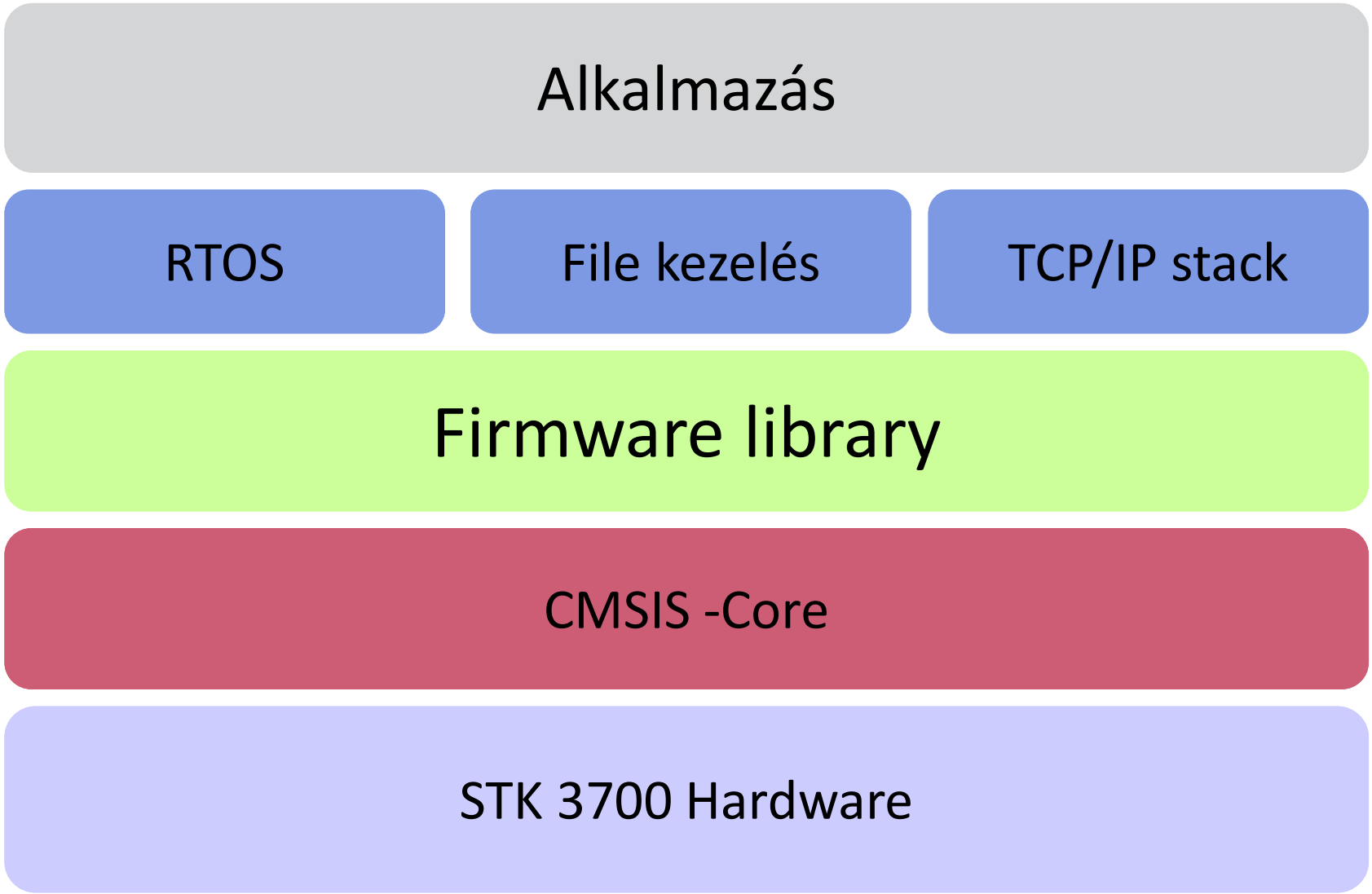
ARM Cortex magú mikrovezérlők

14. Alkalmazás támogatások

Scherer Balázs

Méréstechnika és
Információs Rendszerek
Tanszék

A firmware library-n túli támogatások



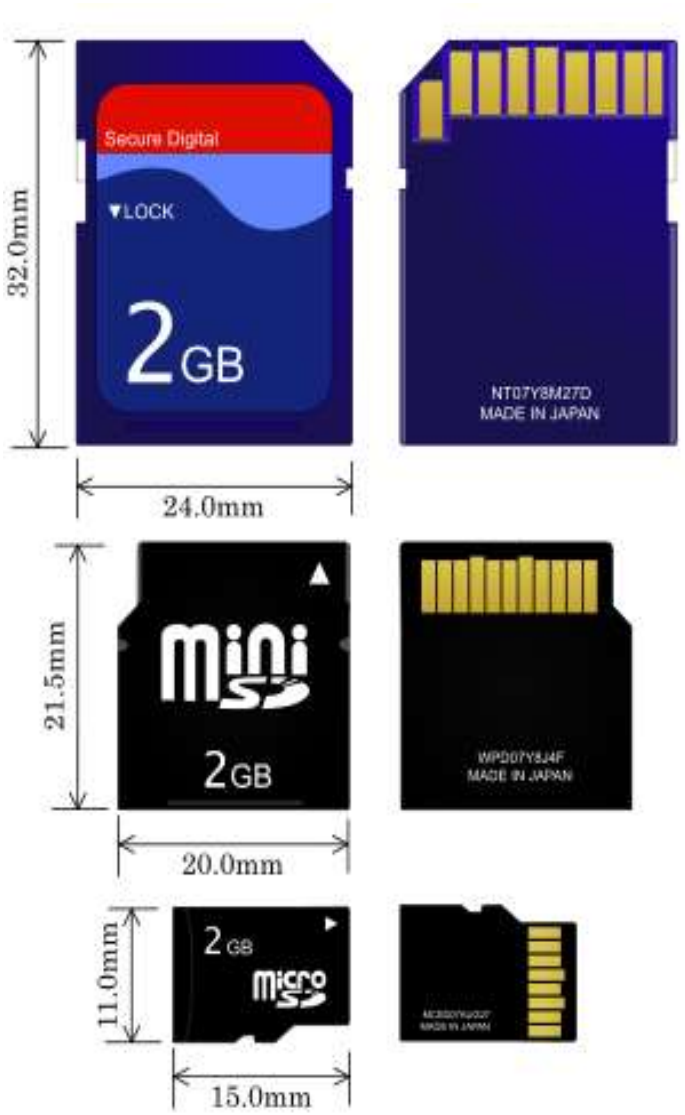
SD kártyák és FAT file rendszerek

SD kártyák megjelenése

- 1999-ben egyezett meg a SanDisk, Matsushita és Toshiba, hogy létrehozza a 24 mm × 32 mm × 2.1 mm méretű SD (Secure Digital) Memory Card-ot.
  - A kártyák 2000 óta elérhetőek a piacon
- A Standard SD kártyáknak max 2 GB-os kapacitása van.
  - Ezt bővítik ki az SDHC (high-capacity) kártyák 4GB feletti méretekre
  - A 2009-es új SDXC (eXtended Capacity) szabvány már 2 TerraB kapacitást is megenged.
- Az egyes szabvány interfészek között vannak különbségek (a fizikai méretek aznososak), amik gondot okozhatnak.

SD kártya típusok

- SD card 32mm x 24mm
- MiniSD card 21,5mm x 20mm
- MicroSD card 15mm x 11mm



SD kártya sebességek

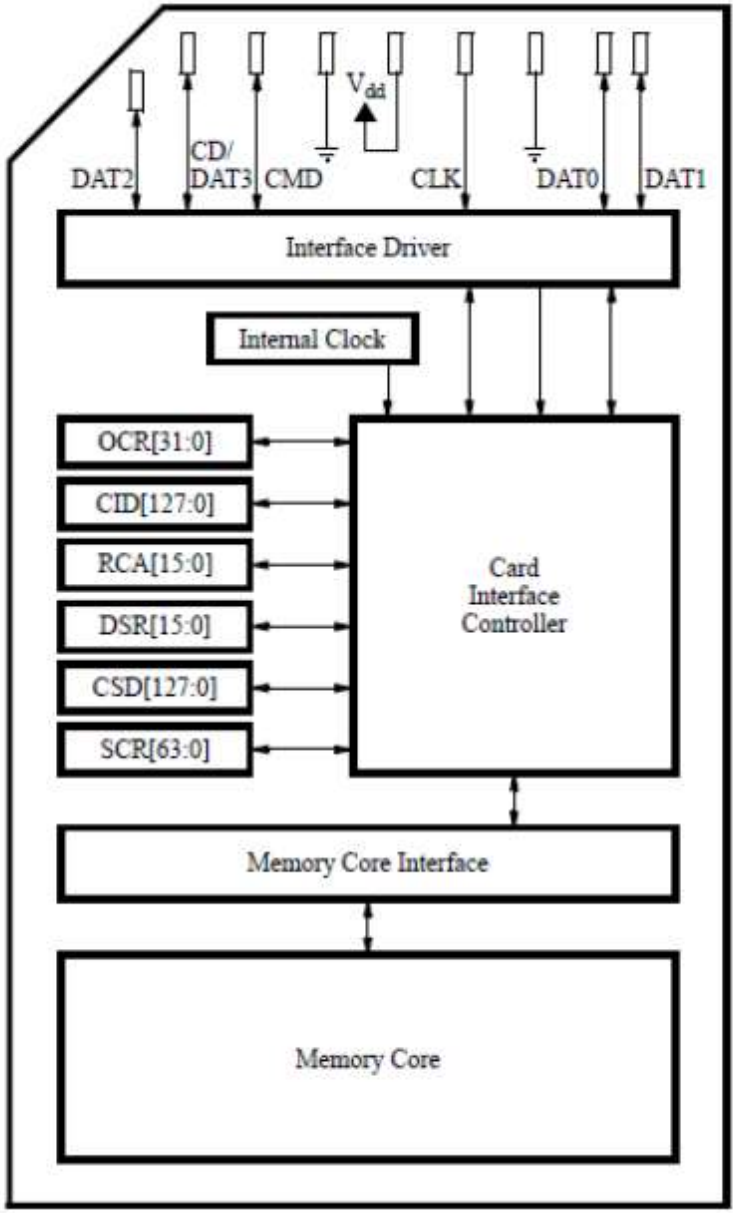
- Az SD kártyák sebességei:
  - Az SD Association által specifikált minimális sebesség 8Mbit/sec
- A jelenleg forgalomba tévő kártyák
  - Class 2: 16 MBit/s (2 MByte/s), 13x
  - Class 4: 32 MBit/s (4 MByte/s), 26x
  - Class 6: 48 MBit/s (6 MByte/s), 40x
  - Class 8: 64 MBit/s (8 MByte/s), 52x
  - Class 10: 80MBit/s (10 Mbyte/s), 60x

A sebességet sokszor "X" rating-ben adják meg, ami a standard CD-ROM 1.2Mbit/s-es sebességéhez viszonyit.

SD belső felépítése

- Az SD kártyák normál esetben hasonlóan a HDD-khez Sector-Block felosztásuak.
  - A block: hány byte írható, olvasható egyszerre a blokkos adatátvitelnél.
    - A tipikus block méret 512byte szokott lenni.
  - A sector pedig azt jelenti, hogy hány blokk törölhető egyszerre.

  Ezek a paraméterek kiolvashatóak az adott kártya információs regiszteréből. A legtöbb kártya a normál data area-n kívül még rendelkezik protected area-val is a bizalmas adatok számára.



SD kártya alapregiszterek

- **OCR (Operation Condition Register):** a kártya működési feszültség tartományát adja meg (tipuksan 2.7V-3.6V)
- **CID (Card Identification Regiszter):** 16byte-os egyedi azonosító, ami a Manufacturer, OEM/Application, Product Name, Product Revision, Serial Number, Manufacture Date Code, CRC7 checksum.
- **Card Specific Data (CSD):** Ez az adatregiszter tartalmaz minden a kártya kezelésével kapcsolatos felhasználói információkat:
  - data read access-time (pl.: 1ms)
  - max. data transfer rate (pl.: 25MHz)
  - max. read data block length (512byte)
  - max. write data block length (512byte)
  - partial blocks for read allowed (Yes)
  - device size
  - max. read current
  - erase single block enable
  - erase sector size (pl.: 32 blocks)
  - write speed factor (pl.: X16)
- **SRC (SD CARD Configuration register):** Ez a regiszter tartalmazza az adott SD kártyára jellemző speciális információkat (általában seccurity supporthoz tartozó dolgok).
- **RCA register:** Címregiszter a kártya azonosítására

SD kártyák kezelése

- Kommunikációs módok
  - **One-bit SD mode:** Különálló parancs és adat csatorna.
  - **Four-bit SD mode:** Extra adatlábak.
  - **SPI mode:** Egyszerűsített kommunikációs elsősorban mikrovezérlők részére.
- Az összes kártyának támogatnia kell ezeket a módokat kivéve a microSD-t ahol az SPI mód opcionális.

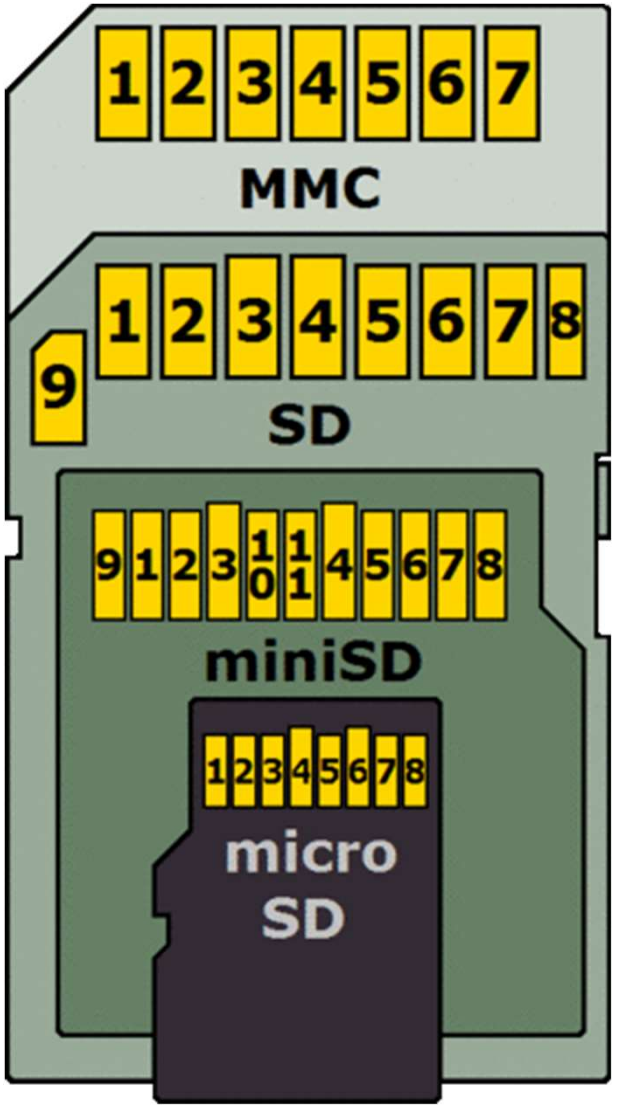
- Az SD kártyák normál esetben hasonlóan a HDD-khez Sector-Block felosztásuak.
  - A block: hány byte írható, olvasható egyszerre a blokkos adatátvitelnél.
    - A tipikus block méret 512byte szokott lenni.
  - A sector pedig azt jelenti, hogy hány blokk törölhető egyszerre.

  Ezek a paraméterek kiolvashatóak az adott kártya információs regiszteréből. A legtöbb kártya a normál data area-n kívül még rendelkezik protected area-val is a bizalmas adatok számára.

SD kártya interfészek

- A vezetékek funkciója függ a felhasznált interfész módjától
- Kommunikációs módok
  - **One-bit SD mode:** Különálló parancs és adat csatorna.
  - **Four-bit SD mode:** Extra adatlábak.
  - **SPI mode:** Egyszerűsített kommunikációs elsősorban mikrovezérlők részére.

