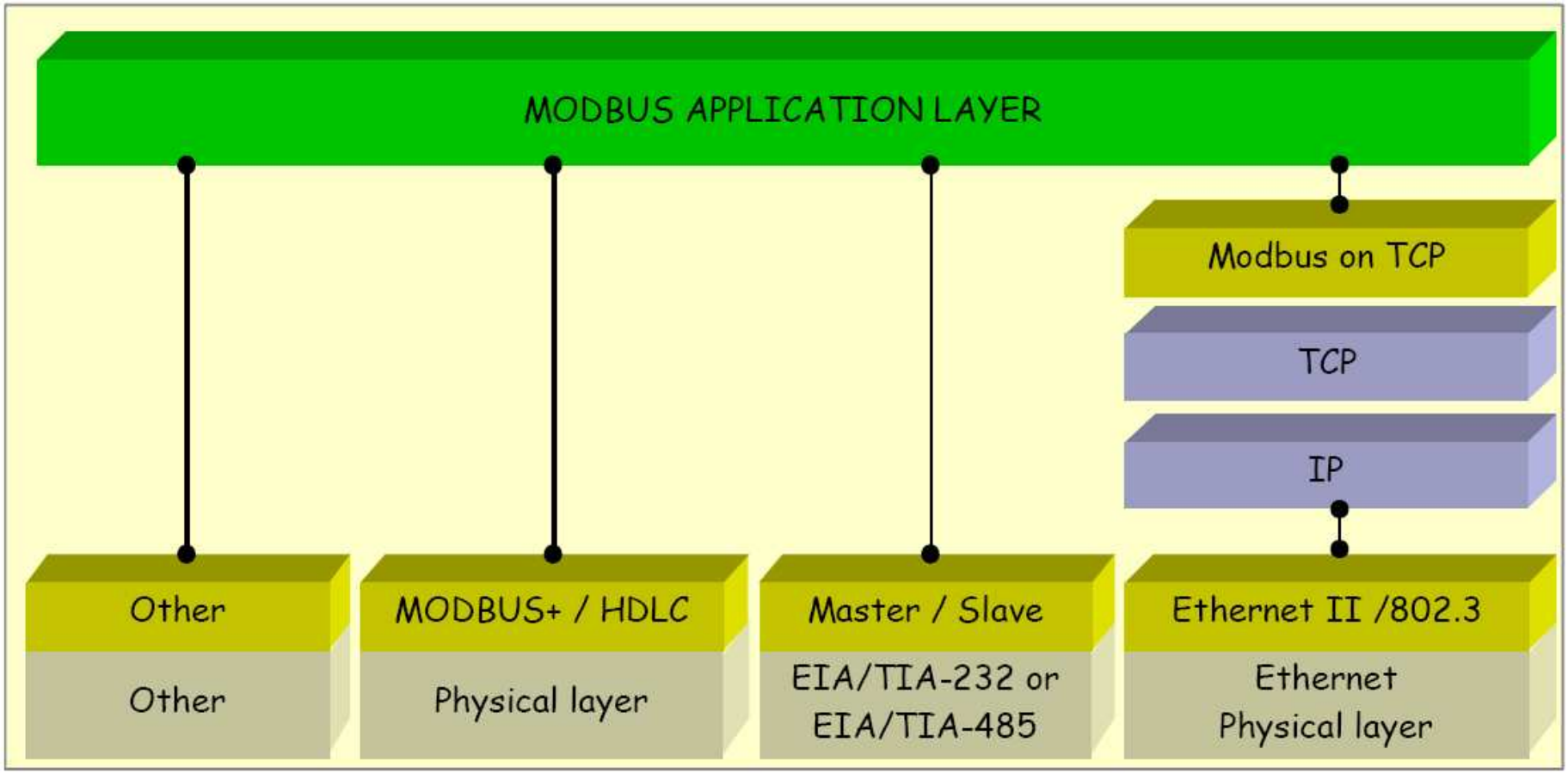
UNIVERSITÉ PARIS 12 VAL de MARNE

FACULTÉ DES SCIENCES ET TECHNOLOGIE

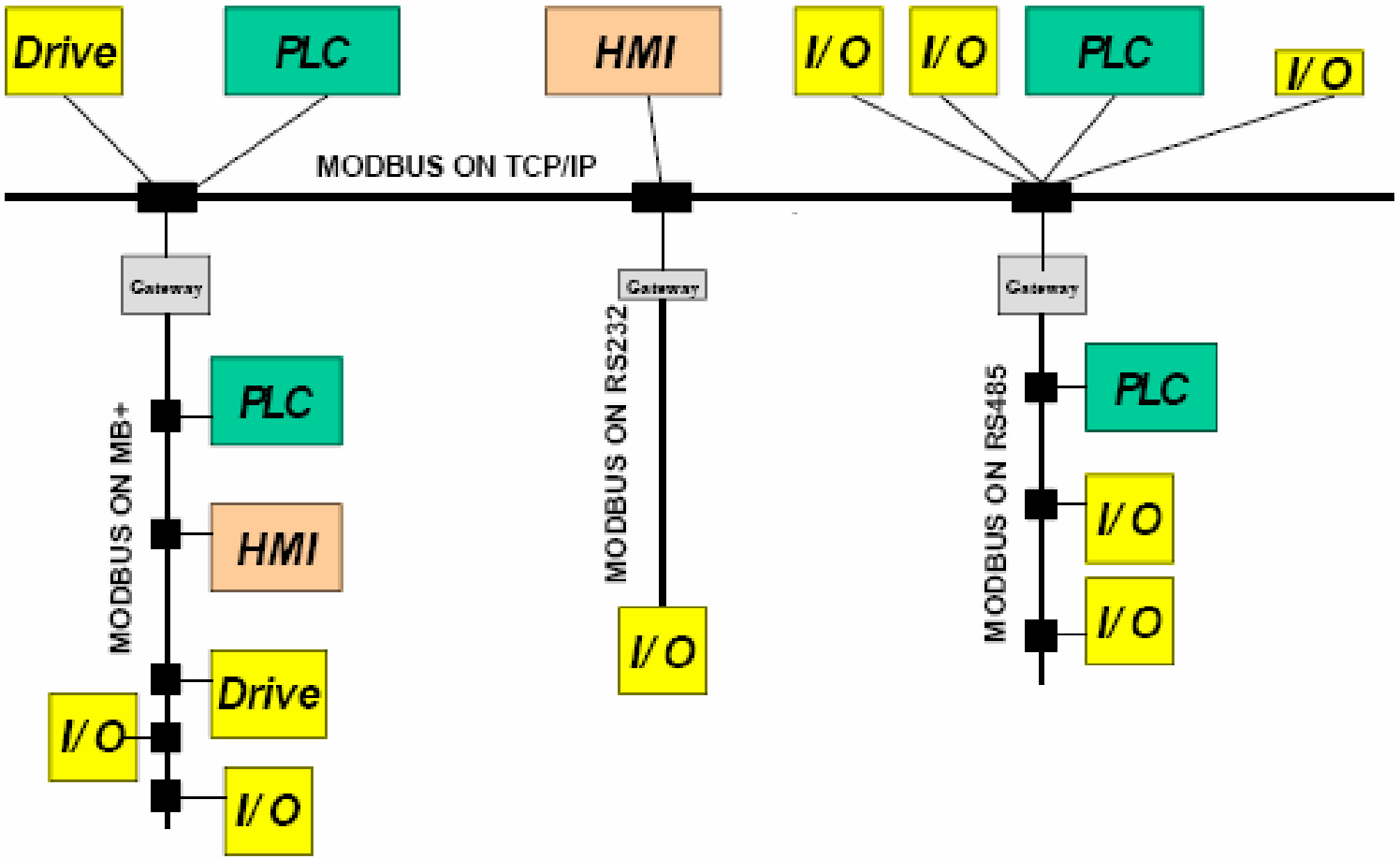
Modbus

Ph. Meyne

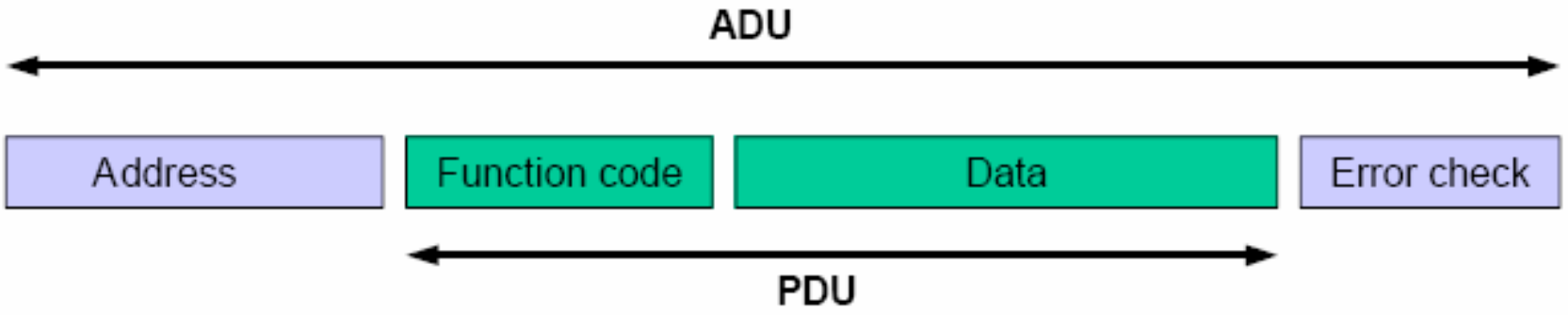
Généralités
