register Query**

|  |  |
| --- | --- |
| **Field Name** | **Example Decimal (Hexadecimal)** |
| Function Code | 3 (03) |
| Byte Count | 6 (06) |
| Data High (Register 40006) | (3A) |
| Data Low (Register 40006) | 75%=15000 (98) |
| Data High (Register 40007) | (13) |
| Data Low (Register 40007) | 25%=5000 (88) |
| Data High (Register 40008) | (00) |
| Data Low (Register 40008) | 1%=200 (C8) |

Tabelul nr. 13 – **Modbus PDU Example – Read Holding Register Response**

Pentru a rezuma exemplul, conținutul registrului 40006 (2 octeți) reprezintă limita superioară de 75% a canalului 0 (15000=3A98H). Conținutul registrului 40007 (2 octeți) reprezintă limita inferioară de 25% a canalului 0 (5000=1388H). Conținutul registrului 40008 reprezintă valoarea de deadband (2 octeți) de 1% a canalului 0 (200=00C8H).

***Read Input Registers (04)***

Această comandă va citi conținutul binar al registrelor de input (adresele de referință 3x) în dispozitivul slave. Transmisia prin difuzie nu este suportată.

Interogarea Read Input Registers specifică registrul de start și numărul de registre care trebuie citite. Registrele sunt adresate începând cu 0. Astfel, registrele 1-16 sunt adresate de fapt ca fiind 0-15. Răspunsul la mesaje al Read Input Registers este împachetat astfel: 2 octeți per registru, conținutul binar fiind aliniat la dreapta (right-justified) în fiecare octet. Pentru fiecare registru, primul octet conține biții de ordin înalt, iar al 2-lea octet conține biții de ordin inferior.

Următorul exemplu citește registrele de input 30003 și 30004 (valoarea și statusul canalului 0 de input).

|  |  |
| --- | --- |
| **Field Name** | **Example Decimal (Hexadecimal)** |
| Function Code | 4 (04) |
| Starting Address High Order | 0 (00) |
| Starting Address Low Order | 2 (02) |
| Number Of Points High Order | 0 (00) |
| Number Of Points Low Order | 2 (02) |

Tabelul nr. 14 – **Modbus PDU Example – Read Input Registers Query**

|  |  |
| --- | --- |
| **Field Name** | **Example Decimal (Hexadecimal)** |
| Function Code | 4 (04) |
| Byte Count | 4 (04) |
| Data High (Register 30003) | (3E) |
| Data Low (Register 30003) | 80%=16000 (80) |
| Data High (Register 30004) | (00) |
| Data Low (Register 30004) | 136 (88) |

Tabelul nr. 15 – **Modbus PDU Example – Read Input Registers Response**

Pentru a rezuma exemplul, conținutul registrului 30003 (2 octeți) reprezintă valoarea de input de 80% a canalului 1 (16000=3E80H). Conținutul registrului 30004 (2 octeți) reprezintă flag-urile de status pentru 136 (0088H) ale canalului 0; adică semnalizează depășirea limitei superioare.

***Force Single Coil (05)***

Forțează un singur coil/output (adresele de referință 0x) să fie ON sau OFF. Cu o transmisie de tip broadcast (adresa 0), forțează același coil în toate serverele din rețea (numai în cazul Modbus serial). Interogarea Force Single Coil specifică adresa de referință a coil-ului care trebuie să fie forțată, iar statusul care de asemenea trebuie impus. Starea ON/OFF este indicată prin intermediul unei constante din câmpul de date al interogării. O valoare de FF00H forțează coil-ul să fie pe ON (adică solid-state-ul releului corespunzător este ON sau închis), iar 0000H forțează coil-ul să fie pe OFF (adică solid-state-ul releului corespunzător este OFF sau deschis). Toate celelalte valori sunt nevalide și nu vor afecta coil-ul.

Coil-urile sunt indexate de la 0 până la 4, fiind adresate de fapt de la 0 la 3 în cazul acestui exemplu și asta corespunde numărului de canal de output discret.

Următorul exemplu forțează ieșirea discretă 3 pe ON.

|  |  |
| --- | --- |
| **Field Name** | **Example Decimal (Hexadecimal)** |
| Function Code | 5 (05) |
| Coil Adress High Order | 0 (00) |
| Coil Adress Low Order | 3 (03) |
| Force Data High Order | 255 (FF) |
| Force Data Low Order | 0 (00) |

Tabelul nr. 16 – **Modbus PDU Example – Force Single Coil Query and Response**

Mesajul de răspuns pentru Force Single Coil este pur și simplu o copie a interogării, așa cum se arată mai sus, dar returnat după executare comenzii de force coil. Nu se returnează niciun răspuns la interogările de difuzie lansate de pe un dispozitiv master (doar în cazul Modbus serial).

***Preset Single Register (06)***

Această comandă va preseta un singur holding register (adresa de referință 4x) la o valoare specifică. Transmisia prin difuzie este suportată de această comandă (doar în cazul Modbus serial) și va acționa pentru a preseta același registru în toate server-ele din rețea.

Interogarea pentru Preset Single Registers specifică adresa de referință a registrului care trebuie să fie presetat și valoarea de preset. Registrele sunt adresate începând cu 0; registrele 1-16 sunt de fapt adresate ca 0-15. Mesajul de răspuns pentru Preset Single Registers reprezintă o copie (echo) al interogării, fiind returnată după ce conținutul registrului a fost presetat.