Pin	Name	Function (SD Mode)	Function (SPI Mode)
1	DAT3/CS	Data Line 3	Chip Select/Slave Select ($\overline{SS}$)
2	CMD/DI	Command Line	Master Out Slave In (MOSI)
3	VSS1	Ground	Ground
4	VDD	Supply Voltage	Supply Voltage
5	CLK	Clock	Clock (SCK)
6	VSS2	Ground	Ground
7	DAT0/DO	Data Line 0	Master In Slave Out (MISO)
8	DAT1/IRQ	Data Line 1	Unused or IRQ
9	DAT2/NC	Data Line 2	Unused



SPI mód

- **Legegyszerűbb átvitel**
  - Egyszerűbb mikrovezérlőkre jellemző
  - Korlátozott sebesség 1bit adat
  - STM32 SPI SCK frekvencia max. 18MHz

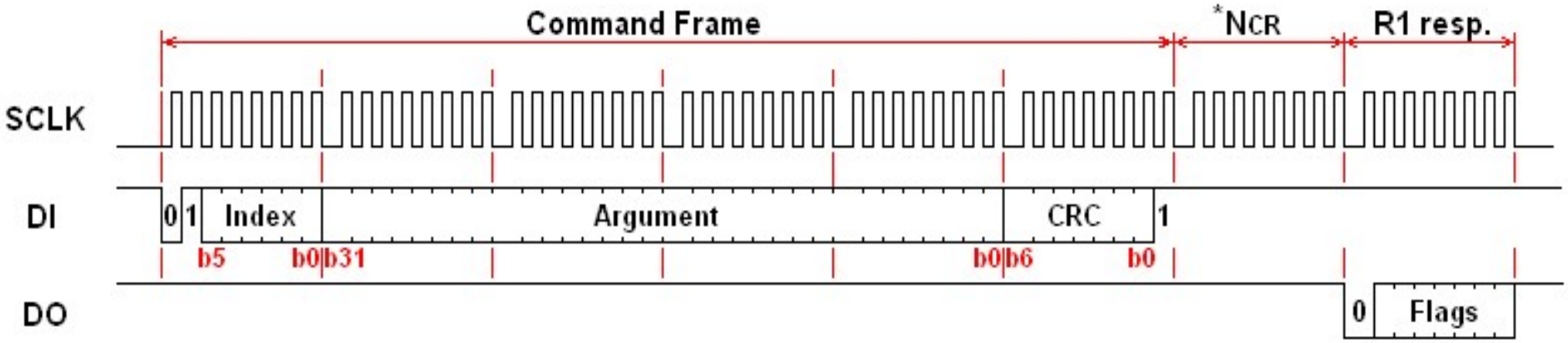
Table 3-2. SPI Bus Mode Pad Definition

Pin #	Name	Type[1]	SPI Description
1	CS	I	Chip Select (Active low)
2	DataIn	I	Host to Card Commands and Data
3	VSS1	S	Supply Voltage Ground
4	VDD	S	Supply Voltage
5	CLK	I	Clock
6	VSS2	S	Supply Voltage Ground
7	DataOut	O	Card to Host Data and Status
8	RSV(2)	I	Reserved
9	RSV(2)	I	Reserved

SPI kommunikációs frame

Az SPI kommunikációs frame 6 byte-ból áll. A CRC bekapcsolható és kikapcsolható.

Byte 1			Bytes 2—5	Byte 6	
7	6	5 0	31 0	7	0
0	1	Command	Command Argument	CRC	1



SPI kommunikációs parancsok

A parancsok a parancs kód +0x40 értékkel küldődnek el az SPI buszon.

Alap parancsok:

CMD INDEX	SPI Mode	Argument	Resp	Abbreviation	Command Description
CMD0	Yes	None	R1	GO_IDLE_STATE	Resets the SD Card
CMD1	Yes	None	R1	SEND_OP_COND	Activates the card's initialization process.

Identifikációs parancsok:

CMD9	Yes	None	R1	SEND_CSD	Asks the selected card to send its card-specific data (CSD).
CMD10	Yes	None	R1	SEND_CID	Asks the selected card to send its card identification (CID).

Adatátvitel leállítás és státusz parancsok:

CMD12	Yes	None	R1b	STOP _TRANSMISSION	Forces the card to stop transmission during a multiple block read operation.
CMD13	Yes	None	R2	SEND_STATUS	Asks the selected card to send its status register.

SPI kommunikációs parancsok

A parancsok a parancs kód +0x40 értékkel küldődnek el az SPI buszon.

Olvasó parancsok

CMD16	Yes	[31:0] block length	R1	SET_BLOCKLEN	Selects a block length (in bytes) for all following block commands (read & write).[1]
CMD17	Yes	[31:0] data address	R1	READ_SINGLE _BLOCK	Reads a block of the size selected by the SET_BLOCKLEN command.[2]
CMD18	Yes	[31:0] data address	R1	READ_MULTIPLE _BLOCK	Continuously transfers data blocks from card to host until interrupted by a STOP_ TRANSMISSION command.

Író parancsok

CMD24	Yes	[31:0] data address	R1[3]	WRITE_BLOCK	Writes a block of the size selected by the SET_BLOCKLEN command.[4]
CMD25	Yes	[31:0] data address	R1	WRITE_MULTIPLE_BLOCK	Continuously writes blocks of data until a stop transmission token is sent (instead of 'start block').

SPI kommunikációs parancsok

A parancsok a parancs kód +0x40 értékkel küldődnek el az SPI buszon.

Törlést kijelölő parancsok

CMD32	Yes	[31:0] data address	R1	ERASE_WR_BLK_START _ADDR	Sets the address of the first write block to be erased.
CMD33	Yes	[31:0] data address	R1	ERASE_WR_BLK_END _ADDR	Sets the address of the last write block in a continuous range to be erased.

Törlő parancsok

CMD38	Yes	[31:0] don't care*	R1b	ERASE	Erases all previously selected write blocks.

SPI kommunikációs parancsok

A parancsok a parancs kód +0x40 értékkel küldődnek el az SPI buszon.

Speciális parancsok

CMD27	Yes	None	R1	PROGRAM_CSD	Programming of the programmable bits of the CSD.
CMD28[1]	Yes	[31:0] data address	R1b	SET_WRITE_PROT	If the card has write protection features, this command sets the write protection bit of the addressed group. The properties of write protection are coded in the card specific data (WP_GRP_SIZE).
CMD29[4]	Yes	[31:0] data address	R1b	CLR_WRITE_PROT	If the card has write protection features, this command clears the write protection bit of the addressed group.
CMD30	Yes	[31:0] write protect data address	R1	SEND_WRITE_ PROT	If the card has write protection features, this command asks the card to send the status of the write protection bits.[2]

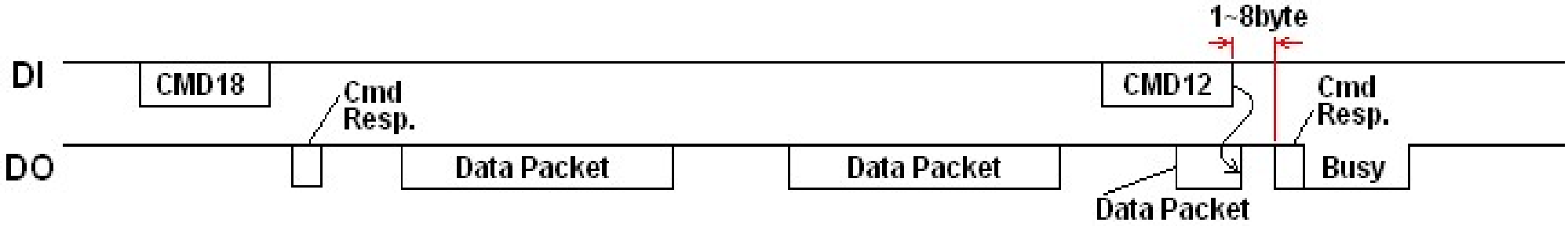
SPI mód inicializáció

- Az SD kártyák alapvetően SD módban indulnak el
  - először is SPI módba kell azokat rakni
  - Reszet parancs alatt a CS lábat le kell húzni.
  - Bár az SPI módban a CRC védelem ki van kapcsolva, az első reszet parancsnál még SD módban van a kártya tehát érvényes CRC mezőt kell átküldeni
    - (mivel a parancsnak nincs aktívan változó paramétere, ezért az egész reszet parancs kezelhető egy byte-sorozatnak 0x40, 0x0, 0x0, 0x0, 0x0, 0x95).
  - Ennél a parancsnál az SPI frekvencia nem haladhatja meg a 400kHz-et.

Olvasás SPI módban

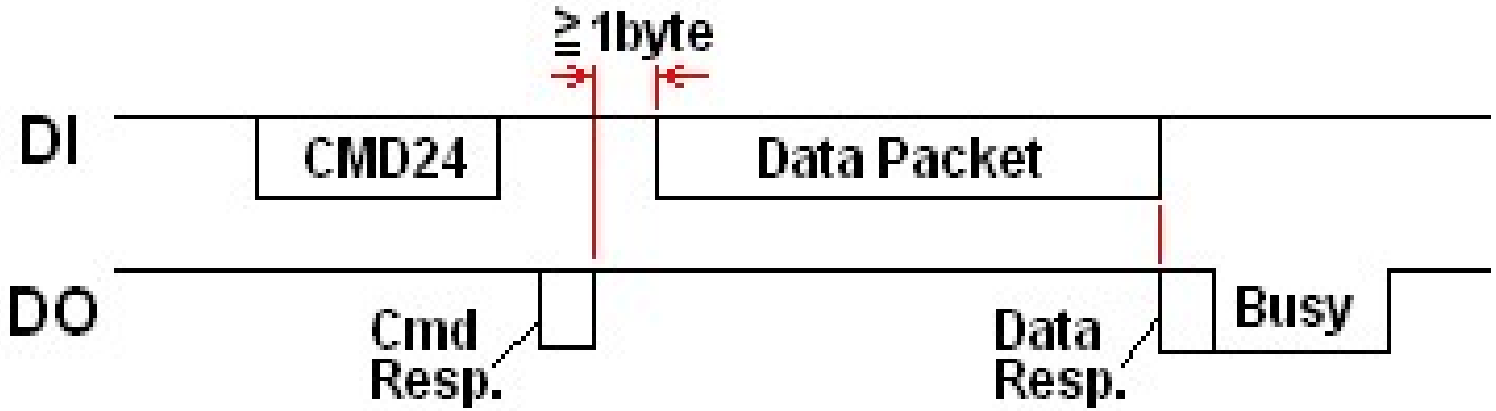
Az SPI mód egy blokk (CMD17) és több blokk (CMD18) egyszerre való olvasását is támogatja. A Data packetben lévő CRC-t mindenképpen ki kell olvasni, akkor is ha nem használjuk.

Több blokk esetében a Stop transmission paranccsal (CMD12) le lehet állítani az adatátvitelt.



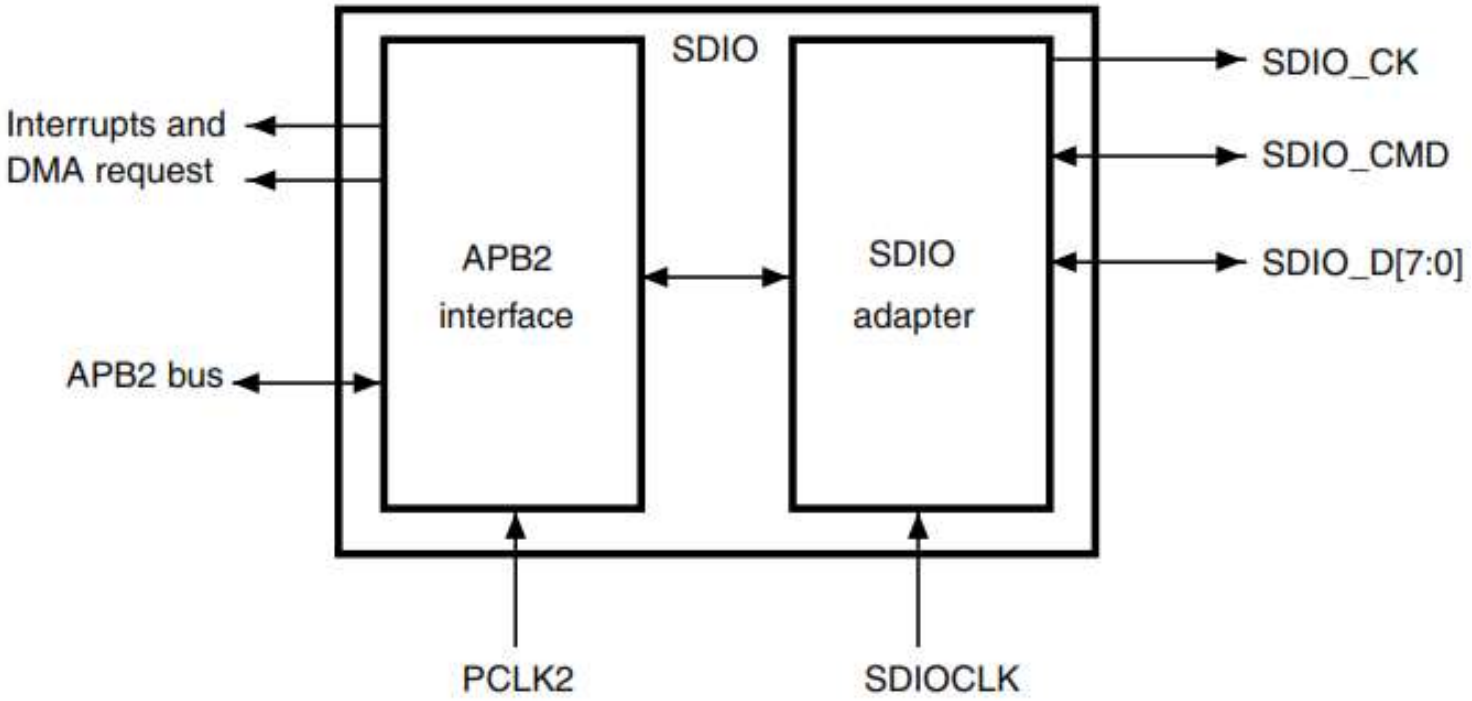
Írás SPI módban

Az SPI blokk támogatja az egy és a több blokkos írást is (CMD24, CMD25). Mindegyik blokk írás egy egy byte-os Start Block tokennel indul. Az adat megérkezése után az SD kártya küld egy data_response tokent, majd amíg az adatokat ténylegesen kiírja a Flash memóriába folyamatosan busy tokent küld a buszra (lent tartja DataOut lábát). Az írás véget érését vagy ennek a busy jelenek a figyelésével, vagy a státusz információ kiolvasásával (CMD13) tudja a hosszt megállapítani.