Architecture générale



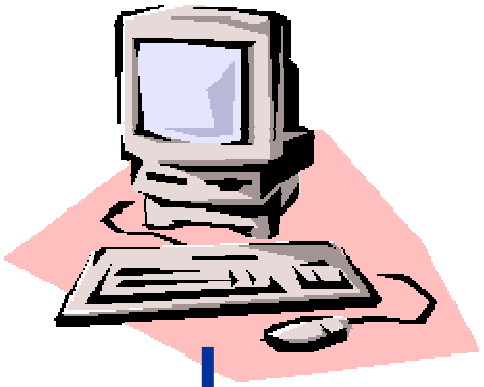
Généralités
Exemple d'architecture



Généralités
Protocole



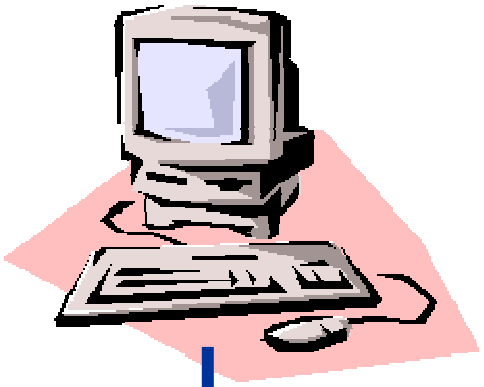
Généralités
Transaction client - serveur