Următorul exemplu scrie un baud rate de 9600bps într-un holding register 40002 (BaudRate).

|  |  |
| --- | --- |
| **Field Name** | **Example Decimal (Hexadecimal)** |
| Function Code | 6 (06) |
| Register Adress High Order | 0 (00) |
| Register Adress Low Order | 1 (01) |
| Preset Data High Order | 0 (00) |
| Preset Data Low Order | 2 (02) |

Tabelul nr. 17 – **Modbus PDU Example – Preset Holding Register Query and Response**

Mesajul de răspuns este pur și simplu o copie a interogării așa cum reiese și de mai sus, dar returnat după ce conținutul registrului a fost presetat. Nu se returnează niciun răspuns la interogările de tip broadcast inițiate de un master (doar în cazul dispozitivelor Modbus).

***Force Multiple Coils (15)***

Forțează simultan o serie de coil-uri (adresa de referință 0x) să fie ON sau OFF. Transmisia prin difuzie este suportată de această comandă (doar pentru Modbus serial) și va acționa pentru a forța blocarea aceluiași bloc de coil-uri în server-ele din rețea.

Interogarea Force Multiple Coils specifică adresa de referință pentru coil-ul de start care trebuie să fie forțat, numărul de coil-uri și datele care trebuie să fie scrise forțat (force data) în ordine crescătoare. Stările ON/OFF sunt specificate de conținuturile câmpului de date al interogării. Valoarea 1 logic a unui bit aflat pe o poziție din acest camp solicită ca respectivul coil să treacă pe ON, în timp ce un 0 logic solicită ca trecerea să se facă în OFF pentru coil-ul corespunzător. Biții neutilizați dintr-un octet de date ar trebui să fie setați pe 0. Coil-urile sunt indexate începând cu 0; până la 4, coil-urile sunt adresate ca 0-3 pentru acest exemplu, iar acest lucru corespunde, de asemenea, numărului canalului de ieșire discret.

Următorul exemplu forțează ieșirile 1&3 pe OFF și ieșirile 0&2 pe ON pentru coil-urile 0-3.

|  |  |
| --- | --- |
| **Field Name** | **Example Decimal (Hexadecimal)** |
| Function Code | 15 (0F) |
| Coil Adress High Order | 0 (00) |
| Coil Adress Low Order | 0 (00) |
| Number Of Coils High Order | 0 (00) |
| Number Of Coils Low Order | 4 (04) |
| Byte Count | 01 |
| Force Data High (First Byte) | 5 (05) |

Tabelul nr. 18 – **Modbus PDU Example – Force Multiple Coils Query**

Primul caracter al adresei de referință 0x este implicat de codul funcției și omis din adresa specificată. În acest exemplu, prima adresă este 00001 și corespunde coil-ului 0; aceasta este referită prin 0000H. Astfel, în acest exemplu octetul de date transmis va adresa coil-urile 3...0, având bit-ul cel mai puțin semnificativ responsabil de adresarea celui mai mic coil din acest set după cum urmează (cei 4 biți din zona superioară a octetului de date care nu sunt utilizați, sunt setați pe 0):

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bin** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **1** |
| **Hex** | **0** | | | | **5** | | | |
| **Coil** | **NA** | **NA** | **NA** | **NA** | **3** | **2** | **1** | **0** |

Tabelul nr. 19

|  |  |
| --- | --- |
| **Field Name** | **Example Decimal (Hexadecimal)** |
| Function Code | 15 (0F) |
| Coil Adress High Order | 0 (00) |
| Coil Adress Low Order | 0 (00) |
| Number Of Coils High Order | 0 (00) |
| Number Of Coils Low Order | 4 (04) |

Tabelul nr. 20 – **Modbus PDU Example – Force Multiple Coils Response**

Mesajul normal de răspuns pentru Force Multiple Coils returnează adresa slave-ului, codul funcției, adresa de start și numărul de coil-uri ,,forțate” după efectuarea instrucțiunii de forțare (force instruction). Nu se returnează numărul octeților sau valoarea impusă (force value). De asemenea, nu se returnează niciun răspuns la interogările de broadcast de la vreun dispozitiv master (Modbus serial).

***Preset Multiple Registers (16)***

Presetează un bloc de holding registers (adresa de referință 4x) la o anumită valoare. Transmisia prin difuzie este suportată de această comandă și va acționa pentru a preseta același bloc de registre în toate server-ele din rețea (doar pentru Modbus serial).

Interogarea în cazul Preset Multiple Registers specifică adresa de referință a registrului de start, numărul de registre și datele care trebuie scrise în ordine crescătoare. Registrele sunt adresate începând de la 0; registrele 1-16 sunt adresate ca 0-15.