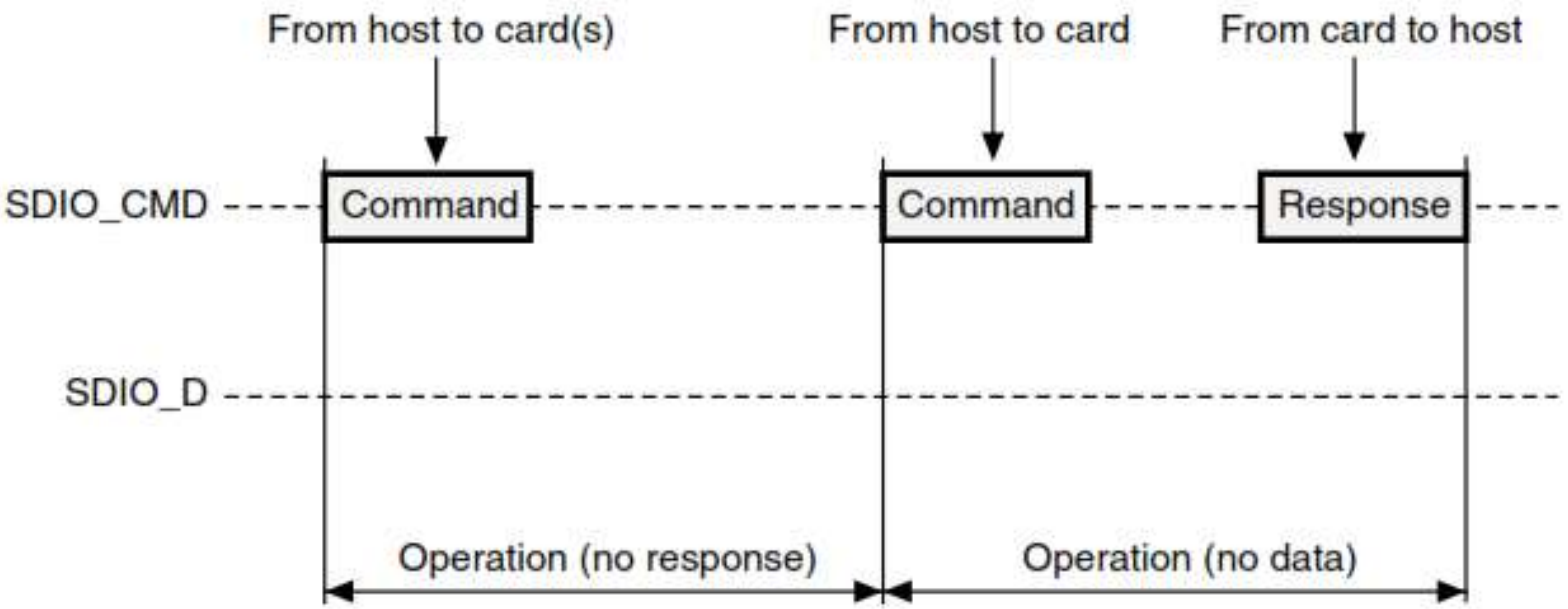


SD mód

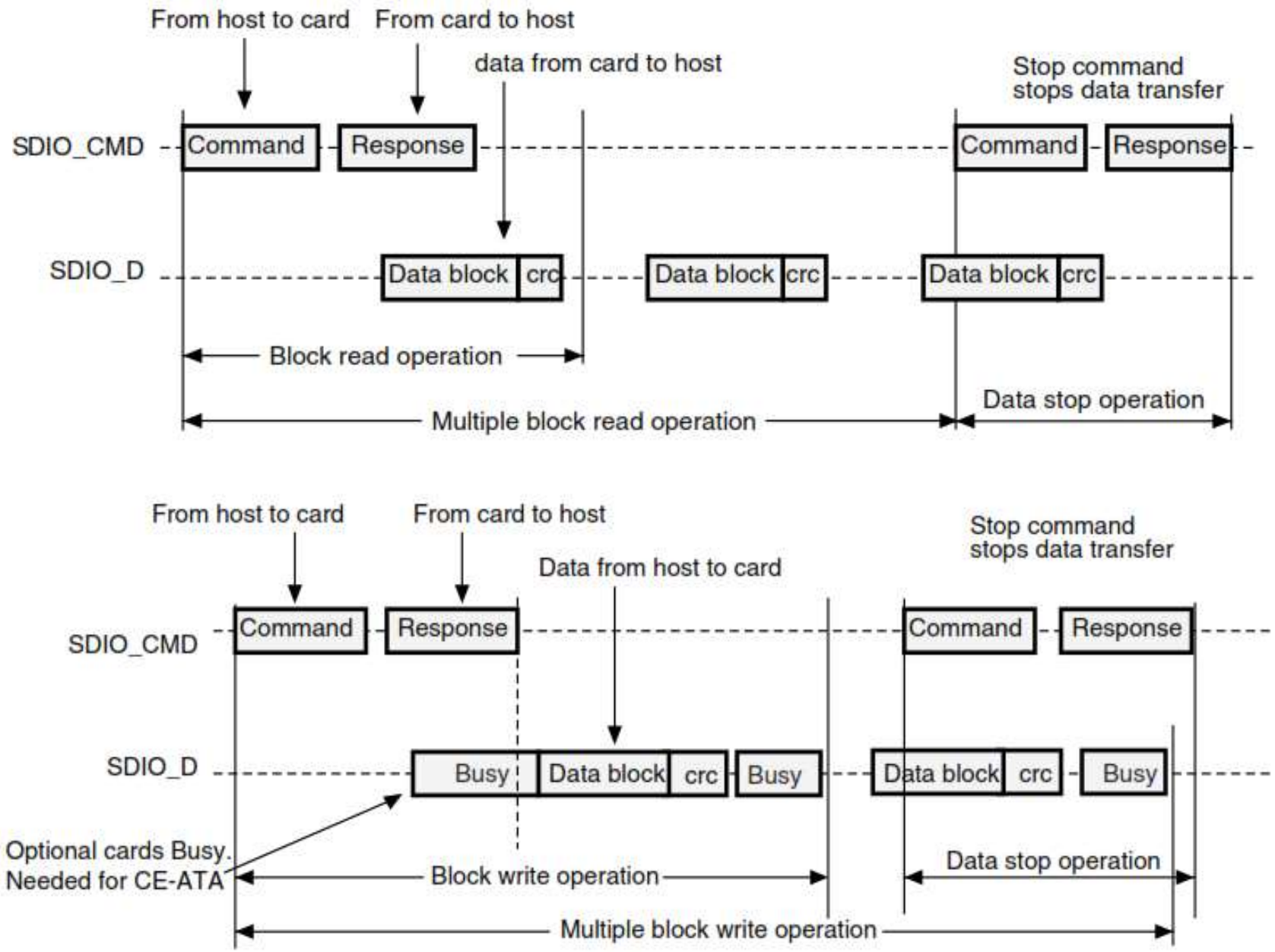
- **Csak akkor ha a hardware támogatja**
  - **Külön CMD vezeték: kétirányú timeout alapú állapotgép**
  - **Data: 4 biten SD kártyára először a felső 4 bit utána az alsó**
  - **Nagy sebességű órajel pl. STM32F429 max. 20MHz**



SD mód parancs kommunikáció



SD mód blokk olvasás és blokk írás



FAT file rendszer

- A FAT első verziója a FAT12-t (32 Mbyte) 1980-ban fejlesztették ki floppy lemezek számára.
- A következő verzió a FAT16-volt (2 Gbyte), ami 1987-ben látott napvilágot.
- Az utolsó a FAT32-volt, ami 1996-ban jelent meg (2 Tbyte), a Windows-on a SCANDISK alkalmazás 16 bites szektor számlálója miatt volt egy kb. 130 Gbyte-os határ.

FAT file rendszer felépítése

Contents	Boot Sector	FS Information Sector (FAT32 only)	More reserved sectors (optional)	File Allocation Table #1	File Allocation Table #2	Root Directory (FAT12/16 only)	Data Region (for files and directories) … (To end of partition or disk)
Size in sectors	(number of reserved sectors)			(number of FATs)* (sectors per FAT)		(number of root entries*32)/Bytes per sector	NumberOfClusters*SectorsPerCluster

- **Boot Sector:** (Particion Boot Record):
  - általában az operációs rendszer bootload-erjét tartalmazza.
  - A lefoglalt terület mérete a Boot sector egy mezője által azonosítódik.
  - Nem mindig a boot sector az első szektor a disk-en.
  - Particionált egységeknél az első szektor a master boot record, nem particionált egységeknél a Volume boot record.
  - Az első 36 byte struktúrája minden FAT file rendszer esetében azonos
    - Ez tartalmaz egy jump vetort ha innen indulunk az itt található címre ugrik a programvégrehajtás. Tartalmazza még az OEM nevét (mire formázták) a szektoronkénti byte-ok számát (ált 512), a clusterenkénti sector-ok számát (2 1-128-ig terjedő hatványa, max 32k byte/ cluster), valamint az összes sector számát.

FAT file rendszer felépítése

Contents	Boot Sector	FS Information Sector (FAT32 only)	More reserved sectors (optional)	File Allocation Table #1	File Allocation Table #2	Root Directory (FAT12/16 only)	Data Region (for files and directories) … (To end of partition or disk)
Size in sectors	(number of reserved sectors)			(number of FATs)* (sectors per FAT)		(number of root entries*32)/Bytes per sector	NumberOfClusters*SectorsPerCluster

- **FS Information sector (csak FAT32):** A FAT32-ben mutatták be elsősorban a szabad terület gyors nyilvántartására.
- **FAT (File Allocation Table):** A partíciók egyenlő méretű clusterekre vannak bonva, a cluster méret függhet az alkalmazott FAT file rendszertől és a partíció méretétől. Általában a 2k és a 32k közötti méreteket preferálják. Minden file méretétől függően egy, vagy több ilyen clustert foglal el. Egy file a clusterek láncolt listájával megadható, bár sokszor még az egy file hoz tartozó clusterek se mindig egymás mellett találhatóak: fragmentálódik a file.

FAT tábla

A FAT egy leíró lista, amely egy térképet ad a partícióban található clusterek-hez a FAT16 esetében 16 bit, a FAT32 esetében a leíró tábla minden egyes sector-hoz 32bitet tartalmaz (a FAT tábla mérete függ a sectorok számától). A leíró lista az alábbi bejegyzéseket tartalmazhatja:

- A következő cluster száma
- Speciális *end of clusterchain* (*EOC*) jelzés a lánc végén, a file utolsó clustere.
- Speciális a bad cluster jelzés
- Speciális jelzés a reserved cluster számára
- A 0 ha a cluster nem használt.

FAT12	FAT16	FAT32	Description
0x000	0x0000	0x00000000	Free Cluster
0x001	0x0001	0x00000001	Reserved value; do not use
0x002–0xFEF	0x0002–0xFFEF	0x00000002–0x0FFFFFEF	Used cluster; value points to next cluster
0xFF0–0xFF6	0xFFF0–0xFFF6	0x0FFFFFF0–0x0FFFFFF6	Reserved values; do not use[30].
0xFF7	0xFFF7	0x0FFFFFF7	Bad sector in cluster or reserved cluster
0xFF8–0xFFF	0xFFF8–0xFFFF	0x0FFFFFF8–0x0FFFFFFF	Last cluster in file

Directory tábla

A directory table egy speciális file. Minden directory, vagy file, ami benne van egy 32byte-os blokkal azonosítódik. Mindegyik blokk tartalmazza a következőket:

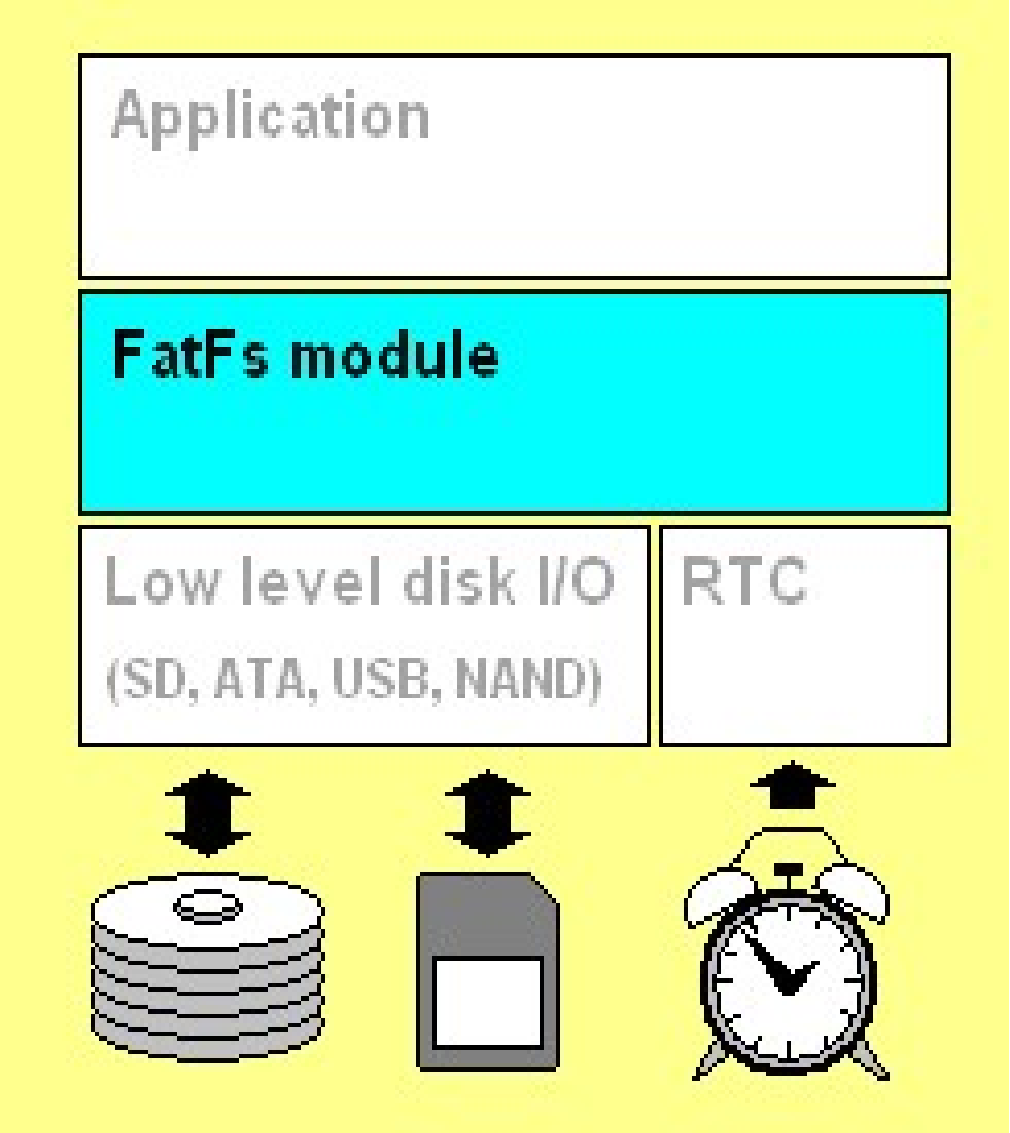
- Nevet
- Kiterjesztést
- A file tulajdonságát: read only hidden stb
- A létrehozás idejét.
- Az utolsó hozzáférés idejét
- A file, vagy directory első clusterének címét
- A file méretét byte-okban.

A FAT16 és a FAT12 esetében van egy kitüntettet root directory rész, míg a FAT32 estében minden directory a DATA részben tárolódik. A hosszú file nevek egy trükkel plusz bejegyzésként adódnak hozzá minden egyes file bejegyzés elején.

Chan FatFS

- Kifejezetten beágyazott rendszerekhez létrehozott FAT file-rendszer:
- ANSI C-ben íródott
- Platform független
- FAT sub-types: FAT12, FAT16 and FAT32.
- Nyitott file-ok száma: Nem limitált a rendelkezésre álló memóriától függ
- File méret: A FAT specifikációtól függően 4G-1 bytes.
- Cluster méret: 64kbytes, vagy 32kbytes.
- Sector méret: FAT specifikáció függő (max. 4K bytes).