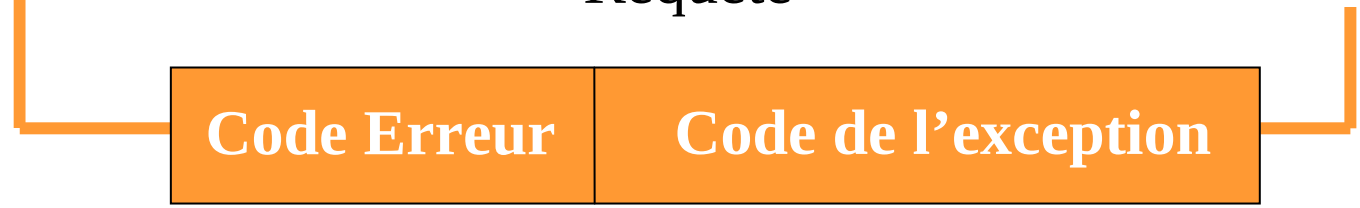
Requête

Réponse à la requête

Généralités
Transaction avec erreur

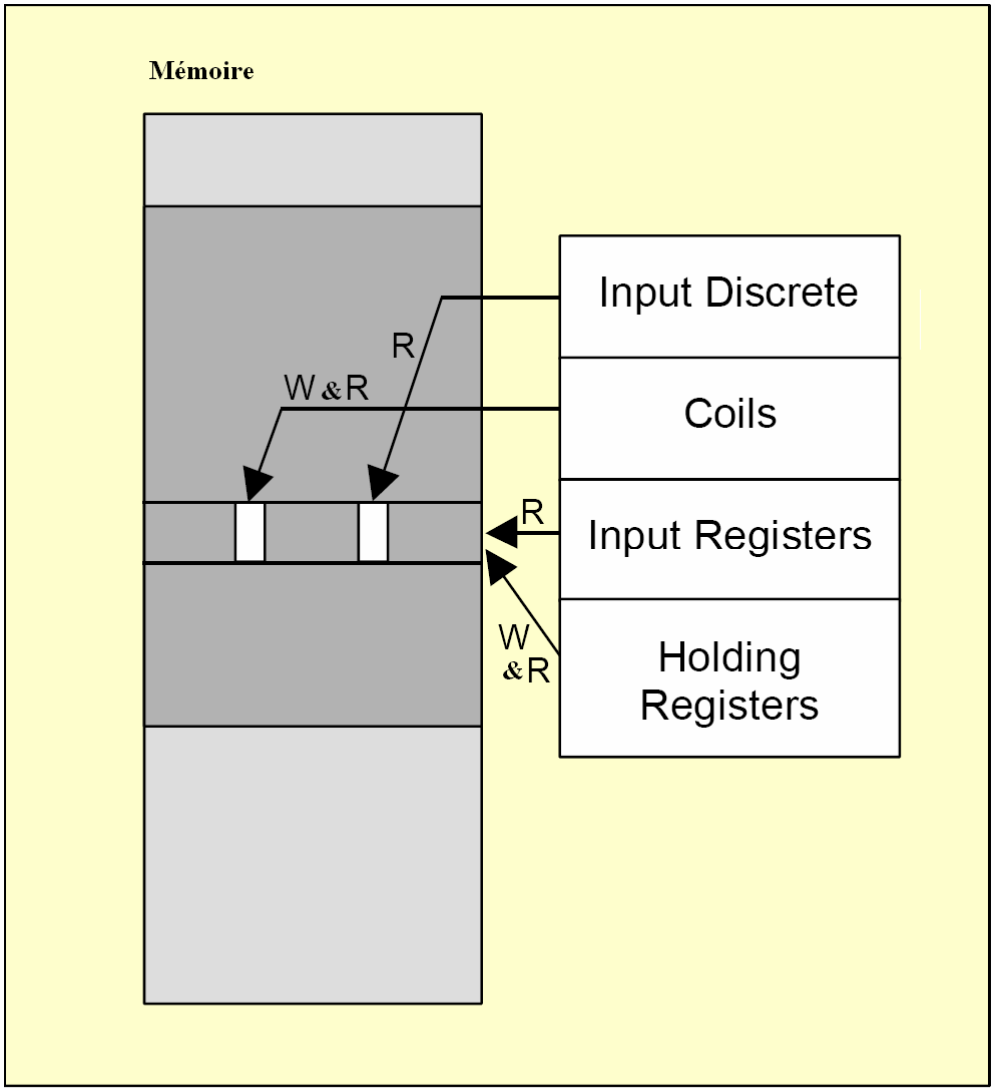


Requête

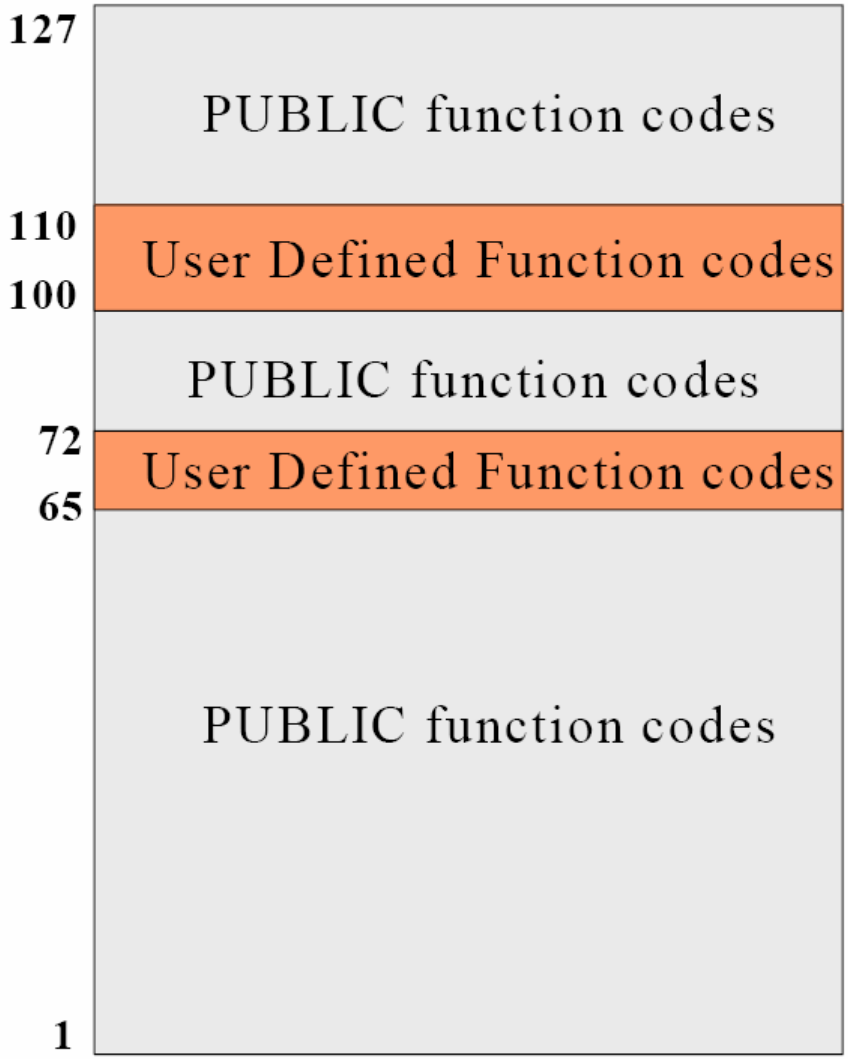


Réponse à la requête

Généralités
Modèle de données



Fonctions
Catégories



Fonctions
Types de fonctions

				Function Codes		
				code	*Sub code*	*(hex)*
Data Access	Bit access	Physical Discrete Inputs	Read Input Discrete	02		02
		Internal Bits Or Physical coils	Read Coils	01		01
			Write Single Coil	05		05
			Write Multiple Coils	15		0F
	16 bits access	Physical Input Registers	Read Input Register	04		04
		Internal Registers Or Physical Output Registers	Read Multiple Registers	03		03
			Write Single Register	06		06
			Write Multiple Registers	16		10
			Read/Write Multiple Registers	23		17
			Mask Write Register	22		16
	File record access		Read File record	20	6	14
			Write File record	21	6	15
Encapsulated Interface			Read Device Identification	43	14	2B

Description des Fonctions
Read Discrete Inputs

Request PDU

Function code	1 Byte	0x01
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of coils	2 Bytes	1 to 2000 (0x7D0)

Response PDU

Function code	1 Byte	0x01
Byte count	1 Byte	N*
Coil Status	n Byte	n = N or N+1

*N = Quantity of Outputs / 8, if the remainder is different of 0 $\Rightarrow$ N = N+1

Error

Function code	1 Byte	Function code + 0x80
Exception code	1 Byte	01 or 02 or 03 or 04

Description des Fonctions
Write Single Register

Request

Function code	1 Byte	0x06
Register Address	2 Bytes	0x0000 to 0xFFFF
Register Value	2 Bytes	0x0000 or 0xFFFF