Următorul exemplu scrie adresa unui nou server de 200, un baud rate de 28800bps și setează paritatea pară prin scrierea în holding register 40001 până la 40003 (schimbări făcute la adresa server-ului, baud rate și paritate vor fi vizibile după următorul software sau resetare).

|  |  |
| --- | --- |
| **Field Name** | **Example Decimal (Hexadecimal)** |
| Function Code | 16 (10) |
| Starting Register High Order | 0 (00) |
| Starting Register Low Order | 0 (00) |
| Number Of Registers High Order | 0 (00) |
| Number Of Registers Low Order | 3 (03) |
| Preset Data High (First Register) | 0 (00) |
| Preset Data Low (First Register) | 200 (C8) |
| Preset Data High (Second Reg) | 0 (00) |
| Preset Data Low (Second Reg) | 5 (05) |
| Preset Data High (Third Reg) | 0 (00) |
| Preset Data Low (Third Reg) | 2 (02) |

Tabelul nr. 21 – **Modbus PDU Example – Preset Multiple Registers Query**

|  |  |
| --- | --- |
| **Field Name** | **Example Decimal (Hexadecimal)** |
| Function Code | 16 (10) |
| Starting Register High Order | 0 (00) |
| Starting Register Low Order | 0 (00) |
| Number Of Registers High Order | 0 (00) |
| Number Of Registers Low Order | 3 (03) |

Tabelul nr. 22 – **Modbus PDU Example – Preset Multiple Registers Response**

Răspunsul normal al unui Preset Multiple Registers returnează adresa server-ului, codul funcției, referința registrului de start și numărul de registre presetate, după ce conținuturile registrelor au fost presetate. Valorile presetate nu sunt repetate. Nu se returnează niciun răspuns pentru interogările de broadcast de la un dispozitiv master (doar pentru Modbus serial).

***Report Slave ID (17)***

Această comandă returnează modelul, seria și numărul de firmware pentru un dispozitiv server/slave Acromag (97xEN de exemplu), statusul indicatorului de run și alte informații specifice dispozitivului. Această comandă nu adresează registre din harta de registre, iar transmisia broadcast nu este suportată (Modbus serial).

Interogarea comenzii pur și simplu trimite identificatorul unității și codul funcției.

|  |  |
| --- | --- |
| **Field Name** | **Example Decimal (Hexadecimal)** |
| Function Code | 17 (11) |

Tabelul nr. 23 – **Modbus PDU Example – Report Slave ID Query**

|  |  |
| --- | --- |
| **FIELD** | **DESCRIPTION** |
| Unit ID | Echo Unit ID Sent In Query |
| Function Code | 11 |
| Byte Count | 42 |
| Slave ID (Model No.) | 08=972EN-4004 (4 Current Outputs)  09=972EN-4006 (6 Current Outputs)  0A=973EN-4004 (4 Voltage Outputs)  0B=973EN-4006 (6 Voltage Outputs) |
| Run Indicator Status | FFH (ON) |
| Firmware Number String (Additional Data Field) | 41 43 52 4F 4D 41 47 2C 39 33 30 30 2D  **31 32 37** 2C 39 **37 32** 45 4E 2D **34 30 30 36**  2C  30 31 32 33 34 35 41 2C 30 31 32 33 34 35  (“ACROMAG, 9300-**127**, 9**72**EN-**4006**, serial number&rev, six-byteMACID”) |

Tabelul nr. 24 – **Report Slave ID Response (Acromag 97xEN)**

1. **Tipuri de date acceptate (module Acromag)**

Toate valorile I/O sunt accesate prin intermediul unui Input Register de 16 biți sau prin intermediul unui Holding Register de 16 biți (vezi harta de registre). Input Registers conțin informații care sunt read-only; de exemplu, valoarea curentă pentru input citită de la un canal de intrare sau stările unui grup de intrări digitale. Holding Registers conțin informații read/write care pot fi date de configurare sau date de ieșire. De exemplu, valoarea limită superioară a unei funcții de alarmare care operează la o intrare sau o valoare de ieșire pentru un canal de ieșire.

Valorile I/O ale modulelor Acromag iau următoarele forme comune de date pentru a reprezenta: temperatura, procentajul, valorile discrete ON/OFF după cum este necesar. Acesta nu este neapărat un standard Modbus și va varia de la un dispozitiv la altul.

|  |  |
| --- | --- |
| **Data Types** | **Description** |
| Count Value | O valoare întreagă cu semn pe 16 biți ce reprezintă o contorizare A/D, o contorizare DAC, valoarea timpului sau frecvența cu valori cuprinse între -32768 și +32767. |
| Count Value | O valoare întreagă fără semn pe 16 biți ce reprezintă o contorizare A/D, o contorizare DAC, valoarea timpului sau frecvența cu valori cuprinse între 0 și 65535. |
| Percentage | O valoare întreagă cu semn pe 16 biți cu o rezoluție de 0,005%/lsb ±20000 este utilizată pentru a rerezenta ±100%. De exemplu, -100%, 0% și +100% sunt reprezentate prin valori zecimale: -20000, 0 și 20000. Gama completă este de la -163.84% (-32768 în zecimal) la +163.835% (+32767 în zecimal). |
| Temperature | O valoare întreagă cu semn pe 16 biți cu o rezoluție de 0.1°C/lsb. De exemplu, o valoare de 12059 este echivalentă cu 1205.9°C, iar o valoare de -187 este egală cu -18.7°C. Intervalul maxim posibil de temperaturi este de la -3276.8°C până la +3276.7°C. |
| Discrete | O valoare discretă este în general indicată de un singur bit dintr-un cuvânt (word) pe 16 biți. Numărul/poziția bit-ului în general corespunde numărului canalului discret pentru acest model. Dacă nu este definit astfel pentru output, un bit 1 semnifică faptul că ieșirea corespunzătoare este închisă sau ON, iar un bit 0 semnifică faptul că ieșirea este deschisă sau OFF. Pentru intrări, o valoare de 1 înseamnă că intrarea este în starea high (în general >>0V), în timp ce o valoare de 0 specifică faptul că intrarea este în starea low (aproape de 0V). |

Tabelul nr. 25 – **Summary Of Data Types Used By Acromag 900MB/900 EN Modules**

În cazul Modbus TCP/IP, verificarea erorilor pentru date este gestionată de protocolul subadiacent TCP, iar clasica verificare serială Modbus nu va fi revizuită aici.