Chan FatFS felépítés



Chan FatFS portolás

Function	Required when:
disk_initialize	Always
disk_status	Always
disk_read	Always
disk_write	_FS_READONLY == 0
disk_ioctl (CTRL_SYNC)	_FS_READONLY == 0
disk_ioctl (GET_SECTOR_COUNT)	_USE_MKFS == 1
disk_ioctl (GET_SECTOR_SIZE)	_MAX_SS >= 1024
disk_ioctl (GET_BLOCK_SIZE)	_USE_MKFS == 1
get_fattime	_FS_READONLY == 0

- disk_initialize – Disk drive inicializáció
- disk_status – Disk drive státusz lekérdezés
- disk_read – Sector olvasás
- disk_write - Sector írás
- disk_ioctl – Disk specifikus tulajdonságok
- get_fattime – Rendszer idő (naptári óra) lekérdezés

Chan FatFS felhasználói interfész

- f_mount - Register/Unregister a Work Area
- f_open - Open/Create a File
- f_close - Close a File
- f_read - Read File
- f_write - Write File
- f_lseek - Move R/W Pointer, Expand File Size
- f_truncate - Truncate File Size
- f_sync - Flush Cached Data
- f_opendir - Open a Directory
- f_readdir - Read a Directory Item
- f_getfree - Get Free Clusters
- f_stat - Get File Status

Chan FatFS felhasználói interfész

- f_mkdir - Create a Directory
- f_unlink - Remove a File or Directory
- f_chmod - Change Attribute
- f_utime - Change Timestamp
- f_rename - Rename/Move a File or Directory
- f_mkfs - Create a File System on the Drive
- f_forward - Forward file data to the stream directly
- f_chdir - Change current directory
- f_chdrive - Change current drive
- f_gets - Read a string
- f_putc - Write a character
- f_puts - Write a string
- f_printf - Write a formatted string

Chan FatFS méret

Memory Usage (R0.07e)

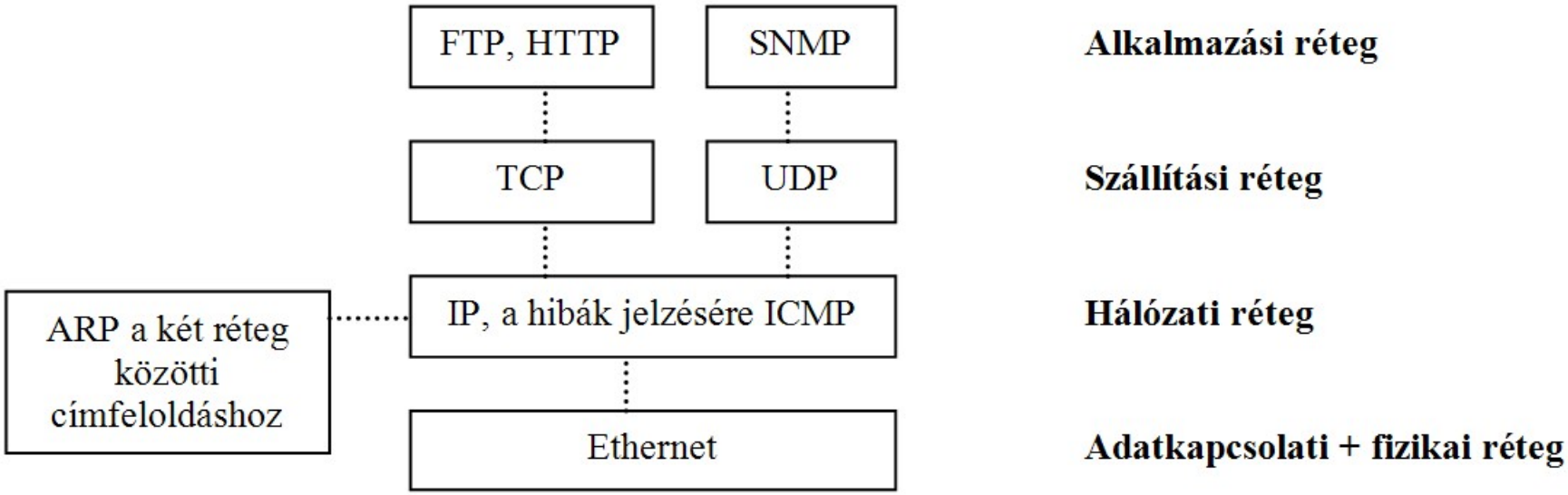
	AVR	H8/300H	PIC	TLCS-870/C	V850ES	SH2	ARM7TDMI	x86
Compiler	WinAVR(gcc)	CH38	C30(gcc)	CC870C	CA850	SHC	WinARM(gcc)	VC6
_WORD_ACCESS	1	0	0	1	1	0	0	1
text (Full, R/W)	12194	10559	10924	15229	7686	8727	10564	7342
text (Min, R/W)	7988	6903	7108	9960	4884	5651	6544	4764
text (Full, R/O)	5532	4753	5020	6760	3462	3777	4624	3316
text (Min, R/O)	4040	3631	3736	5083	2556	2907	3284	2510
bss	D*2 + 2	D*4 + 2	D*2 + 2	D*2 + 2	D*4 + 2	D*4 + 2	D*4 + 2	D*4 + 2
Work area (_FS_TINY == 0)	D*560 + F*544	D*560 + F*550	D*560 + F*544		D*560 + F*550	D*560 + F*550	D*560 + F*550	D*560 + F*550
Work area (_FS_TINY == 1)	D*560 + F*32	D*560 + F*36	D*560 + F*32	D*560 + F*32	D*560 + F*36	D*560 + F*36	D*560 + F*36	D*560 + F*36

- D: a kötetek száma
- F: nyitott file-ok száma

TCP/IP protocol stack

Alap beágyazott szoftver architektúrák I.

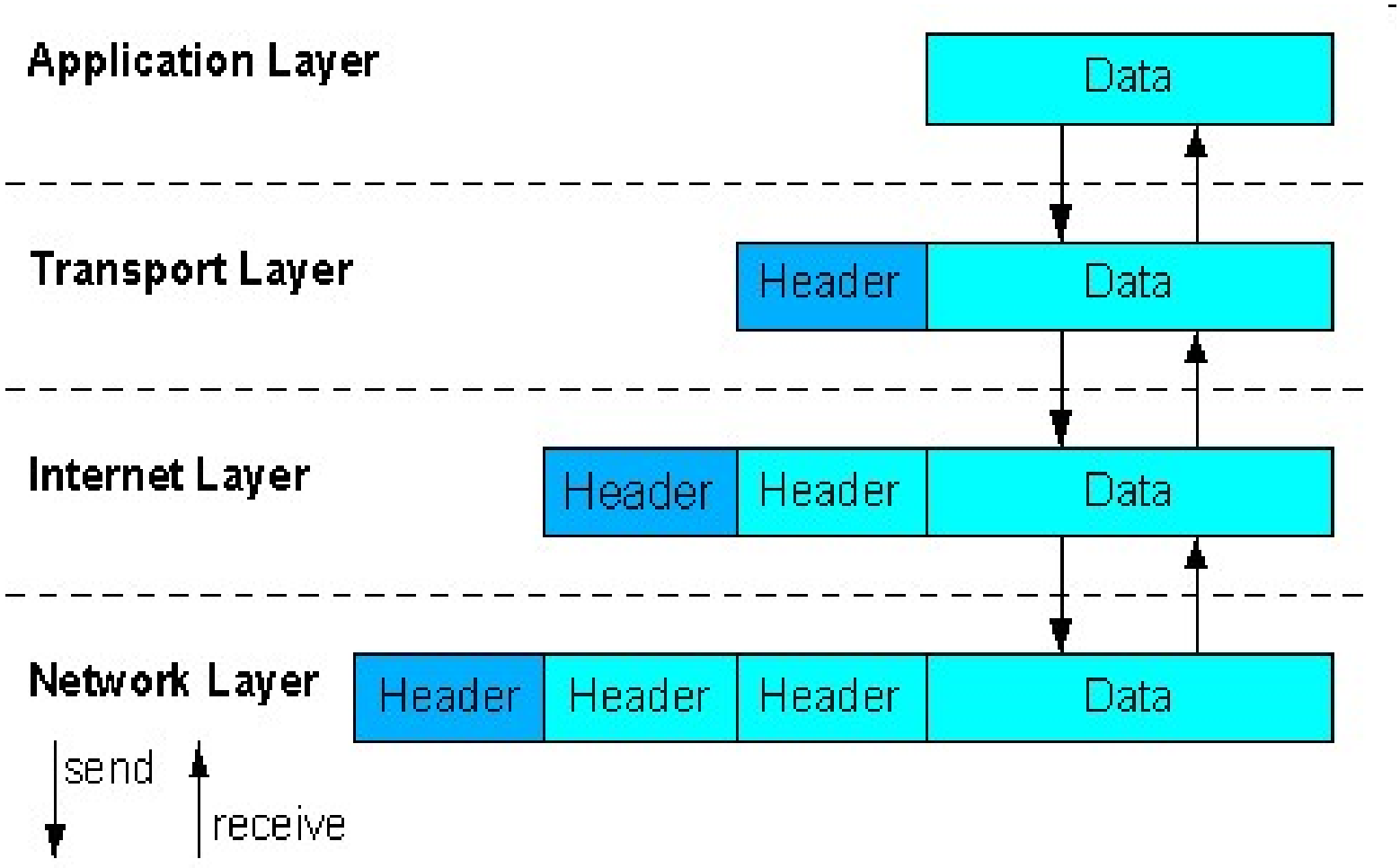
- Szükséges TCP/IP protokollok



IP : Internet Protocol
ICMP : Internet Control Message Protocol
ARP : Address Resolution Protocol
TCP : Transmission Control Protocol

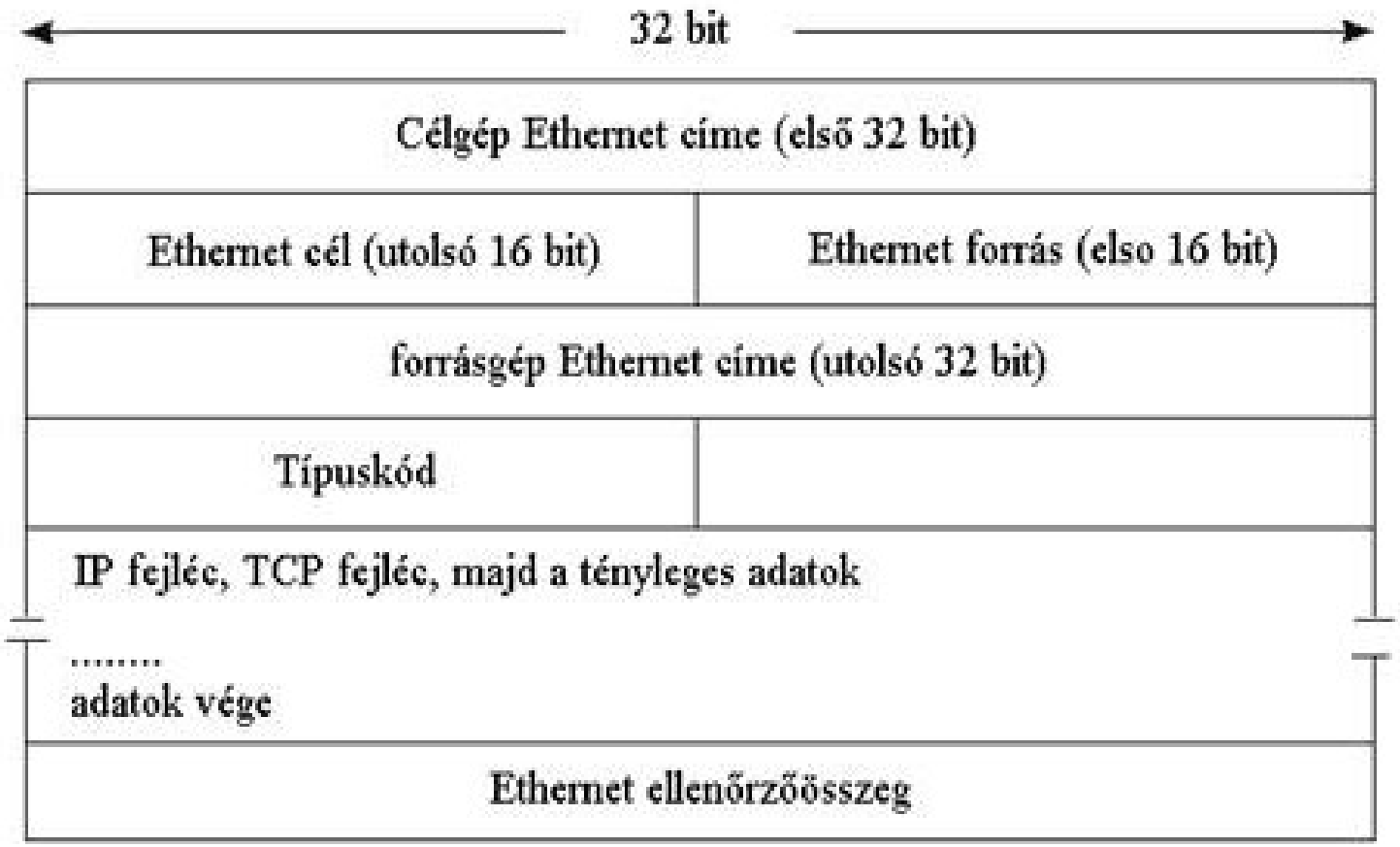
UDP : User Datagram Protocol
FTP : File Transfer Protocol
HTTP : Hyper Text Transfer Protocol
SNMP : Simple Network Management Protocol

Keretezés a hálózaton



Ethernet

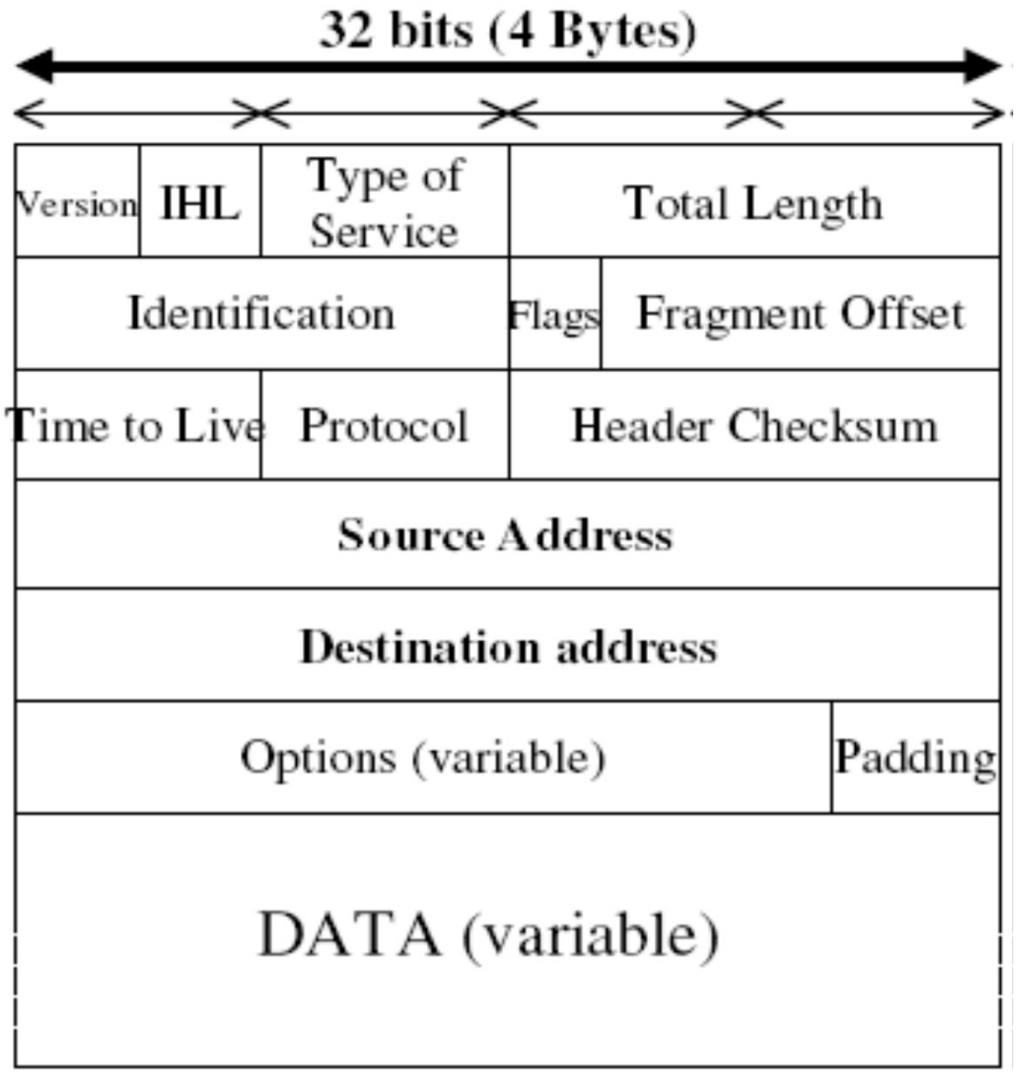
- Típuskód: *TCP/IP, DECnet…*



IP (Internet Protocol)

- Csomagokat továbbít, darabol és összerak
- Megbízhatatlan kapcsolatmentes datagramm szolgáltatást nyújt.
- Nincs garancia a sikeres célba érésre
- Ha bármilyen hiba lép fel: eldobja a datagrammot
- A datagrammok sorrendje megváltozhat

IPv4 (Internet Protocol v4)