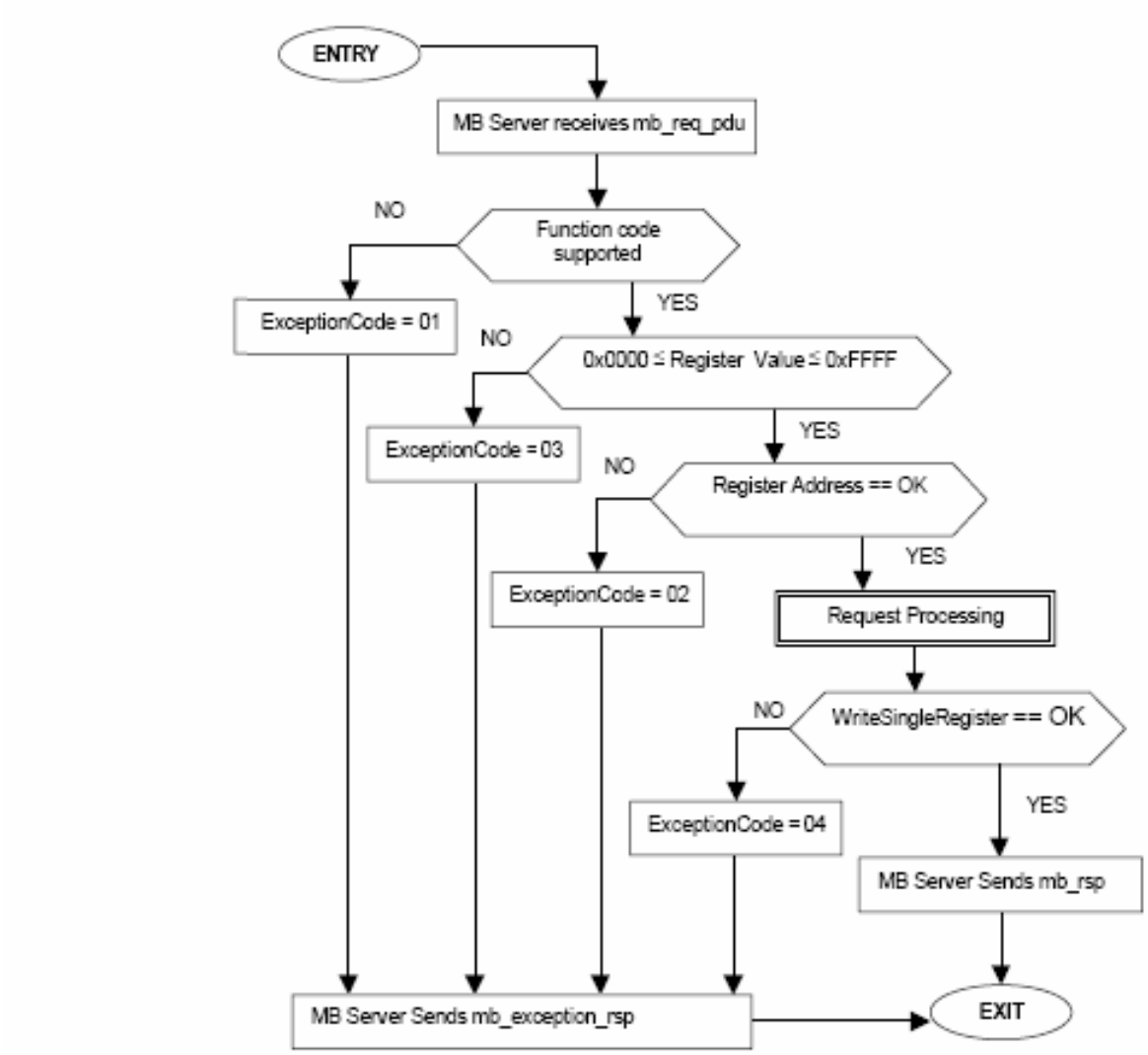
Response

Function code	1 Byte	0x06
Register Address	2 Bytes	0x0000 to 0xFFFF
Register Value	2 Bytes	0x0000 or 0xFFFF