1. **Excepții Modbus**

Reamintim că un server poate genera un răspuns de excepție la o cerere a clientului și acest lucru este în mod normal marcat prin returnarea codului funcției originale la care se adaugă 80H (codul original având MSB setat). În plus, acesta poate returna, de asemenea, un cod de excepție în câmpul de date al răspunsului care poate fi utilizat pentru rezolvarea problemei.

De exemplu, dacă un client solicită o opțiune care nu este suportată (specifică un cod de funcție invalid), atunci server-ul poate returna codul de excepție 01 (Illegal Function) în câmpul de date al mesajului de răspuns. De asemenea, dacă un holding register este scris cu o valoare invalidă, atunci codul de excepție 03 (Illegal Data Value) va fi returnat în câmpul de date al mesajului de răspuns. Următorul tabel oferă câteva coduri de eroare comune:

|  |  |  |
| --- | --- | --- |
| **Code** | **Exception** | **Description** |
| 01 | Illegal Function | Codul funcției primit în interogare nu este permis sau este invalid. |
| 02 | Illegal Data Address | Adresa de date primită în interogare nu este o adresă permisă pentru slave sau este nevalidă. |
| 03 | Illegal Data Value | O valoare conținută în câmpul de date al interogării nu este o valoare permisă pentru slave sau este o valoare nevalidă. |
| 04 | Slave/Server Device Failure | Server-ul a eșuat în timpul execuției. A apărut o eroare irecuperabilă în timp ce server-ul încerca să efectueze acțiunea solicitată. |
| 05 | Acknowledge | Server-ul a acceptat cererea și o prelucrează, dar este necesară o durată prea mare de timp pentru finalizarea procesării. Acest răspuns este returnat pentru a preveni apariția unei erori de tip timeout în cadrul dispozitivului master. |
| 06 | Slave/Server Device Busy | Server-ul este angajat în procesarea unei comenzi de program de lungă durată. Master-ul ar trebui să retransmită mesajul ulterior, atunci când server-ul este liber. |
| 07 | Negative Acknowledge | Server-ul nu poate să efectueze funcția de program receptată în interogare. Acest cod este returnat în cazul unei cereri de program nereușite, utilizând codul funcției 13 sau 14 (codurile nu sunt valabile și pentru acest model). Dispozitivul master ar trebui să solicite o serie de informații de diagnoză de la slave. |
| 08 | Memory Parity Error | Server-ul a încercat să citescă memoria extinsă, dar a detectat o eroare de paritate în memorie. Master-ul poate reîncerca solicitarea, dar serviciul poate fi necesar și la dispozitivele de tip slave. |
| 0A | Gateway Problem | Căile de acces (gateway path(s)) nu sunt disponibile. |
| 0B | Gateway Problem | Dispozitivul țintă nu a reușit să răspundă (gateway-ul generează această excepție). |
| FF | Extended Exception Response | Răspunsul de excepție PDU conține informații extinse legate de excepție. Un câmp ulterior cu lungimea de 2 octeți indică dimensiunea în octeți a codului funcției specific informației legate de excepție. |

Tabelul nr. 26 – **Modbus Exception Codes**

Într-un răspuns normal, server-ul pur și simplu copiază codul funcției al interogării originale în câmpul funcției din cadrul răspunsului. Toate codurile de funcții au MSB-ul setat pe 0 (valorile lor sunt sub 80H). Într-un mesaj de excepție, server-ul setează MSB din codul funcției la valoarea 1 în răspunsul returnat (adică exact cu 80H mai mare decât în mod normal) și returnează codul de excepție în cadrul câmpului de date. Acesta este utilizat de aplicațiile client/master de fapt pentru a recunoaște un răspuns de excepție și pentru a dirija o examinare (evaluare) a câmpului de date pentru codul de excepție aplicabil.

TCP este un protocol bazat pe stream-uri de date; poate trimite pachete de aproape orice lungime IP pe care le alege și poate parsa mesajele după cum se solicită. De exemplu, acesta poate încapsula 2 mesaje de încapsulare back-to-back într-un singur pachet TCP/IP/MAC sau poate diviza un mesaj de încapsulare în 2 pachete separate TCP/IP/MAC.

1. **Modbus TCP/IP ADU Format**

În introducere, am prezentat modul în care un mesaj clasic Modbus (Modbus Application Data Unit) a fost separat de suma de control și de câmpul de adresă al dispozitivului; ulterior la acesta s-a adăugat un antet MBAP (Modbus Application Protocol) pentru a construi o unitate Modbus TCP/IP Application Data Unit (ADU). Această informație este apoi imbricată în câmpul data/payload dintr-un cadru standard TCP total, din care ulterior s-a imbricat în cadrul IP, care apoi a fost imbricat în cadrul Ethernet/MAC pentru transmisia prin Ethernet. Această imbricare reprezintă de fapt procesul de încapsulare a mesajelor la care se face referire în mod obișnuit. Următoarele secțiuni vor încerca să descrie încapsularea care apare la fiecare nivel pe măsură ce deplasarea se realizează descendent în cadrul stivei către mediul de conexiune, pornind de la *nivelul aplicație* Modbus.