Kérdések az IP protokoll megvalósítással szemben

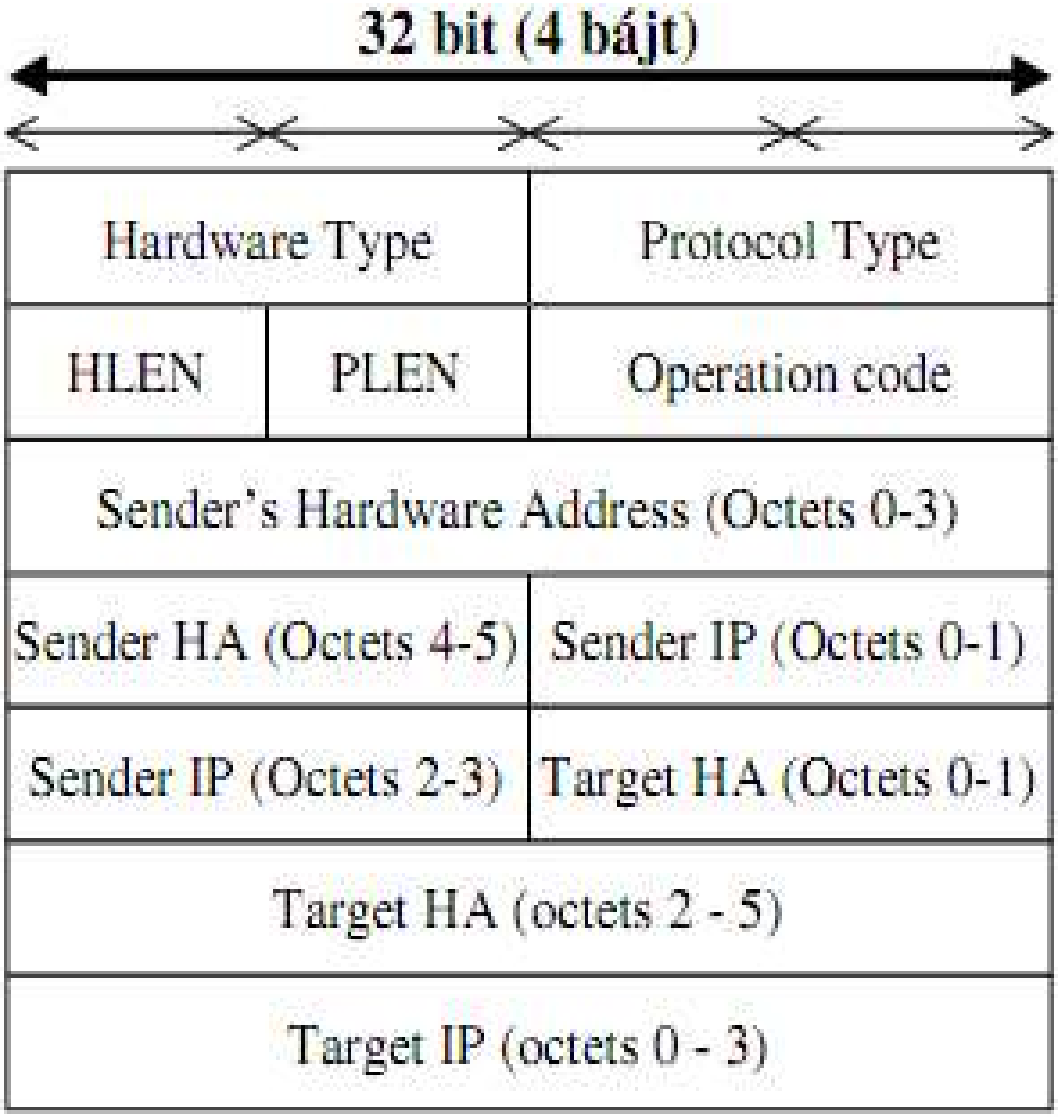
- Option küldés
  - Statisztikai jellegű feladatok:
    - Milyen csomópontokon keresztül jutott el a csomag a célhoz
  - Kis méretű protokollstack-ek nem tudják kezelni
- Fragmentáció
  - Elméletileg mindenkinek tudnia kell
  - Gyakorlatilag ritkán használt és sok memória kell hozzá
- 576byte-os csomag fogadása
  - Elméletileg kötelező gyakorlatilag nem biztos, hogy működik

ARP (Address Resolution Protocol)

- IP üzenetek küldéséhez ismerni kell a célállomás IP címét és a fizikai Ethernet címét is
- Az ARP segítségével az IP cím használatával meg lehet határozni a cél fizikai (Ethernet) címét
- Az ARP Ethernet broadcast-ot használ
- A feloldott Ethernet – IP cím párosokat egy helyi tárban tárolja az ARP

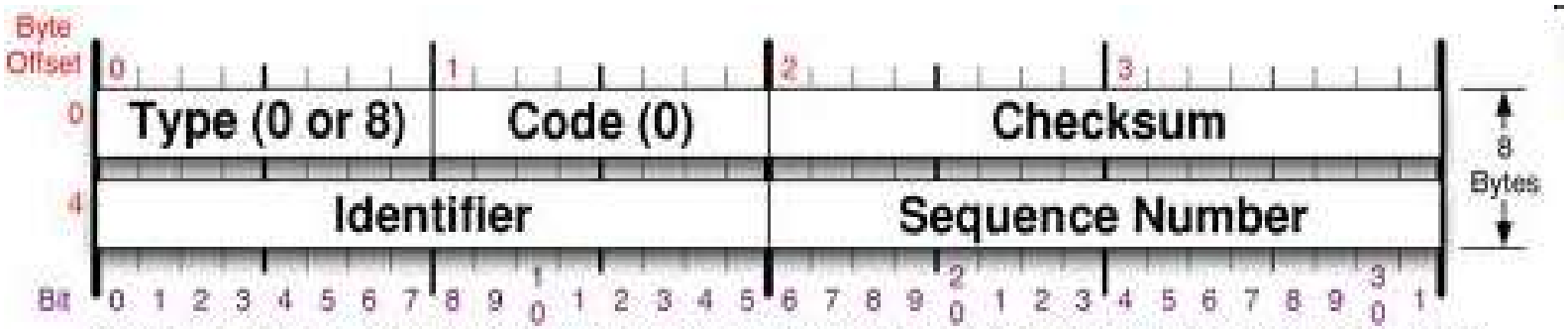
Az ARP fejléc

- Hardware Type:
  - Ethernet: 1
- Protocol Type:
  - Ethernet: 0x0800
- HLEN, PLEN
  - Címek hossza byteokban
- Operation code
  - Request
  - Reply



ICMP (Internet Control Message Protocol)

- Echo, and Echo Reply
- Destination unreachable
- Traceroute
- TTL exceeded



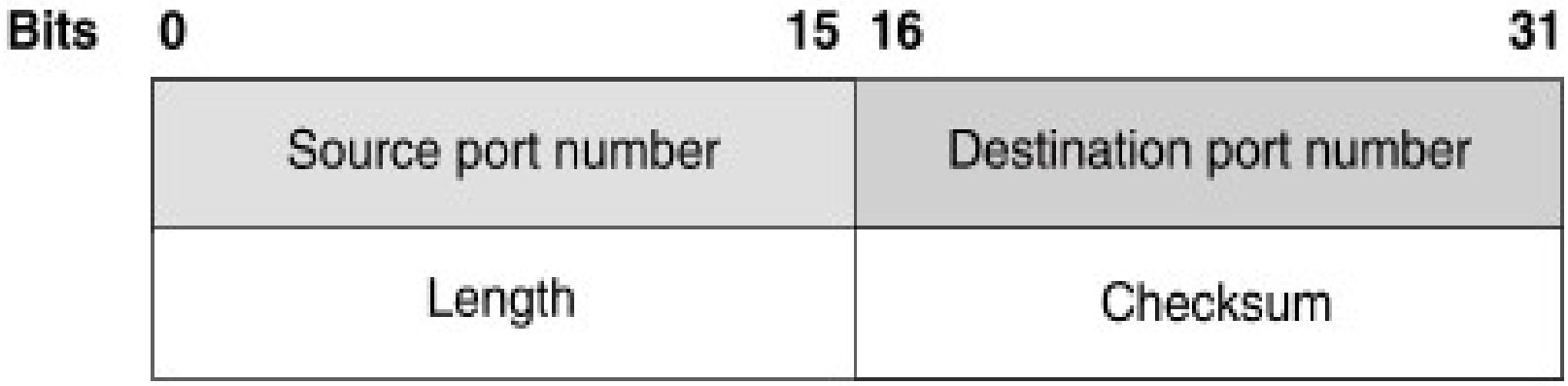
ICMP megvalósítási kérdések

- Echo küldés, válasz
  - Mindig implementált
- Destination Unreachable …:
  - Ezeket már nem mindig szokták implementálni

UDP (User Datagram protocol)

- Megbízhatatlan üzenettovábbítás
- Nem kapcsolat alapú
- Üzenet szórásra alkalmas
- Rendkívül kis erőforrás igényű

UDP (User Datagram protocol)

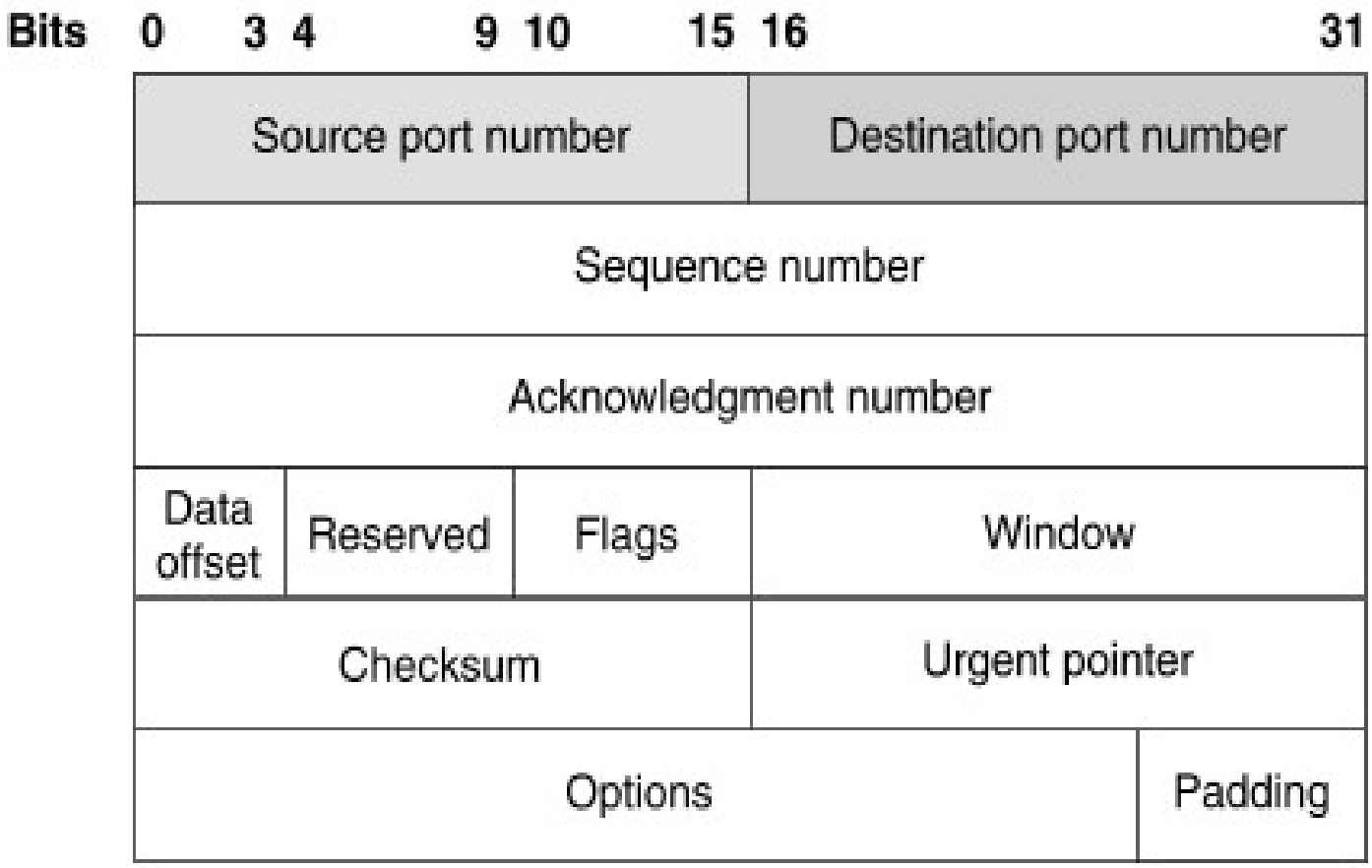


- Checksum mező
  - Opcionális a használata, nem mindig töltik ki.
    - Ethernet kereten úgyis van CRC ellenörzés

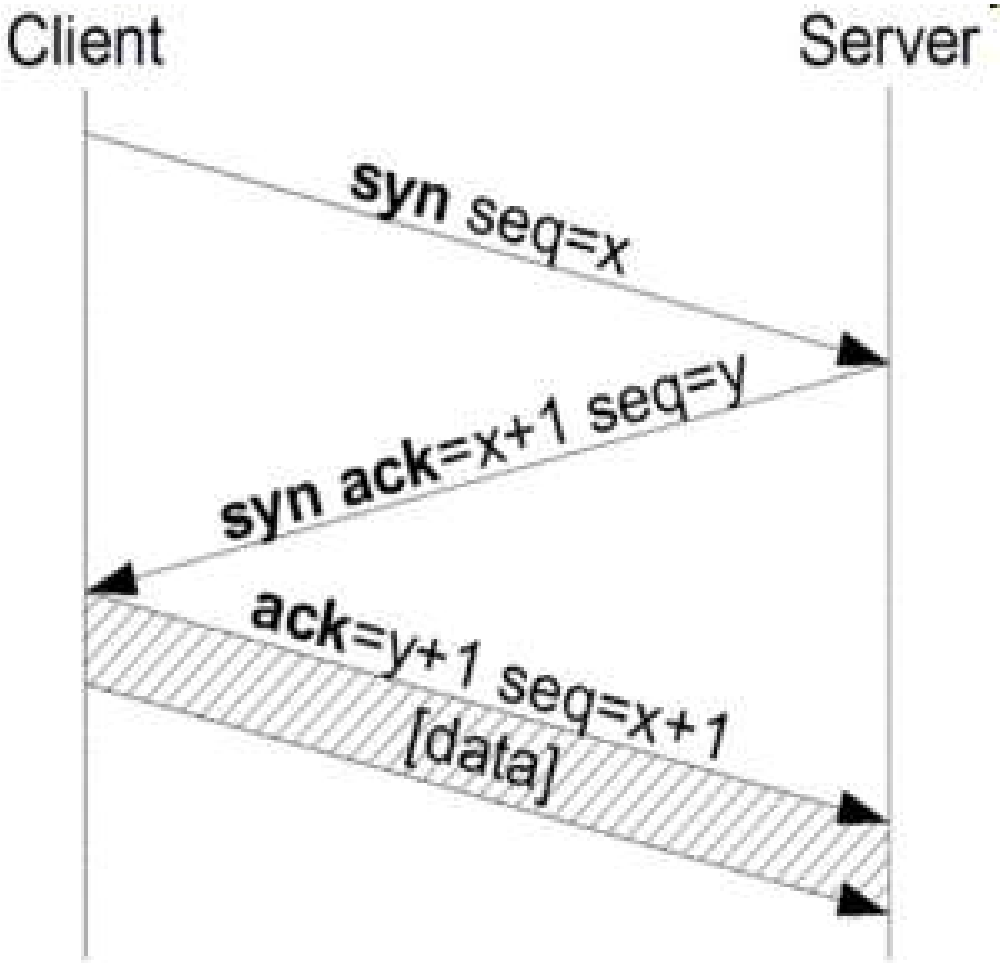
TCP (Transmission Control Protocol)

- Kapcsolat alapú megbízható adatátvitel
- Nem alkalmazható üzenetszórásra
- Az adatfolyamot részekre bontja
  - Jelentős helyi tárkapacitást igényel a szegmensek összerakásához

TCP (Transmission Control Protocol)



TCP (Transmission Control Protocol)



Web szerver alapú vezérlés

- TCP alapú, tehát nagy erőforrás
- HTML lapok nem túl erőforrás takarékosok
- HTML lapok tárolási módja
  - Flash, be statikusan
  - Dinamikusan valahol
- Web browserben elküldött
  - Post, get parancsokra reagállás