Error

Error code	1 Byte	0x86
Exception code	1 Byte	01 or 02 or 03 or 04

Description des Fonctions
Write Single Register - suite



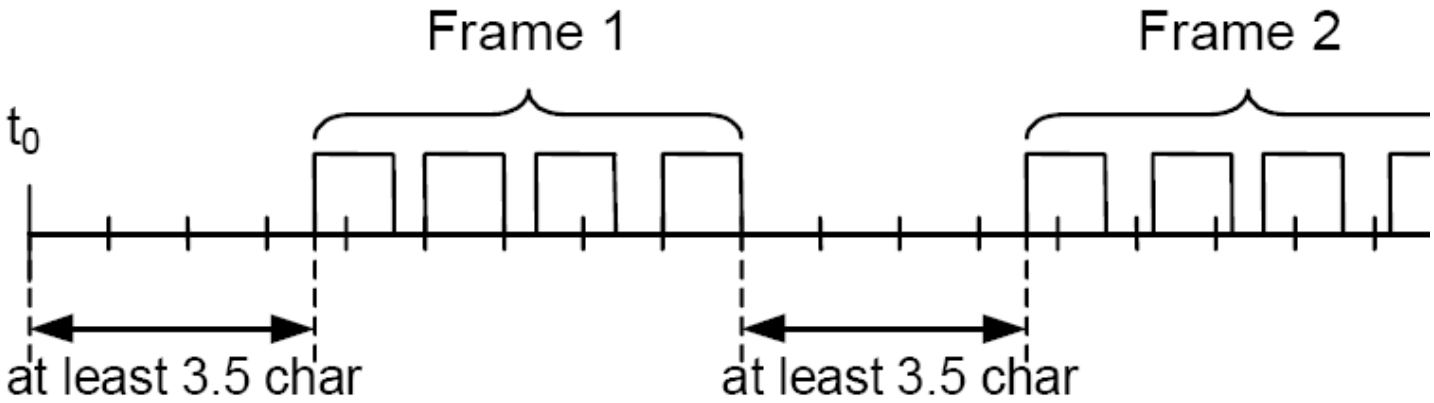
With Parity Checking

Start	1	2	3	4	5	6	7	8	Par	Stop

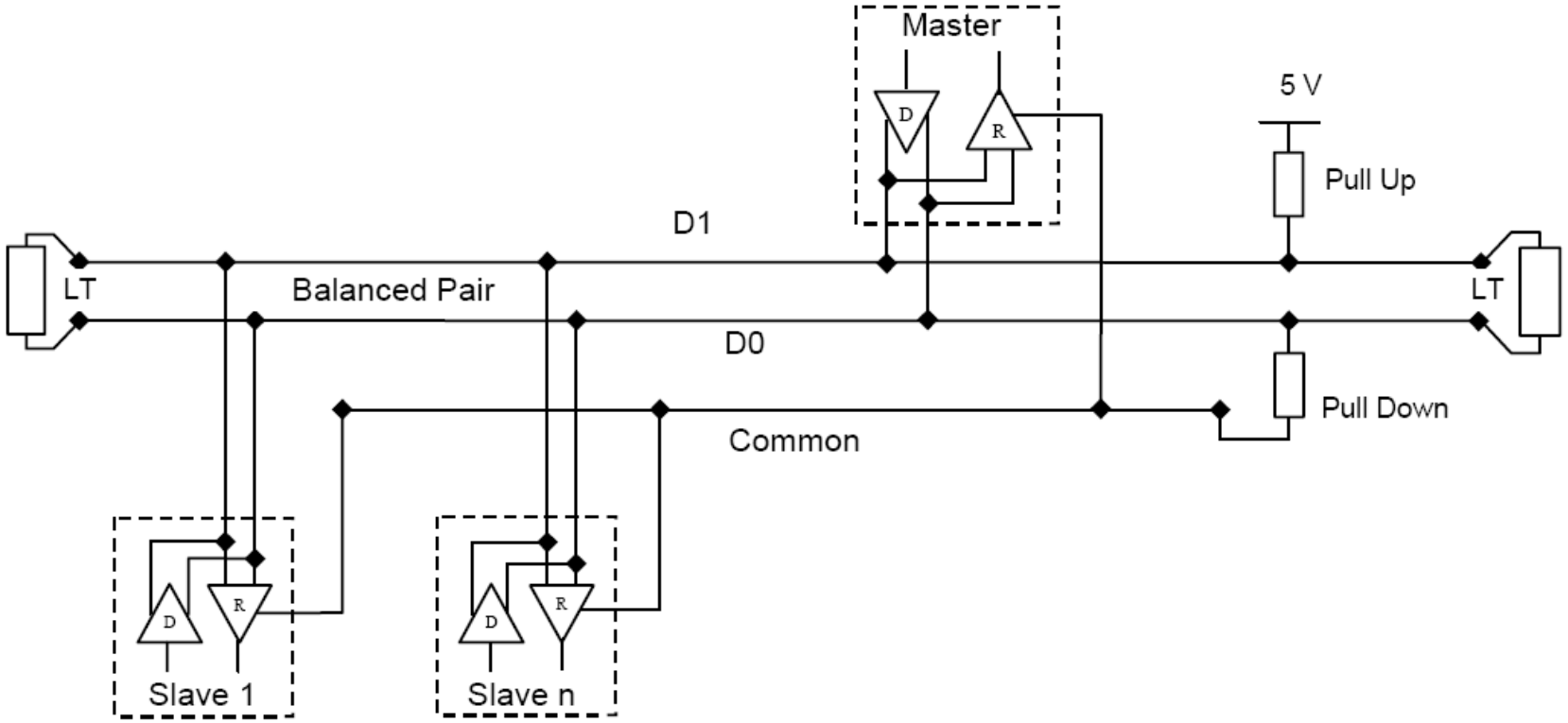