Se spune că nivelul aplicație rulează deasupra TCP. Înainte de transmiterea mesajului de aplicație prin intermediul TCP, un Applcation Data Unit Modbus TCP/IP este format dintr-un antet Protocol de Aplicație Modbus (MBAP) de 7 octeți și Protocol Data Unit (codul funcției și datele Modbus).

Acest pachet are următoarea formă:

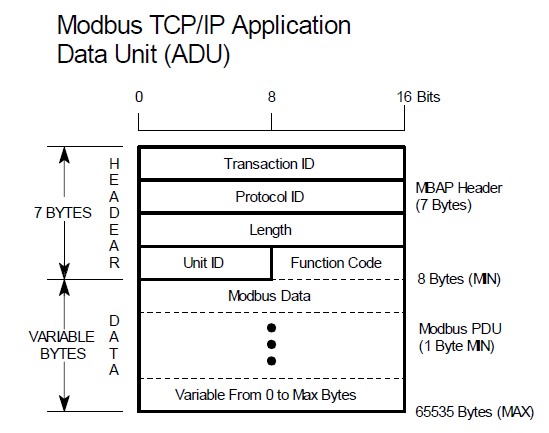


Fig. 6 **Modbus TCP/IP Application Data Unit (ADU)**

Antetul MBAP de 7 octeți include următoarele câmpuri:

* **Transaction/Invocation Identifier** (2 octeți): acest câmp de identificare este utilizat pentru asocierea tranzacțiilor atunci când mai multe tranzacții Modbus sunt trimise în cadrul aceleiași conexiuni TCP fără a se aștepta finalizarea tranzacției anterioare.
* **Protocol Identifier** (2 octeți): acest câmp este mereu 0 pentru serviciile Modbus, iar alte valori sunt rezervate pentru viitoarele extensii.
* **Length**: acest câmp reprezintă un număr de octeți din câmpurile rămase și include identificarea destinației și câmpurile de date.
* **Unit Identifier** (1 octet): acest câmp este utilizat pentru a identifica un remote server localizat într-o rețea non-TCP/IP (pentru conectarea Ethernet la o subrețea serială). Într-o aplicație clasică server, ID-ul unității este ignorat și este pur și simplu copiat în răspuns. Se recomandă ca un ID de unitate FF să fie utilizat pentru a păstra această valoare insignifiantă spre un bridge sau gateway (vezi mai jos).

Protocol Data Unit (PDU) reprezintă codul funcției Modbus și câmpul de date în formele lor originale. Câmpul original de verificare a erorilor Modbus (suma de control) nu este utilizat; în cazul Ethernet TCP/IP, metodele standard de checksum pentru *nivelul legătură* sunt utilizate în scopul garantării integrității datelor. Mai mult, cadrul câmpului de adresă Modbus este înlocuit de identificatorul unității în Modbus TCP/IP și devine parte integrantă a antetului Modbus Application Protocol (MBAP). Adresa originală a dispozitivului nu este necesară deoarece dispozitivele Ethernet deja conțin propriile adrese unice MAC. Cu toate acestea, este utilizat dacă un bridge/gateway serial este utilizat pentru a realiza legătura dintre Ethernet și o subrețea serială a dispozitivelor Modbus.

Prin intermediul protocolului clasic Modbus serial, un client poate să trimită o singură cerere la un moment de timp și trebuie să aștepte un răspuns înainte de a trimite a 2-a cerere. Cu toate acestea, dispozitivele Modbus TCP/IP pot trimite mai multe solicitări către același server fără a aștepta răspunsul anterior. În acest caz, identificatorul de tranzacție este utilizat pentru a realiza potrivirea dintre un viitor răspuns cu cererea sa inițială și trebuie să fie unic per tranzacție. De obicei este un număr de secvență dat de un counter care este incrementat la fiecare cerere. Numărul maxim de tranzacții client va varia în funcție de dispozitiv, dar în general este un număr cuprins între 1 și 16. De asemenea, numărul maxim de tranzacții server poate varia. De exemplu, pentru modulele Acromag 9xxEN-4, acest număr este 10.

Câmpul de identificare a unității a fost destinat să faciliteze comunicația între dispozitivele Ethernet Modbus TCP/IP și dispozitivele clasice Modbus serial prin utilizarea unui bridge/gateway pentru rutarea traficului dintr-o rețea către o linie serială dintr-o subrețea. În acest caz, adresa destinație IP identifică bridge-ul/gateway-ul spre care trimite mesajul, în timp ce dispozitivul bridge în sine folosește identificatorul unității Modbus pentru a trimite mai departe cererea către dispozitivul slave specific subrețelei. Modbus serial utilizează adresele de la 1 la 247 (în zecimal) și rezervă adresa 0 ca adresă de difuzie. Astfel, identificatorul unității își asumă același task pentru aceste aplicații. Mai mult, acesta este singurul mod prin care mesajele broadcast sunt acceptate de Modbus TCP/IP; dacă se folosește simplu TCP, sunt trimise doar mesajele unicast (point-to-point).

1. **Modbus TCP/IP Application Data Unit**

Cu dispozitivele TCP/IP, server-ul Modbus este adresat prin utilizarea adresei sale IP, făcând identificatorul unității non-funcțional, iar FFH este utilizat în locul său. Adresa în hexa a FF rămâne nesemnificativă pentru gateway sau bridge și va continua să fie ignorată dacă ulterior rețeaua este expandată sau mărită cu dispozitive de tip bridge/gateway serial.