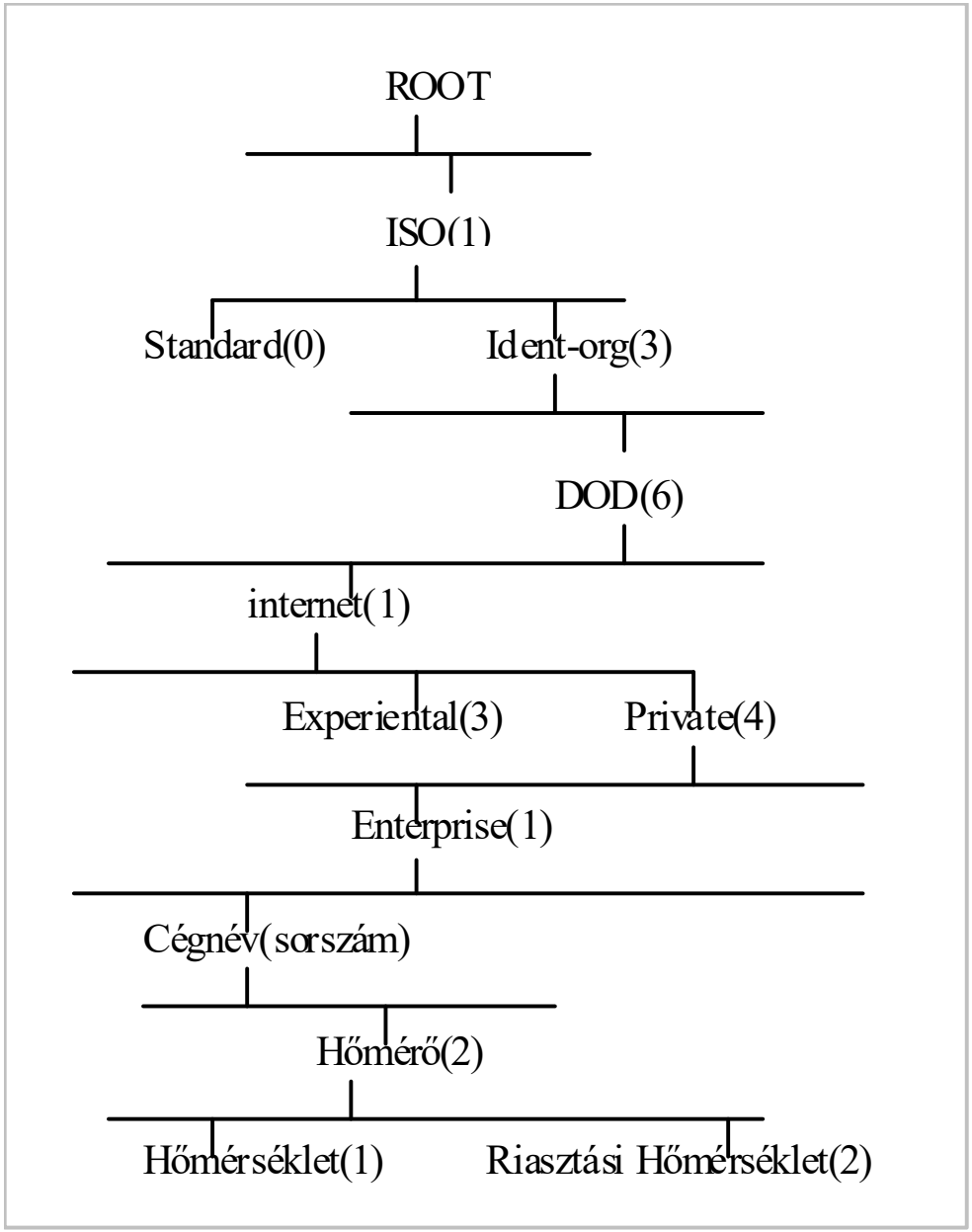
FTP alapú megoldás

- Viszonylag erőforrás igényes, TCP alapú
- Szenzor kezelés nem túl hatékony
- Nehezen konfigurálható
- Nagy mennyiségű adat átvitelére alkalmas

SNMP

- Erőforrás takarékos, UDP
- Jól használható sensor kezelésre, konfigurációra
- MIB-ek kezelése közepesen bonyolult
- Az ID-k kezelése nagyon szószátyár
- Nem hatékony nagymennyiségű információ átvitelére

SNMP MIB struktúra

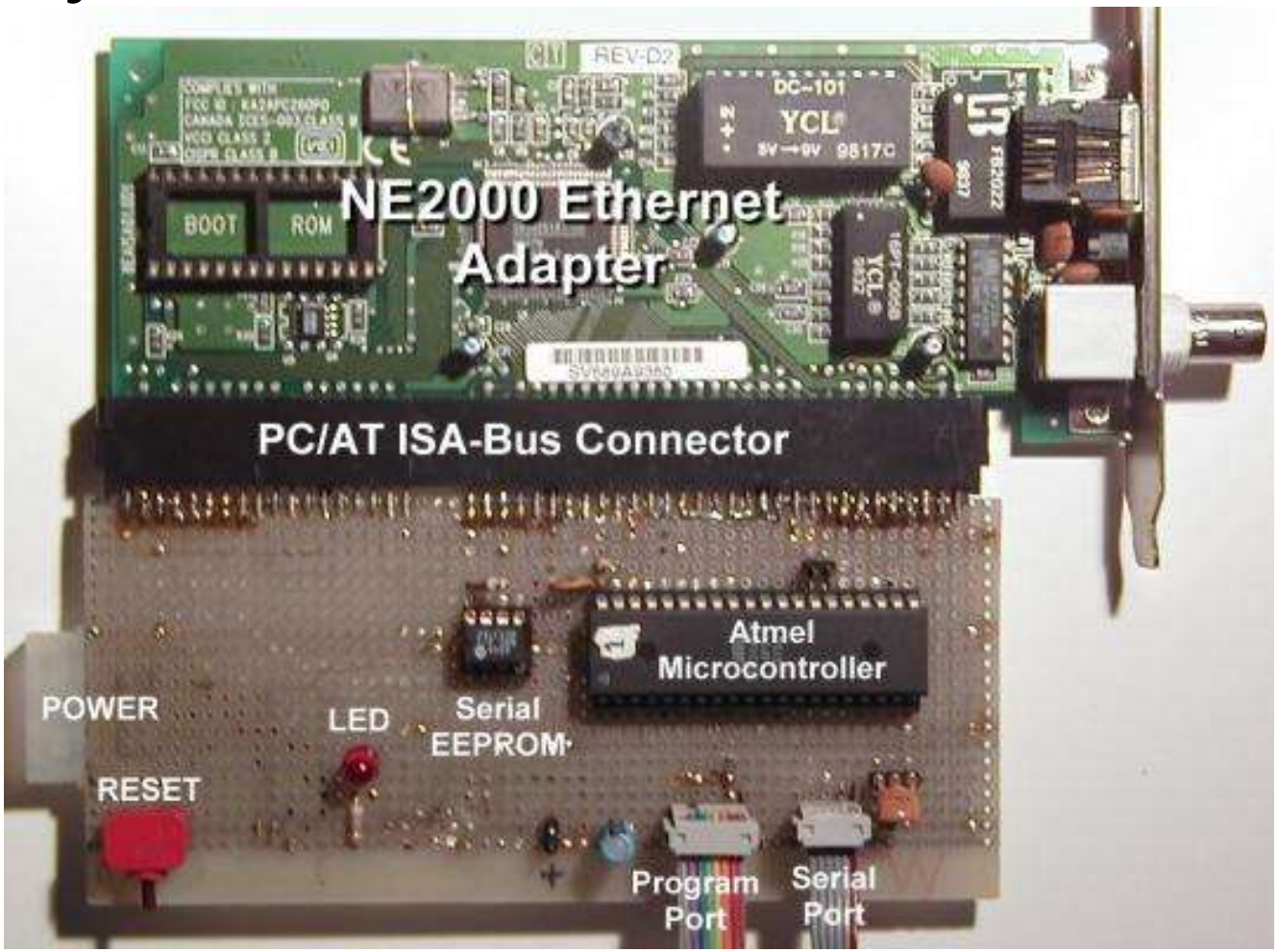


TFTP

- Erőforrás takarékos, UDP
- Igen primitív
- Bootolás-ra szokták használni

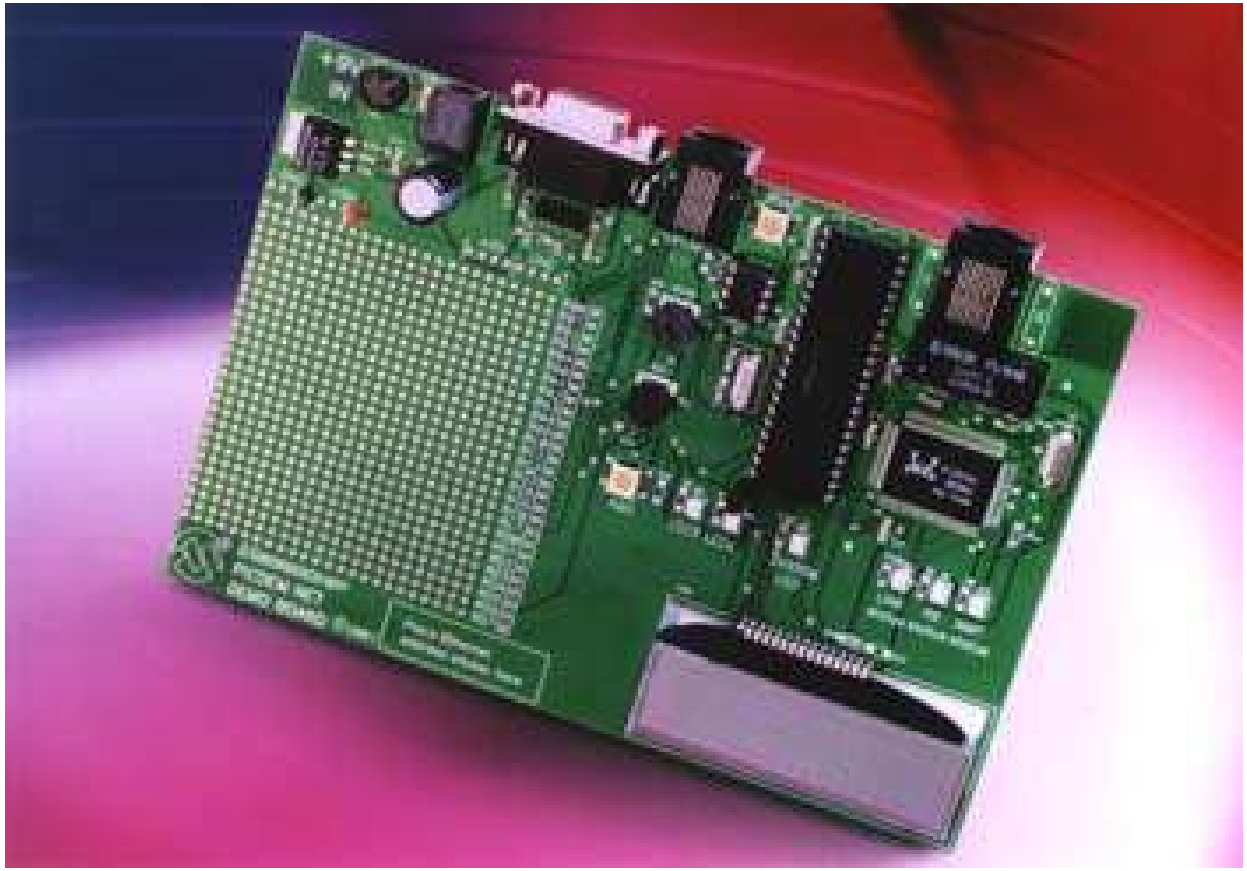
Első generációs megoldások 1995-2000

- ISA kártya + 8 bites kontroller



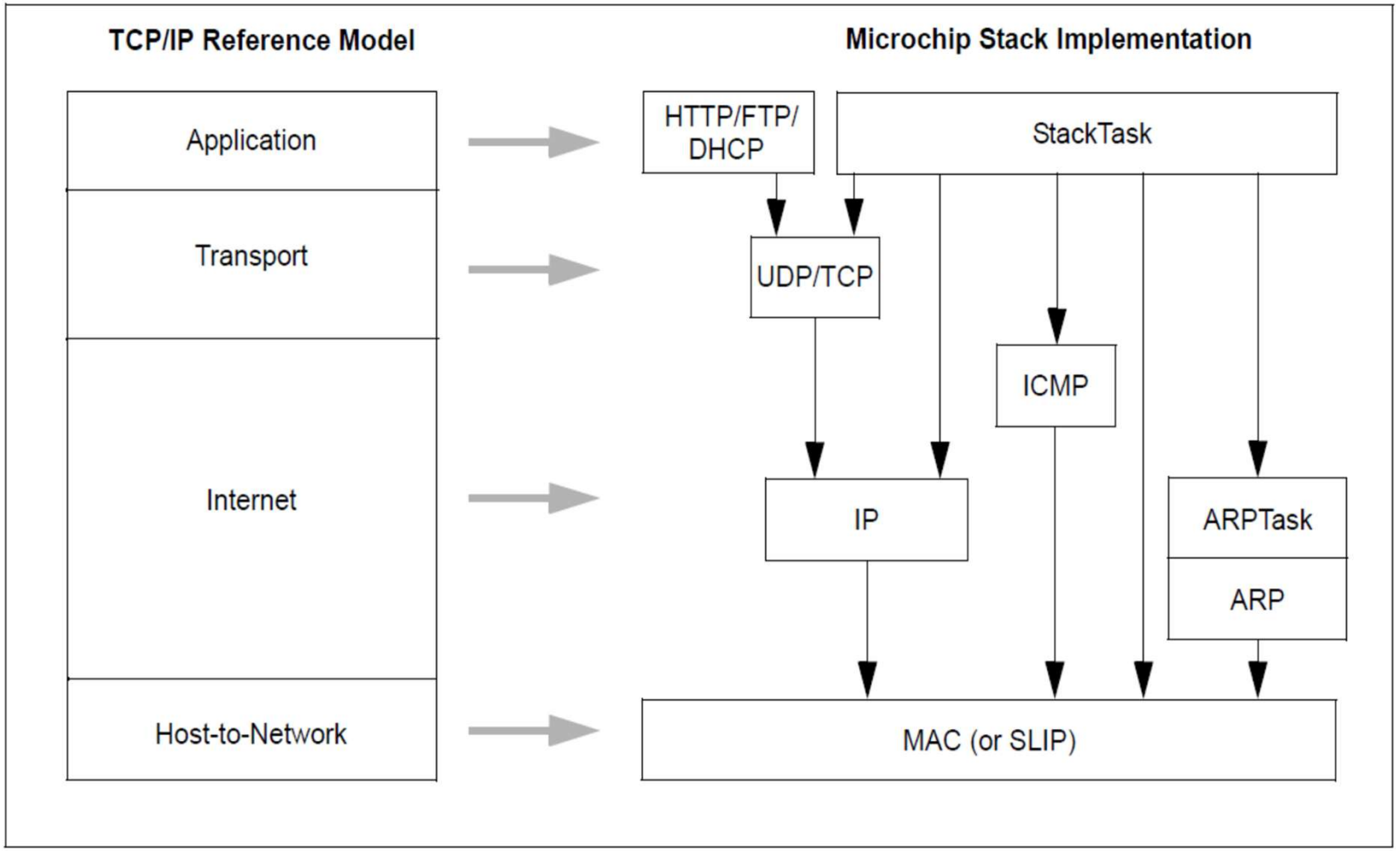
Második generáció megoldások 2002-2005

- 8 bites kontroller + Ethernet kontroller
  - Ethernet kontroller belső RAM-jának használata



Megvalósítások, Microchip AN833

FIGURE 2: COMPARING THE MICROCHIP TCP/IP STACK STRUCTURE TO THE TCP/IP REFERENCE MODEL



Microchip AN833, „ütemezése”

```
// Main entry point
void main(void)
{
        // Perform application specific initialization
        ...

        // Initialize Stack components.
        // If StackApplication is used, initialize it too.
        TickInit();
        StackInit();
        HTTPInit(); // Only if HTTP is used.
        FTPInit(); // Only if FTP is used.

        // Enter into infinite program loop
        while(1)
        {
                // Update tick count. Can be done via interrupt.
                TickUpdate();

                // Let Stack Manager perform its task.
                StackTask();

                // Let any Stack application perform its task.
                HTTPServer();   // Only if HTTP is used.
                FTPServer();    // Only if FTP is used.

                // Application logic resides here.
                DoAppSpecificTask();
        }
}
```

Microchip, AN833 erőforrás használata

TABLE 4: MEMORY USAGE FOR THE VARIOUS STACK MODULES USING HI-TECH® PICC-18™ COMPILER

Module	Program Memory (words)	Data Memory (bytes)
MAC (Ethernet)	906	5(1)
SLIP	780	12(2)
ARP	392	0
ARPTask	181	11
IP	396	2
ICMP	318	0
TCP	3323	42
HTTP	1441	10
FTP Server	1063	35
DHCP Client	1228	26
IP Gleaning	20	1
MPFS(3)	304	0
Stack Manager	334(4)	12+ICMP Buffer