Without Parity Checking

Start	1	2	3	4	5	6	7	8	Stop	Stop

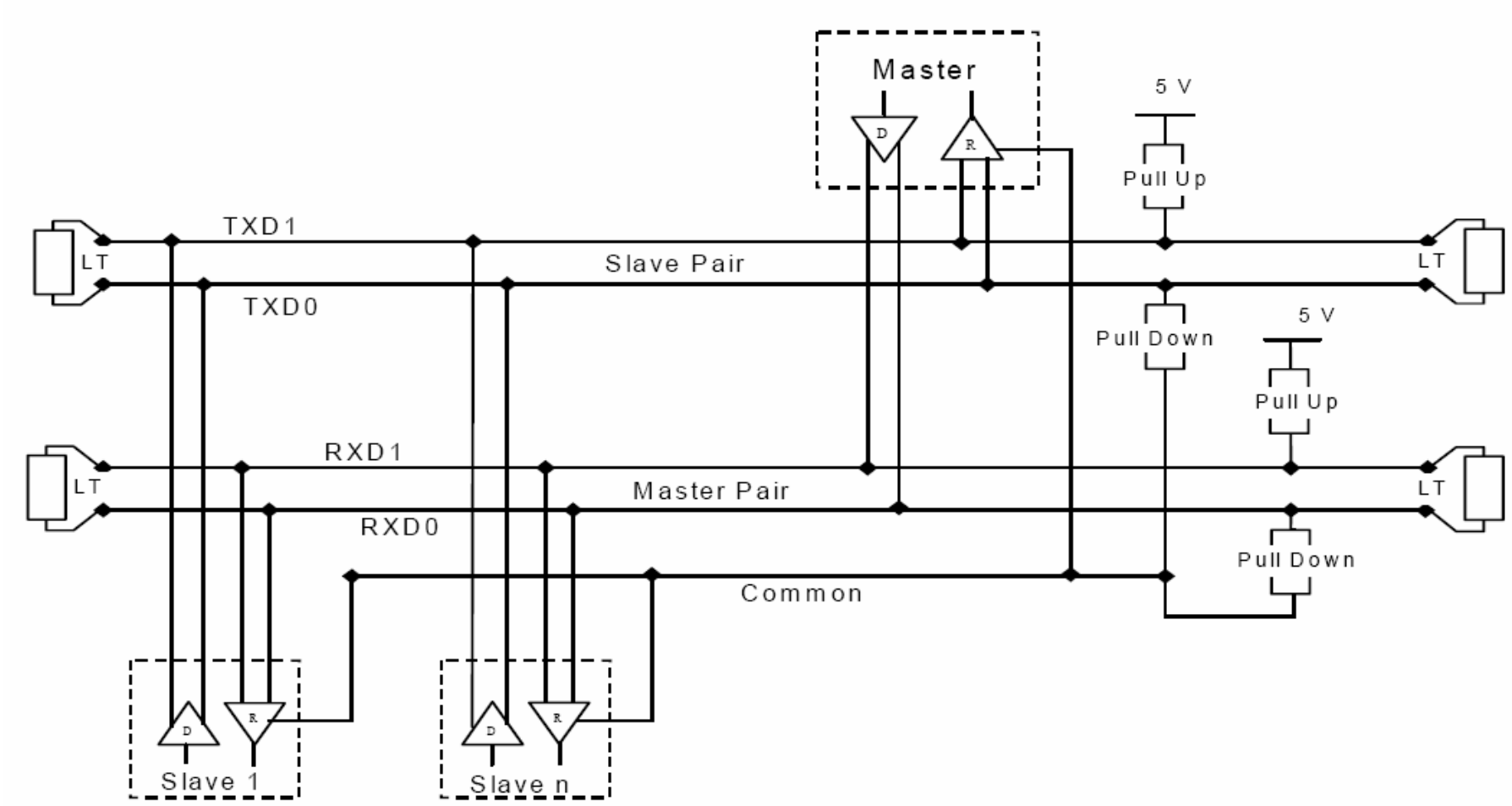
Modbus sur ligne série
Format des octets (suite)



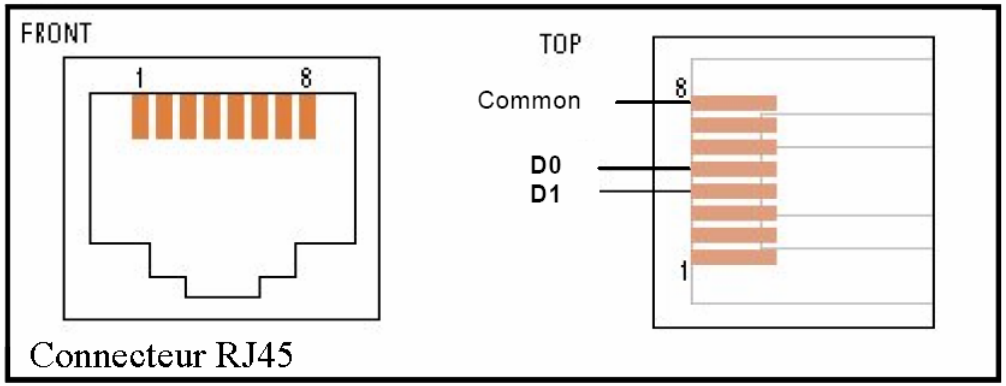
Modbus sur ligne série
Couche physique – liaison deux fils