ADU în cazul Modbus TCP/IP este apoi inserat în câmpul de date al unui cadru standard TCP pe binecunoscutul port 502, care este rezervat special aplicațiilor Modbus. Astfel, acest pachet este încapsulat de cadrele de date impuse de stiva de protocoale TCP/IP (TCP/IP/MAC) înainte de a fi transmis în rețea. Termenul de încapsulare se referă la acțiunea de a împacheta (încorpora) acest mesaj în containerul TCP, containerul IP și containerul MAC. Nivelul de bază de încapsulare este ilustrat după cum urmează:

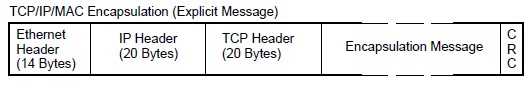


Fig. 7 **TCP/IP/MAC Encapsulation (Explicit Message)**

1. **Connection Manager**

Deoarece TCP este un protocol orientat pe conexiune, trebuie să se stabilească mai întâi o conexiune TCP înainte ca un mesaj să poată fi transmis prin intermediul Modbus TCP/IP. Urmând principiul client/server, această conexiune este stabilită de către client (master). Această conexiune poate fi gestionată în mod explicit de către aplicația software client sau automat de către managerul de conexiune client TCP. Mai frecvent, acest lucru este gestionat automat de către sofware-ul protocolului client prin intermediul interfeței socket TCP și această operațiune rămâne transparentă pentru aplicație.

Toate conexiunile de mesaje TCP/IP sunt de tip comunicații cu căi point-to-point între 2 dispozitive, care necesită o adresă sursă, o adresă destinație și un ID de conexiune în fiecare direcție. Astfel, comunicațiile TCP/IP sunt restricționate doar la tipul de mesaje unicast.

Binecunoscutul port 502 a fost rezervat în mod special pentru aplicațiile Modbus. Un server Modbus va asculta (listen) comunicațiile pe portul 502. Când un client Modbus dorește să trimită un mesaj către un server remote Modbus, se deschide o conexiune cu portul remote 502. De îndată ce conexiunea a fost stabilită, aceasta poate fi utilizată pentru a trimite date utilizator în ambele direcții între client și server. Un client și un server pot stabili mai multe conexiuni TCP/IP simultan. Când o conexiune este stabilită, toate transmisiile care fac parte din respectiva conexiune sunt asociate cu un ID de conexiune (Connection ID - CID). Dacă această conexiune implică transmisii în ambele direcții, atunci sunt asignate 2 ID-uri de conexiune. Numărul maxim de conexiuni permise depinde de specificațiile particulare ale interfeței TCP/IP (pentru modulul Acromag 9xxEN, acest număr este 10). În cazul trasmisiei ciclice între client și server, poate fi utilizată o conexiune permanentă. Dacă este necesar doar transferul unor parametri sau a unor informații de diagnoză în cazul în care se produce un eveniment, atunci acestă conexiune poate fi închisă după fiecare transmisie de date și redeschisă atunci când este necesar.

În ceea ce privește Modbus TCP/IP, următoarele afirmații sunt adevărate:

* O conexiune TCP este stabilită de către client. Server-ele nu pot iniția tranzacții TCP. Este o bună practică să păstrați o conexiune TCP deschisă cu un server remote și să nu o deschideți și să o închideți pentru fiecare mesaj Modbus. Clientul poate închide o conexiune dapă cum este necesar, dar poate de asemenea procesa și un mesaj de cerere de închidere de la un server și apoi să închidă conexiunea.
* Unele dispozitive Modbus pot funcționa atât ca servere, cât și drept clienți. Modulele Acromag 900EN-40xx sunt strict servere.
* Pentru dispozitivele care îndeplinesc ambele funcționalități (client&server), comunicația este posibilă în 2 direcții, având conexiuni separate deschise pentru fluxul de date client și pentru fluxul de date server.
* Clientul Modbus poate avea mai multe conexiuni TCP deschise simultan la un moment dat. Acesta utilizează un port local pentru a trimite mesajul (diferit de 502 și diferit pentru fiecare conexiune), în timp ce server-ul remote primește mesajul pe binecunoscutul port 502.
* Server-ul Modbus ascultă întotdeauna mesajele pe binecunoscutul port 502 (în unele aplicații poate avea și alte porturi TCP pentru serviciul Modbus). Un client Modbus nu trimite mesaje pe portul local 502. Portul 502 este rezervat pentru ascultare/recepție (listening/receiving).
* Un client poate iniția mai multe mesaje Modbus cu un server remote fără să aștepte terminarea unuia precedent. Astfel, mai multe mesaje Modbus, pot fi trimise pe aceeași conexiune TCP (dar nu pe același PDU). În acest caz, identificatorul de tranzacție Modbus (al antetului MBAP) este utilizat pentru a realiza asocierea dintre cereri și răspunsurile corespunzătoare.
* Un cadru TCP trebuie să transporte un singur Modbus Application Data Unit (ADU) la un moment dat. Deși este posibil, nu este recomandat să se trimită mai multe cereri Modbus pe același PDU.
* Un client Modbus va deschide un minim de conexiuni cu un server remote Modbus cu aceeași adresă IP, de regulă una per aplicație.
* Un server Modbus poate avea mai multe conexiuni TCP în același timp. Acesta utilizează portul 502 pentru a recepționa aceste mesaje și ID-ul conexiunii pentru a realiza asocierea dintre mesaj și răspuns. Numărul maxim de conexiuni TCP va varia în funcție de dispozitiv. Pentru modulele Acromag 9XXEN-40xx Modbus TCP/IP, acest număr este 10.