Harmadik generáció megoldások 2005+

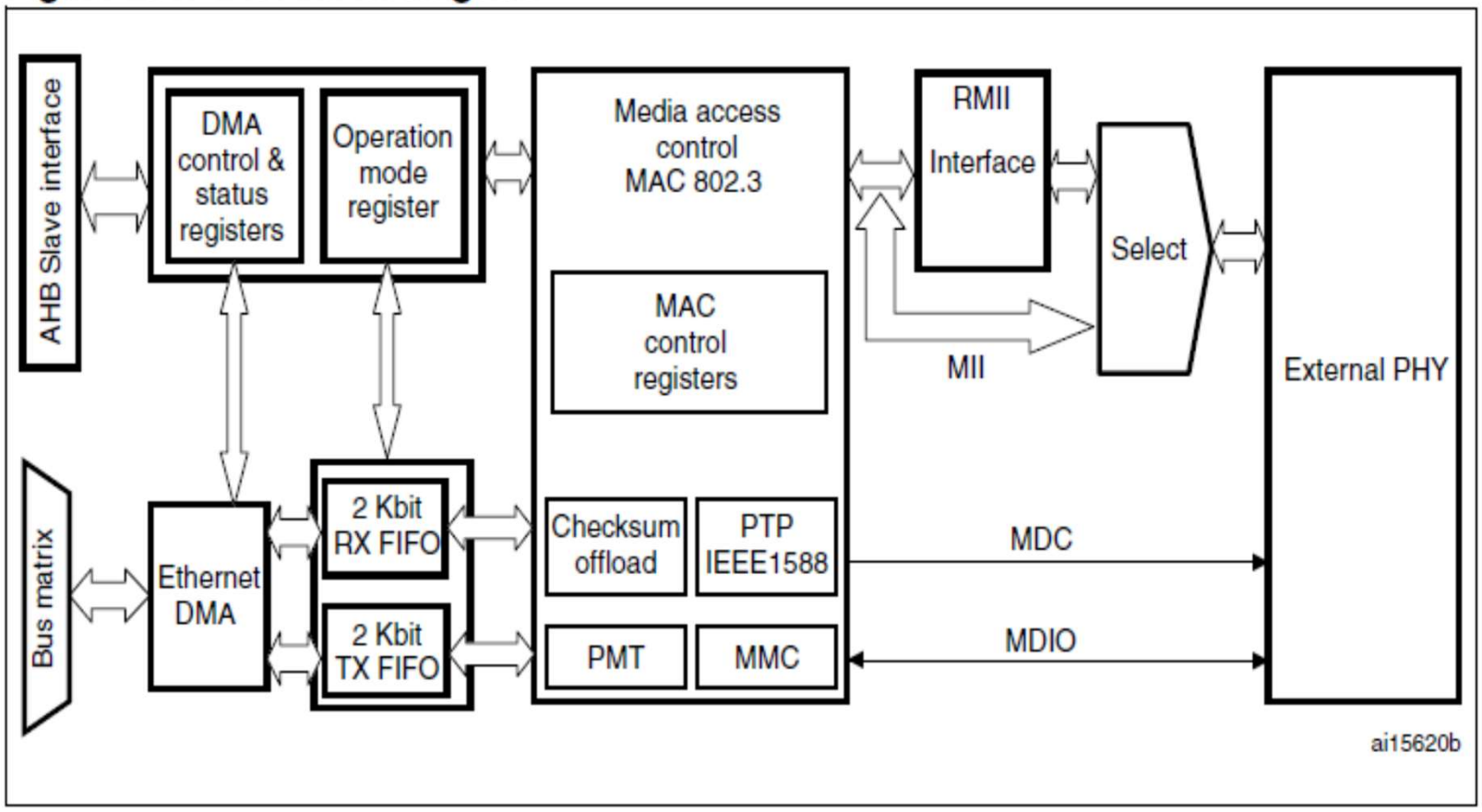
- Különálló TCP/IP chip
  - SPI: Microchip
  - Párthuzamos
  - Sorosporti protokoll: Matcport
- 32 bites vezérlők
  - Integrált Ethernet controller
  - Megfelelő méretű belső RAM
  - Elég gyors processor
- Adam Dunkels protokollstackjei
  - LwIP
  - uIP

Az STM32 Ethernet Controller-e

- 10/100 Mbit/s adatátviteli mód támogatás
- LAN wakeup üzenet azonosítás,
- Segíti az IPv4 header checksum és TCP, UDP, ICMP checksum-ok ellenörzését.
- 2-KB Transmit FIFO
- 2-KB Receive FIFO
- IEEE 1588-2002. 64 bites timestampek.

Az STM32 Ethernet Controller-e

Figure 295. ETH block diagram



LwIP, uIP

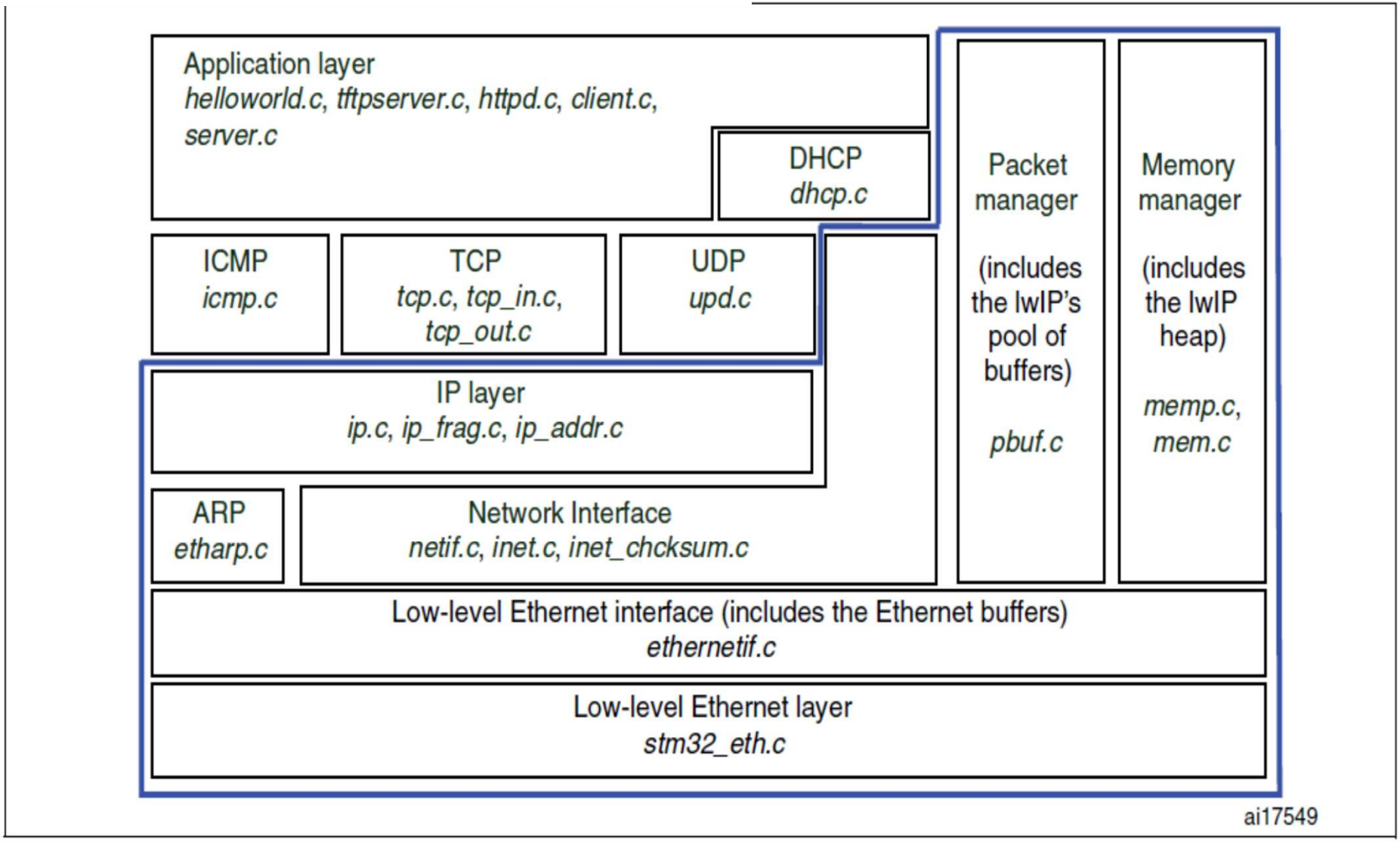
- Mind a kettő RFC kompatibilis
- A uIP alacsonyabb kategóriájú vezérlőkre, mint a LwIP
- Buffer memoria a legkritikusabb,
  - Kevessebb a RAM, mint a ROM
- Traditionális BSD socket API
  - Nem eseményvezérelt programozáshoz
  - Rengeteg plusz kód.
- Saját lwIP és uIP API-k
  - Egyszerűbb kód
  - Esemény vezérelt orientált IF

LwIP könyvtárstruktúra

- **lwip/src lwip/src/api** - the Netconn API, Socket API, and the tcpip thread
- **lwip/src/core** - core code: DHCP, TCP, UDP, and support code (memory, netif, etc)
- **lwip/src/core/ipv4** - IPv4, ICMP
- **lwip/src/core/ipv6** - IPv6
- **lwip/src/core/snmp** - SNMP
- **lwip/src/include** - all headers and includes
- **lwip/src/netif** - ARP and sample Ethernet driver
- **lwip/src/netif/ppp** - PPP

LwIP architektúra

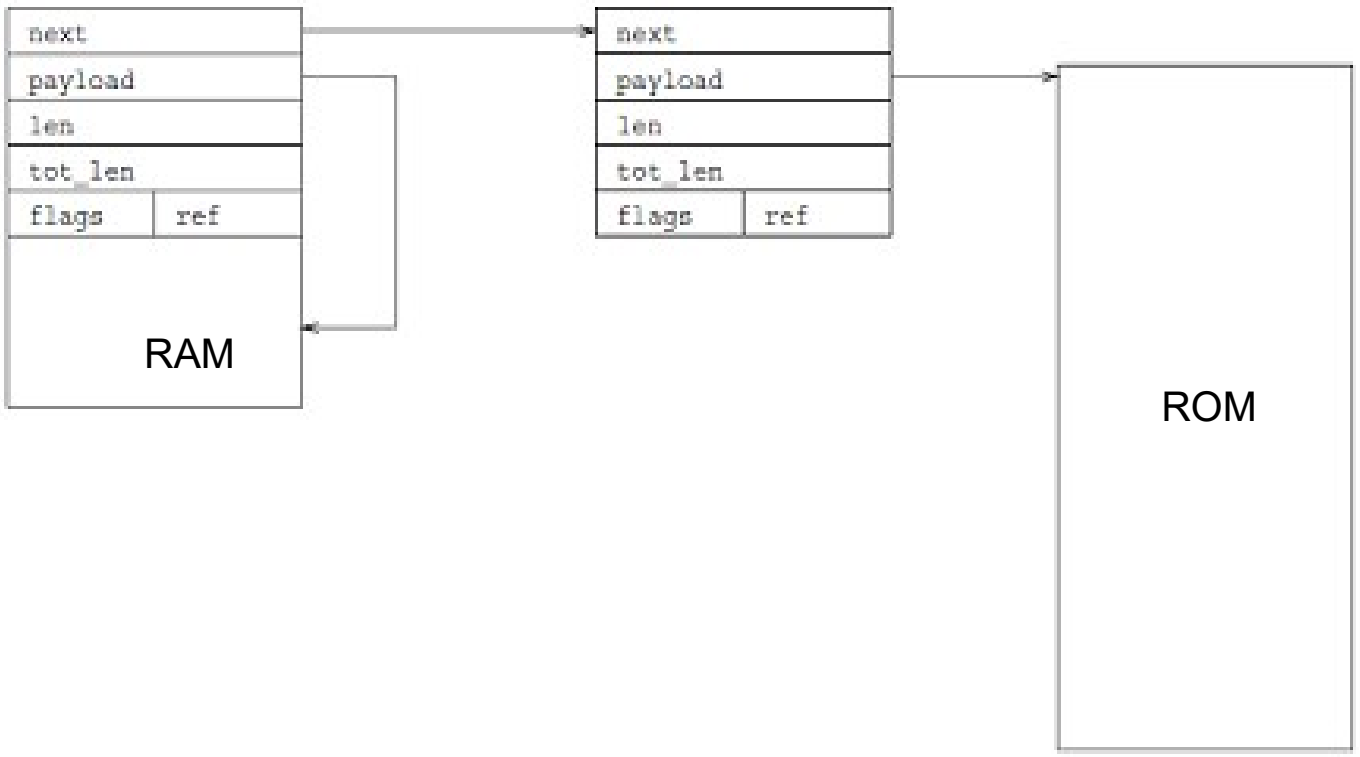
LwIP az STM32 Conectivity Line sorozatra



LwIP memóriakezelés

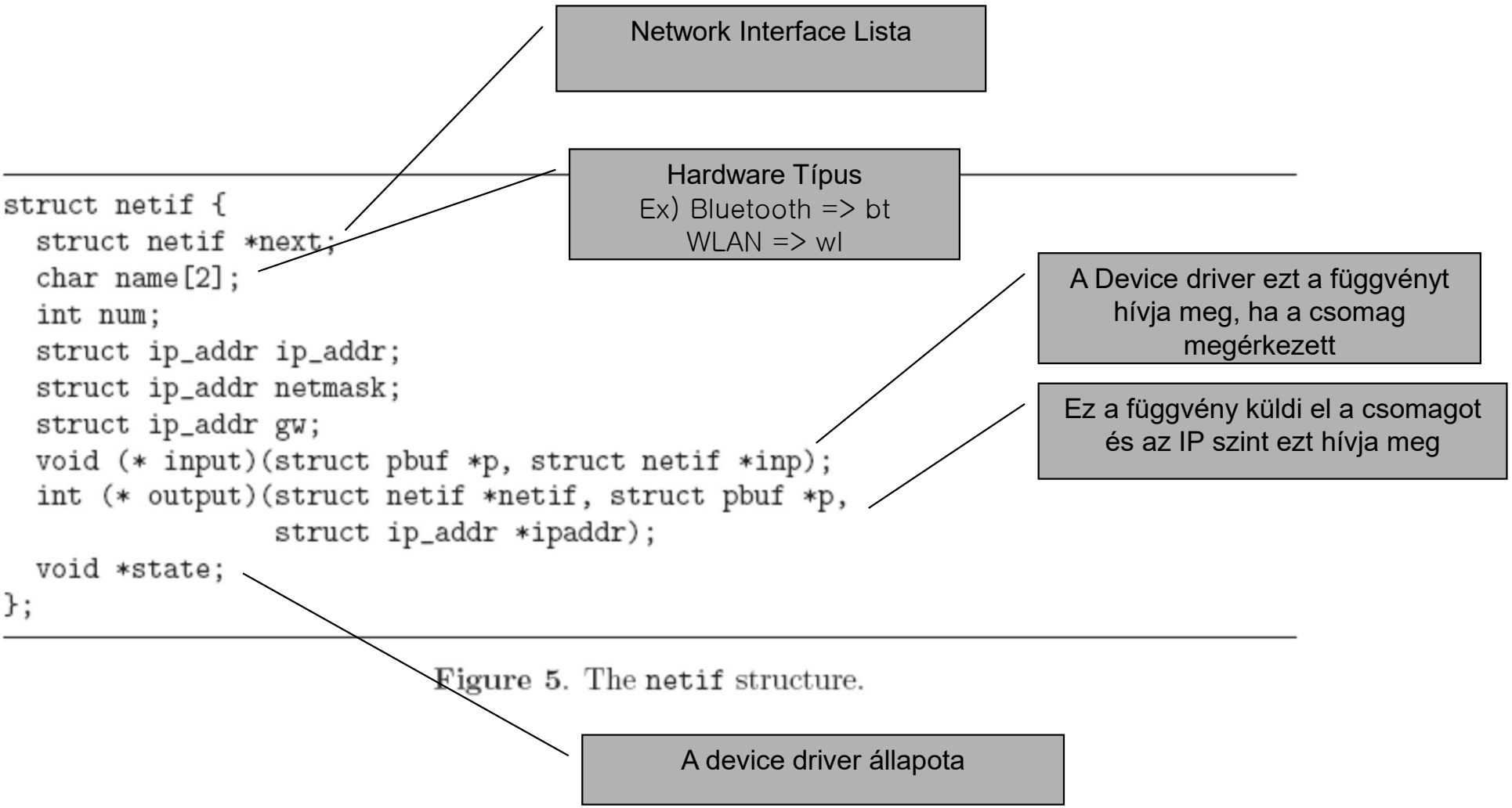
- pbuf sorok használatával oldja ezt meg.
  - pbub-ok lehetnek RAM (dinamikusan foglalt)
  - ROM és POOL (fix méretű memóriaterület) típusúak.