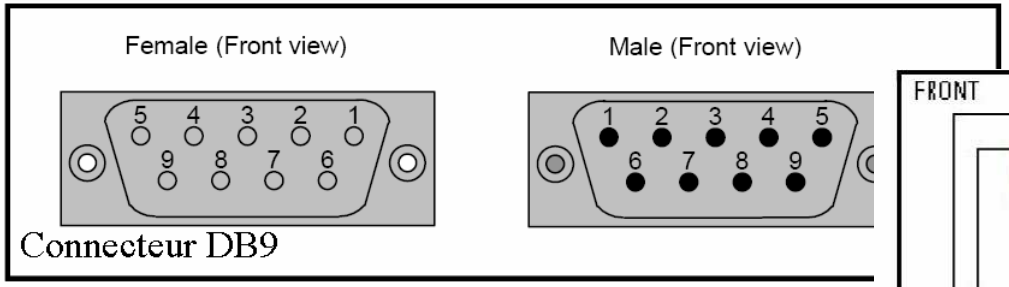


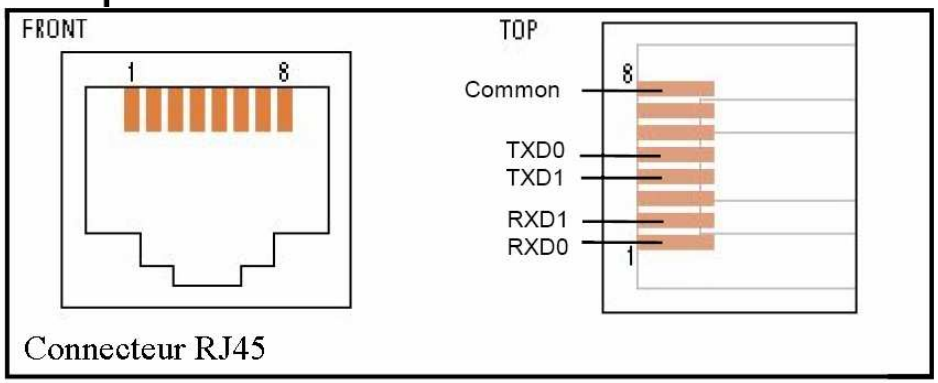
Modbus sur ligne série
Couche physique – liaison quatre fils

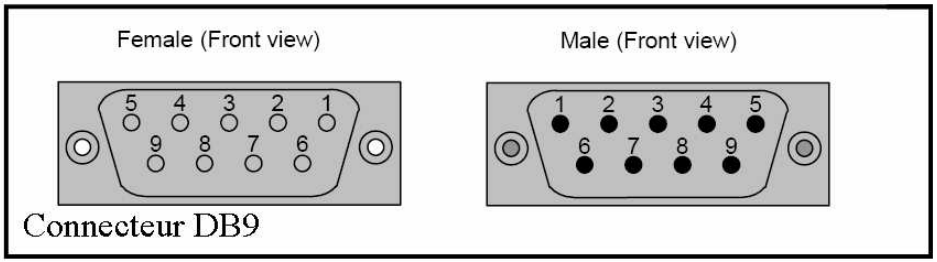


Modbus sur ligne série
Couche physique – Connecteur









Modbus sur ligne série
Exemple de mise en œuvre

Modbus sur ligne série
Mise en œuvre - Programmation

```
void InitPortSerie(void)
{       unsigned char car;
        int erreur;
        erreur = ioperm(0x03F8,10,666);
        if (erreur == -1){
                                printf("permission refusee\n");
                                _exit(errno);
                        }
        // acces à Divisor Latch et proegarmmation de la vitesse
        outb(inb(LCR) |0x80;,LCR);          // bit DLAB à 1
        outb(0x00,DLM);             // baud
        outb(0x0C,DLL);
        // programmation de la liaison
        outb(0x07,LCR);
}// fin init port série
```

Modbus sur ligne série
Mise en œuvre – Programmation (suite)

```
void ecrire( unsigned char CarEnvoi )
{
        int erreur;
        erreur = ioperm(0x03F8,10,666);
        if (erreur == -1){
                        printf("permission refusees\n");
                        _exit(errno);
                }
        // tant que le registre d'émission est plein attendre
        while((inb(LSR)&0x20)==0)
        {}
        outb(CarEnvoi,THR);
}
```

Modbus sur ligne série
Mise en œuvre – Programmation (suite)

Emission de la trame	Réception de la trame
0x1 0x6 0x0 0x0 0xff 0xff 0x88 0x7a	0x1 0x6 0x0 0x0 0x0 0xf0 0x89 0x8e
0x1 0x6 0x0 0x1 0xff 0xff 0xd9 0xba	0x1 0x6 0x0 0x1 0xff 0xff 0xd9 0xba
0x1 0x6 0x0 0x2 0xff 0xff 0x29 0xba	0x1 0x6 0x0 0x2 0xff 0xff 0x29 0xba
0x1 0x6 0x0 0x3 0xff 0xff 0x78 0x7a	0x1 0x6 0x0 0x3 0xff 0xff 0x78 0x7a
0x1 0x6 0x0 0x4 0xff 0xff 0xc9 0xbb	0x1 0x6 0x0 0x4 0xff 0xff 0xc9 0xbb
0x1 0x6 0x0 0x5 0xff 0xff 0x98 0x7b	0x1 0x6 0x0 0x5 0xff 0xff 0x98 0x7b
0x1 0x6 0x0 0x6 0xff 0xff 0x68 0x7b	0x1 0x6 0x0 0x6 0xff 0xff 0x68 0x7b
0x1 0x6 0x0 0x7 0xff 0xff 0x39 0xbb	0x1 0x6 0x0 0x7 0xff 0xff 0x39 0xbb
0x1 0x6 0x0 0x8 0xff 0xff 0x9 0xb8	0x1 0x6 0x0 0x8 0xff 0xff 0x9 0xb8
0x1 0x6 0x0 0x9 0xff 0xff 0x58 0x78	0x1 0x6 0x0 0x9 0xfe 0xff 0x59 0xe8
0x1 0x6 0x0 0xa 0xff 0xff 0xa8 0x78	0x1 0x6 0x0 0xa 0xff 0xff 0xa8 0x78

Conclusion

- Modèle client serveur pour une application de terrain
- Concept simple qui permet d'avoir un accès standard
- Système ouvert