Desigur, trebuie să existe și un proces pentru a putea stabili conexiuni între dispozitivele care nu sunt conectate încă. Acesta este scopul Managerului de Mesaje Neconectate (Unconnected Message Manager UCMM), care servește la procesarea acestor solicitări de conectare. Odată ce UCMM a stabilit o conexiune, atunci toate resursele necesare conexiunii dintre 2 dispozitive (incluzând bridge-uri/routere) sunt rezervate pentru respectiva conexiune.

1. **Nivelul Transport**

*Nivelul transport* se află chiar sub *nivelul aplicație* și este responsabil de transmisia, recepția și verificarea erorilor datelor. Există o serie de protocoale de *nivel transport* care pot acționa asupra acestui nivel, dar principalul și cel mai de interes pentru Modbus TCP/IP este Transport Control Protocol (TCP).

1. **TCP – Transport Control Protocol**

Transport Control Protocol (TCP) se află pe un nivel deasupra protocolului IP și este responsabil de transportul datelor aplicației și asigurarea securității, în timp ce IP este responsabil pentru adresarea efectivă și livrarea datelor. Pachetul TCP este inserat în zona de date a pachetului IP de sub acesta. IP în sine este un protocol nesecurizat, fără conexiuni și trebuie să lucreze împreună cu TCP pentru a putea funcționa. În acest fel, TCP este considerat, în general, nivelul superior al platformei IP care servește la garantarea transferului sigur de date. Utilizarea etichetei Modbus-TCP (prin comparație cu Modbus TCP/IP) nu implică faptul că IP nu este utilizat sau nu este important.

Dacă datele se pierd, acestea trebuie retransmise. Acest tip de schimb de date se referă la mesajele explicite și este utilizat în mod obișnuit pentru schimbul de informații care nu sunt time-critical, dar rămân totuși necesare. TCP folosește mesajele explicite și va lucra pentru a asigura faptul că mesajele sunt recepționate, dar nu neapărat la timp (on time).

TCP este un protocol orientat pe conexiuni. TCP stabilește o conexiune între 2 stații de rețea pe durata transmisiei datelor. În timp ce se stabilește conexiunea, condiții precum dimensiunea pachetelor de date sunt specificate (care se aplică întregii sesiuni de conexiune).

TCP urmează modelul de comunicație client/server. De aceea, stația din rețea care care va lua inițiativa și va stabili conexiunea va fi denumită Client TCP. Stația către care s-a realizat conexiunea se numește Server TCP. În Modbus TCP/IP, comunicația este mereu controlată de către master (client) și tot master-ul va realiza conexiunea. Server-ul nu poate iniția comunicația din inițiativă proprie, ci doar așteaptă ca master-ul să realizeze un contact între ei. Clientul utilizează apoi serviciul oferit de către server (în funcție de serviciu, un server poate găzdui mai mulți clienți simultan).

TCP verifică datele utilizator trimise cu o sumă de control și asignează secvențial numere (o secvență de numere) pentru fiecare pachet trimis. Receptorul unui pachet TCP utilizează o sumă de control pentru a verifica primirea/recepționarea corectă a datelor. Odată ce server-ul a recepționat corect pachetul, acesta utilizează un algoritm predefinit pentru a calcula numărul de confirmare (acknowledgement number) din secvența de numere. Numărul de confirmare este returnat clientului împreună cu următorul pachet pe care îl trimite ca o confirmare (acknowledgement). De asemenea, server-ul atribuie o secvență de numere pachetelor pe care le trimite, care este confirmat apoi de către client cu un număr de confirmare. Acest proces ajută la asigurarea că orice pierdere de pachete TCP va fi notificată și dacă este nevoie, pot apoi să fie retrimise în ordinea corectă.

TCP direcționează datele utilizator de pe computerul destinație către programul de aplicație corect prin accesarea diverselor servicii de aplicație, folosind diverse numere de porturi binecunoscute. De exemplu, Telnet poate fi accesat prin intermediul portului 23, FTP prin intermediul portului 21 și Modbus prin intermediul portului 502. În acest fel, numărul portului este același cu numărul camerei în cadrul unei clădiri de birouri – dacă adresați o scrisoare către biroul de relații publice din camera 312, indicați că doriți să utilizați serviciile biroului de relații publice.

Un port este adresa care este utilizată local la *nivelul transport* (pe un singur nod) și identifică sursa și destinația pachetului în interiorul aceluiași nod. Numerele de porturi sunt împărțite între binecunoscutele numere de porturi (0-1023), numere de port user înregistrate (1024-49151) și numere de porturi private/dinamice (49152-65535). Porturile permit TCP/IP să multiplexeze și să demultiplexeze o secvență de datagrame IP care trebuie să ajungă (simultan) la diferite procese aplicație.