  Többnyire a kimenő adatok RAM és ROM buffereket használnak, míg a bejövők POOL típusút.



Network Interface

- Linkelt Listában




```
struct netif {
  struct netif *next;
  char name[2];
  int num;
  struct ip_addr ip_addr;
  struct ip_addr netmask;
  struct ip_addr gw;
  void (* input)(struct pbuf *p, struct netif *inp);
  int (* output)(struct netif *netif, struct pbuf *p,
                 struct ip_addr *ipaddr);
  void *state;
};
```

Figure 5. The netif structure.

IP feldolgozás

- **Csomag fogadás**
  - Network device driver megívja az ***ip_input()*** függvényt.
    - IP version, header length ellenörzés
    - Header checksum számolás
    - Destination address ellenörzés
- **Csomag küldés**
  - ***ip_output() függvényen keresztül***
    - A megfelelő network interface megtalálása
    - IP header mezők kitöltése
    - IP header checksum kiszámítása

IP feldolgozás

- Csomag forwardolás
  - Forwardolni kell, ha
    - Ha az egyik Network interface-nek sem ugyanaz a címe, mint a kimenő csomag cél IP címe
  - ***ip_forward()*** *függvény*
    - TTL mező csökkentés
    - Ha TTL nulla akkor ICMP error message küldése

IP feldolgozás

- ICMP processing

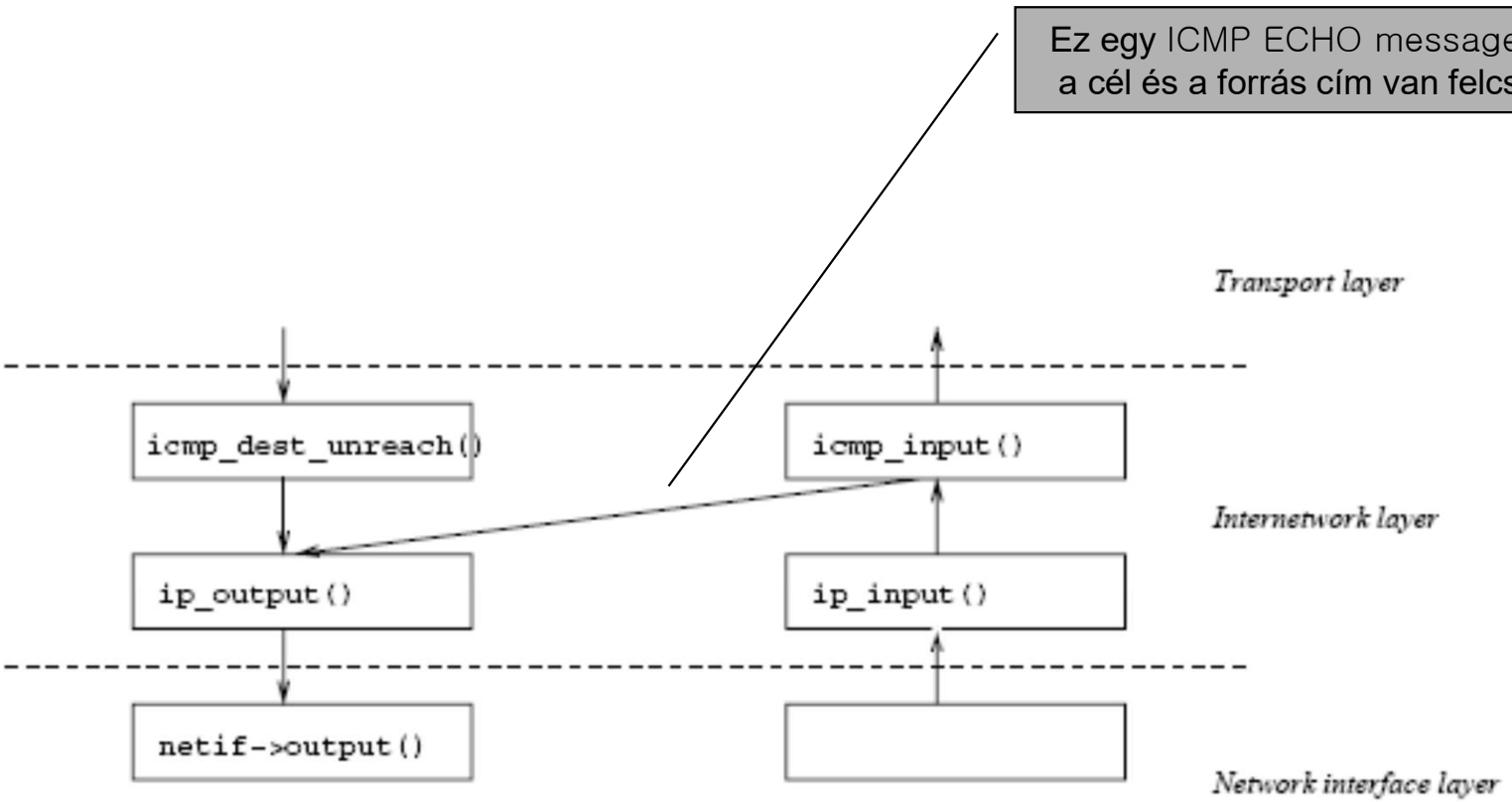


Figure 6. ICMP processing

UDP feldolgozás

- Az udp_pcb structure

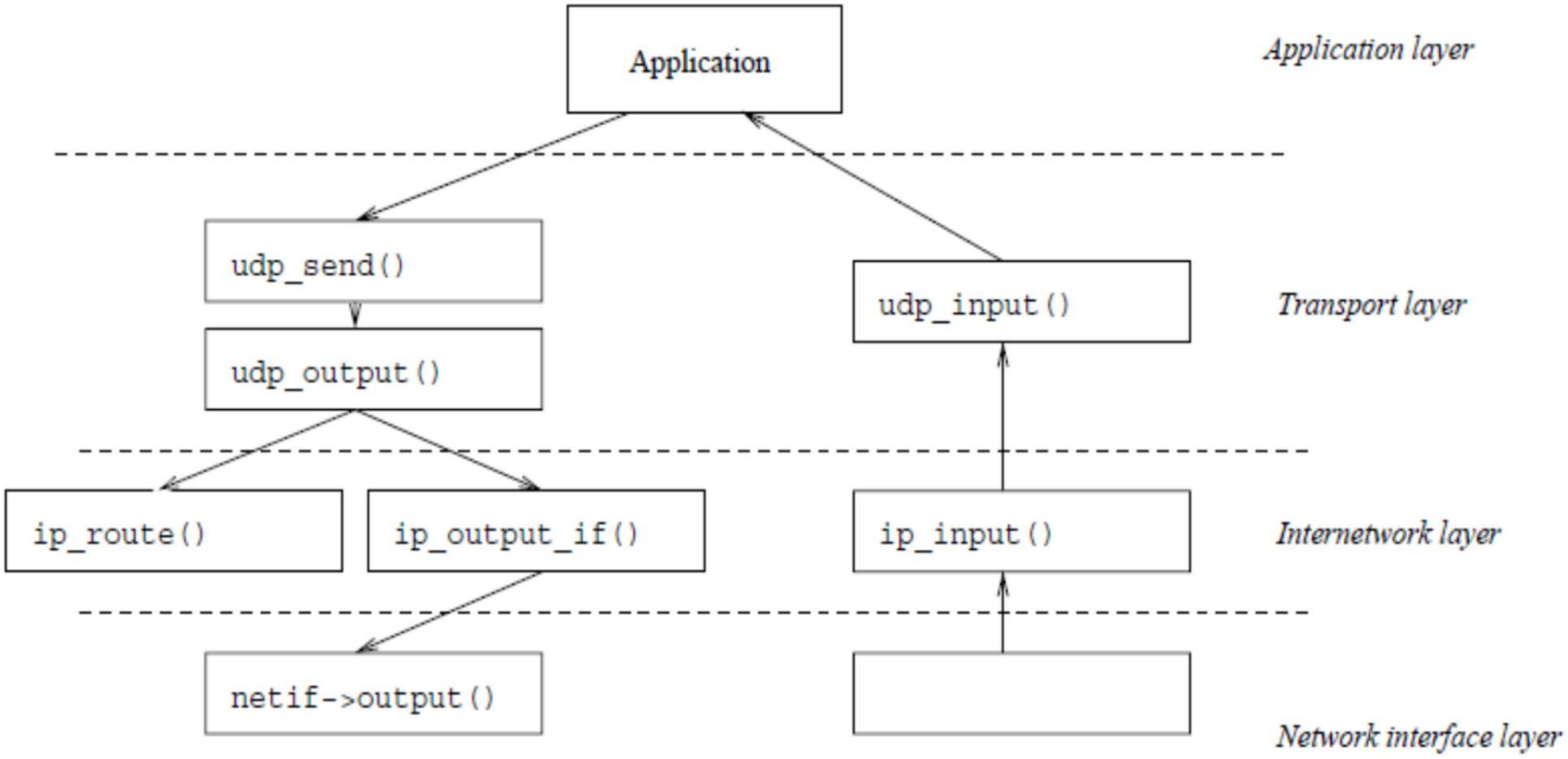
Az UDP csomagok egy likelt listában vannak nyilvántartva

Ez a függvény hívódik meg amikor a datagram megérkezik

```
struct udp_pcb {
  struct udp_pcb *next;
  struct ip_addr local_ip, dest_ip;
  u16_t local_port, dest_port;
  u8_t flags;
  u16_t chksum_len;
  void (* recv)(void *arg, struct udp_pcb *pcb, struct pbuf *p);
  void *recv_arg;
};
```

Figure 7. The udp_pcb structure

UDP alapú alkalmazás



Feldolgozás

```
struct tcp_pcb {
  struct tcp_pcb *next;
  enum tcp_state state;      /* TCP state */
  void (* accept)(void *arg, struct tcp_pcb *newpcb);
  void *accept_arg;
  struct ip_addr local_ip;
  u16_t local_port;
  struct ip_addr dest_ip;
  u16_t dest_port;
  u32_t rcv_nxt, rcv_wnd;    /* receiver variables */
  u16_t tmr;
  u32_t mss;                 /* maximum segment size */
  u8_t flags;
  u16_t rttest;              /* rtt estimation */
  u32_t rtseq;               /* sequence no for rtt estimation */
  s32_t sa, sv;              /* rtt average and variance */
  u32_t rto;                 /* retransmission time-out */
  u32_t lastack;             /* last ACK received */
  u8_t dupacks;              /* number of duplicate ACKs */
  u32_t cwnd, u32_t ssthresh;  /* congestion control variables */
  u32_t snd_ack, snd_nxt,    /* sender variables */
    snd_wnd, snd_wl1, snd_wl2, snd_lbb;
  void (* recv)(void *arg, struct tcp_pcb *pcb, struct pbuf *p);
  void *recv_arg;
  struct tcp_seg *unsent, *unacked, /* queues */
        *ooseq;
};
```

Ez a függvény hívódik meg, ha a Listener csatlakozott

Következő sequence number

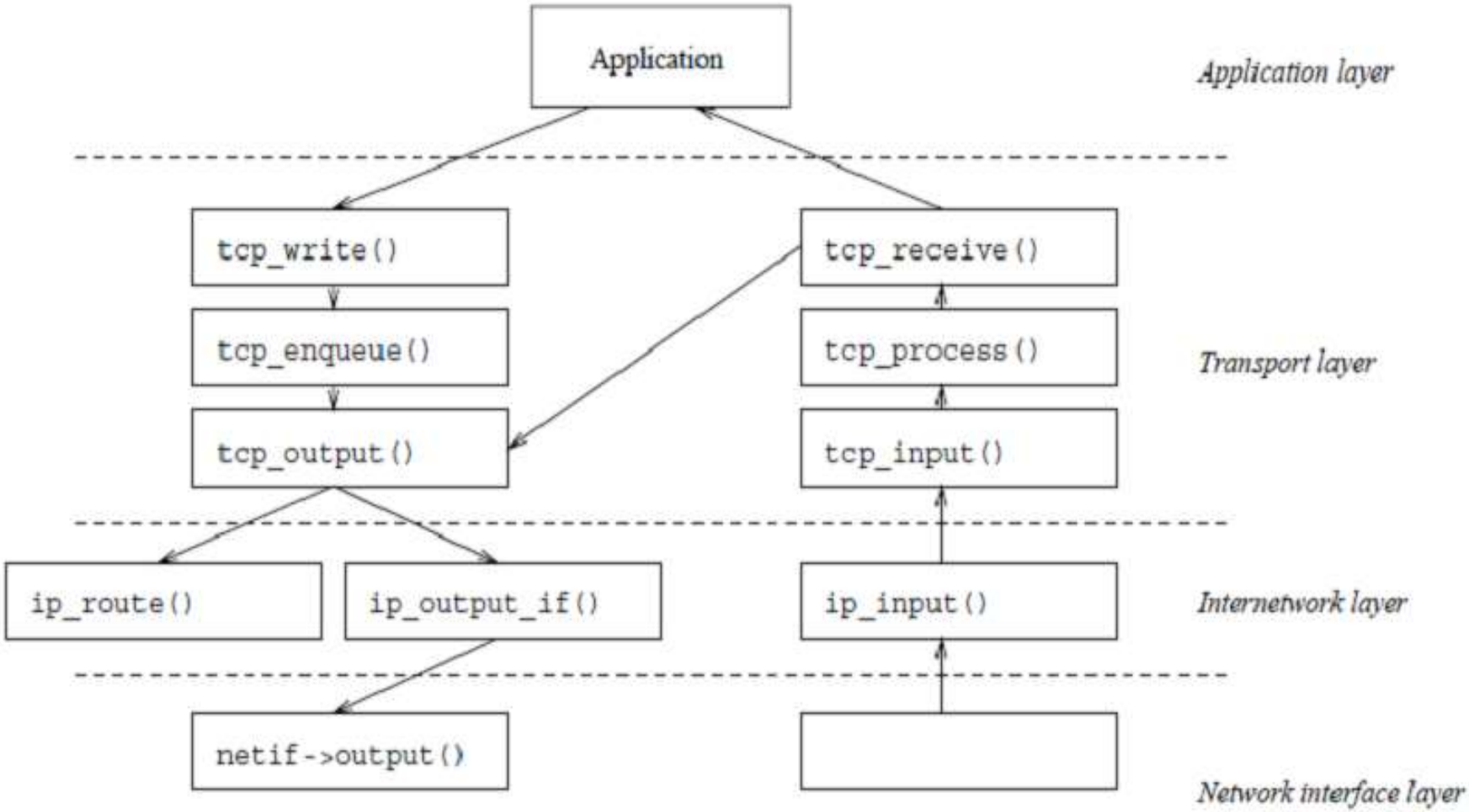
A Receiver's window

Timer a TIME-WAIT állapothoz

Ezzel a függvénnyel adjuk át a vett adatot az Alkalmazás rétegnek

Figure 10. The tcp_pcb structure

TCP alapú alkalmazás



A LwIP memóriaigénye

Table 2. STM32F107xx lwIP demonstration footprint(1)

Modules	Description	Flash memory (bytes)	SRAM (bytes)
Mandatory modules	Ethernet driver and interface, lwIP memory management and IP modules	7848	49590
TCP modules	TCP packet handling using the raw API	7562	80
UDP modules	UDP datagram handling using the raw API	856	4
Optional modules	ICMP	394	0
	DHCP	3164	4
Application modules	Hello word	376	0
	TFTP server	1467	1684
	Web server	32607(2)	4617(3)
	Server	328	0
	Client	412	4
STM32 firmware	STM32F107xx's firmware library	2296	24
STM3210C-EVAL board	STM3210C-EVAL dedicated files	8852(4)	64
Main and system initialization	main file and system initialization	2480	1624(5)
efsl	File system	8338	0
Others	Standard libraries	1884	105
Total		78764	57800

uIP

- Lényegesen kisebb kód
- Egy globális memória pool
  - Egyszerre egy üzenet tárolása
  - A küldésre is felhasználja
- Nagyon limitált fragmentáció támogatás (egyszerre egy csomag)
- Nagyon Limitált TCP retransmission
  - Csak az utolsó csomag újraküldése

- Nagyon kicsivel tud többet, mint a Microchip AN833