Pentru a reitera, cu TCP, emițătorul așteaptă ca receptorul să recunoască (confirme) primirea pachetelor de date. Eșuarea confirmării primirii pachetelor va face ca transmițătorul să mai transmită o dată pachetele sau va face ca legătura de comunicație să fie întreruptă. Deoarece fiecare pachet este numerotat, receptorul poate, de asemenea, să detemine dacă un pachet de date lipsește sau poate reordona pachetele care nu au fost primite în ordinea corectă. Dacă anumite date figurează ca fiind nerecepționate, toate datele primite ulterior vor fi bufferate. Datele vor fi apoi transmise în stiva de protocoale către aplicație, dar numai atunci când sunt complete și în ordine corectă. Desigur, acest mecanism de verificare a erorilor al protocolului TCP orientat pe conexiune necesită timp și va funcționa mai lent decât unul fără conexiuni cum ar fi protocolul UDP (UDP nu este utilizat în Modbus TCP/IP). Astfel, trimiterea unui mesaj prin intermediul TCP este fezabilă în cazul stream-urilor de date continue sau pentru cantități mari de date care trebuie schimbate (exchanged) sau atunci când se impune neapărat un grad ridicat de integritate a datelor (accentul fiind pus pe date sigure).

Următoara figură ilustrează contrucția unui pachet TCP:

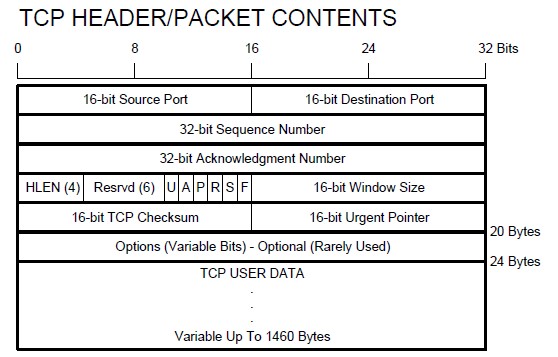


Fig. 8 **TCP HEADER/PACKET CONTENTS**

**Definirea câmpurilor header-ului TCP (de la stânga la dreapta și de sus în jos):**

* **Source Port (SP)**: portul de aplicație al expeditorului (portul expeditorului așteptând să asculte un răspuns de la mașina destinație).
* **Destination Port (DP)**: numărul portului de aplicație al receptorului (portul mașinii remote unde va fi recepționat pachetul trimis).
* **Sequence Number (SN)**: offset-ul de la primul octet de date în raport cu începutul fluxului TCP care este utilizat pentru a garanta că o secvență este menținută atunci când un mesaj de dimensiuni mari necesită mai mult de o transmisie.
* **Acknowledgement Number (AN)**: acesta reprezintă secvența de numere așteptată în următorul pachet TCP care urmează să fie trimis și funcționează prin confirmarea numărului de ordine așa cum a fost trimis de către gazda remote (host remote). Adică, AN-ul gazdei este o referință la SN-ul mașinii și SN-ul mașinii locale este în legătură cu AN-ul mașinii remote.
* **Header Length (HLEN)**: o măsură a lungimii antetului cu incrementare prin cuvinte de dimensiune egală cu 32 biți.
* **Reserved**: acești 6 biți sunt rezervați pentru pentru o posibilă utilizare viitoare.
* **UARPSF Flags (URG, ACK, PSH, RST, SYN, FIN)**: U=Urgent flag care specifică faptul că urgent point-ul inclus în acest pachet este valid; A=Acknowledgement flag care specifică faptul că porțiunea de antet care conține numărul de confirmare este validă; P=Push flag care indică stivei TCP/IP că aceasta (conținutul ei) ar trebui să fie trimisă sus către *nivelul aplicație* al programului care are nevoie de aceasta sau o solicită imediat ce timpul permite acest lucru; R=Reset flag care este utilizat pentru a reseta conexiunea; S=Synthesis flag care este utilzat pentru a sincroniza numerele de secvență cu numerele de confirmare pentru ambele gazde (sinteza conexiunii); F=Finish flag care este utilizat pentru a specifica faptul că o conexiune este finalizată în conformitate cu partea care a trimis pachetul având flag-ul F setat.
* **Window Size (WS)**: aceasta indică numărul de octeți care pot fi primiți pe partea de recepție înainte de a fi opriți din sliding mai departe și primirea mai multor octeți ca un rezultat al unui pachet la începutul ferestrei glisante care nu au fost recunoscuți sau acceptați.
* **TCP Checksum (TCPCS)**: aceasta este o sumă de control care acoperă antetul și porțiunea de date a unui pachet TCP pentru a permite gazdei recepția pentru a verifica integritatea unui pachet TCP de intrare.
* **Urgent Pointer (UP)**: acesta permite unei secțiuni de date să fie specificate de către urgent pointer să fie trimisă de către receiving host rapid.
* **IP Options**: acești biți sunt opționali și sunt folosiți rar.
* **TCP User Data**: această porțiune din pachet poate conține orice număr de protocoale de nivel aplicație (CIPTM, HTTP, SSH, FTP, Telnet, etc.).

1. **Acronime utilizate**

* ADU = Application Data Unit
* ARP = Address Resolution Protocol
* CSMA/CD = Carrier Sense Multiple Access / Collision Detection
* HTTP = Hyper Text Transfer Protocol
* IP = Internet Protocol
* MAC = Medium Access Control or Media Access Control
* MBAP = Modbus Application Protocol
* PDU = Protocol Data Unit
* PHY = Physical Interface
* TCP = Transport Control Protocol or Transmission Control Protocol
* UDP = Universal Datagram Protocol