PROJET SimEnOM G2

Tests de performances de La liaison internet d'un module Rabbit RCM3700

Régis Schmidt – LESIA

Imprimé le dimanche 23 juillet 2006

Table des matières

Table des figures

Glossaire

CPU : Central Processing Unit
TBD : To Be Defined
TBC : To Be Confirmed
CORBA : Common Object Request Broker Architecture
c-PCI : Compact PCI
I/O : Inputs/Outputs

Documents de référence

[1] Réseaux Andrew Tannenbaum (merci Loïc)
[2] Dynamic C V9.21 User Manual

Objet

L'objet de ce document est de chiffrer les performances de transfert du lien Internet disponible qui équipe le module microcontrôleur RABBIT 3000 RMC 3700.

Spécifications

Le lien Internet du module RMC3700 est un lien 10BaseT, ce qui veut dire 10Mbits/S, et connexion RJ45. Nous nous proposons de mesurer la vitesse maxi d'échange entre un PC et le module en question. On utilise TCP, mode connecté, qui garantit la réorganisation des paquets à l'arrivée dans le cas de grosses données. Comme nous sommes avec des paquets IP, un CRC est systématiquement associé aux paquets et des instructions de services aussi. Ce protocole bas niveau est donc robuste et semble une bonne solution pour remplacer les indéboulonnables ports série qui ont été utilisés jusqu'à maintenant. On constate qu'il est plus facile de mettre en œuvre une communication TCP qu'une communication sur une ligne série. Il n'y a plus de baudrate ni d'autres paramètres du même genre à fixer des deux côtés de la liaison. De plus, plusieurs PC peuvent communiquer avec le module et, des modules peuvent communiquer entre eux. La distance entre n'est plus limitée à quelques mètres (15m c'est la norme en RS232), l'accès à votre expérience depuis l'autre bout du monde vous est maintenant possible.

Conditions de test

Côté réseau

Les deux connections au réseau Internet se font au travers des prises murales qui aboutissent dans un Hub qui gère tout l'étage. Le réseau est segmenté et les perturbations extérieures à la simple interconnexion sont donc supprimées en grande partie.

Côté module

Le programme utilisé est écrit en C et repris presque à l'identique d'un des exemples livré avec le compilateur Dynamic C : il s'agit du programme "echo.c". Ce programme est à l'écoute d'une connection TCP sur le port 80 et reçoit un paquet d'une taille maxi de 4096 octets (c'est la valeur par défaut) et le renvoie vers l'expéditeur. Les impressions de debug ont été supprimées pour ne pas ralentir inutilement l'exécution. Voici le source de ce programme:

/**

```
 *              Samples/TCPIP/echo.c
 *              Copyright (c) 2001, Z-World (jjb)
 *              This program demonstrates the tcp_listen call.
 *
 *      A basic server, that when a client connect, echoes back to them
 *      any data that they send.
 *
 ************************************************************************/
#class auto


/***********************************
 * Configuration                   *
 * -------------                   *
 * All fields in this section must *
 * be altered to match your local  *
 * network settings.               *
 ***********************************/

/*
 * Pick the predefined TCP/IP configuration for this sample.  See
 * LIB\TCPIP\TCP_CONFIG.LIB for instructions on how to set the
 * configuration.
 */
#define TCPCONFIG 1

#use "dcrtcp.lib"

/**
 *  Port Number    Purpose
 *  -----------    -------
 *     13        Daytime
 *     23        Telnet
 *                                      Note: this program doesn't handle telnet option negotiation.
 *     80        HTTP
 */
#define PORT 80


/*********************************
 * End of configuration section *
 *********************************/

//////////////////////////////////////////////////////////////////////////

void main()
{
        int bytes_read;
        char    buffer[100];    /* Currently (DC 7.30), printf() has max 127 bytes it can output. */
        tcp_Socket socket;

        sock_init();

        while(1) {
                tcp_listen(&socket,PORT,0,0,NULL,0);

                printf("Waiting for connection...\n");
```

```
                while(!sock_established(&socket) && sock_bytesready(&socket)==-1)
                        tcp_tick(NULL);

                printf("Connection received...\n");

                do {
                        bytes_read=sock_fastread(&socket,buffer,sizeof(buffer)-1);
//                      bytes_read=sock_read(&socket,buffer,sizeof(buffer)-1);

                        if(bytes_read>0) {
                                buffer[bytes_read]=0;
//                              printf("%s",buffer);
                                sock_write(&socket,buffer,bytes_read);
                        }
                } while(tcp_tick(&socket));

                printf("Connection closed...\n");
        }
}
```

Côté PC

Un programme écrit en LabVIEW 7.1 envoi des paquets dont la taille peut être variable, en TCP sur le port 80. Il mesure la cadence de réception et le nombre d'octets émit par le programme LabVIEW. Dans la foulée il reçoit ce qui arrive, qu'il suppose être de la même taille que ce qui a été émit. Comme on reçoit le même nombre d'octets on peut donc en déduire que le débit est le double de celui affiché.
Afin d'avoir une résolution suffisante, 1000 paquets sont émis à chaque test.

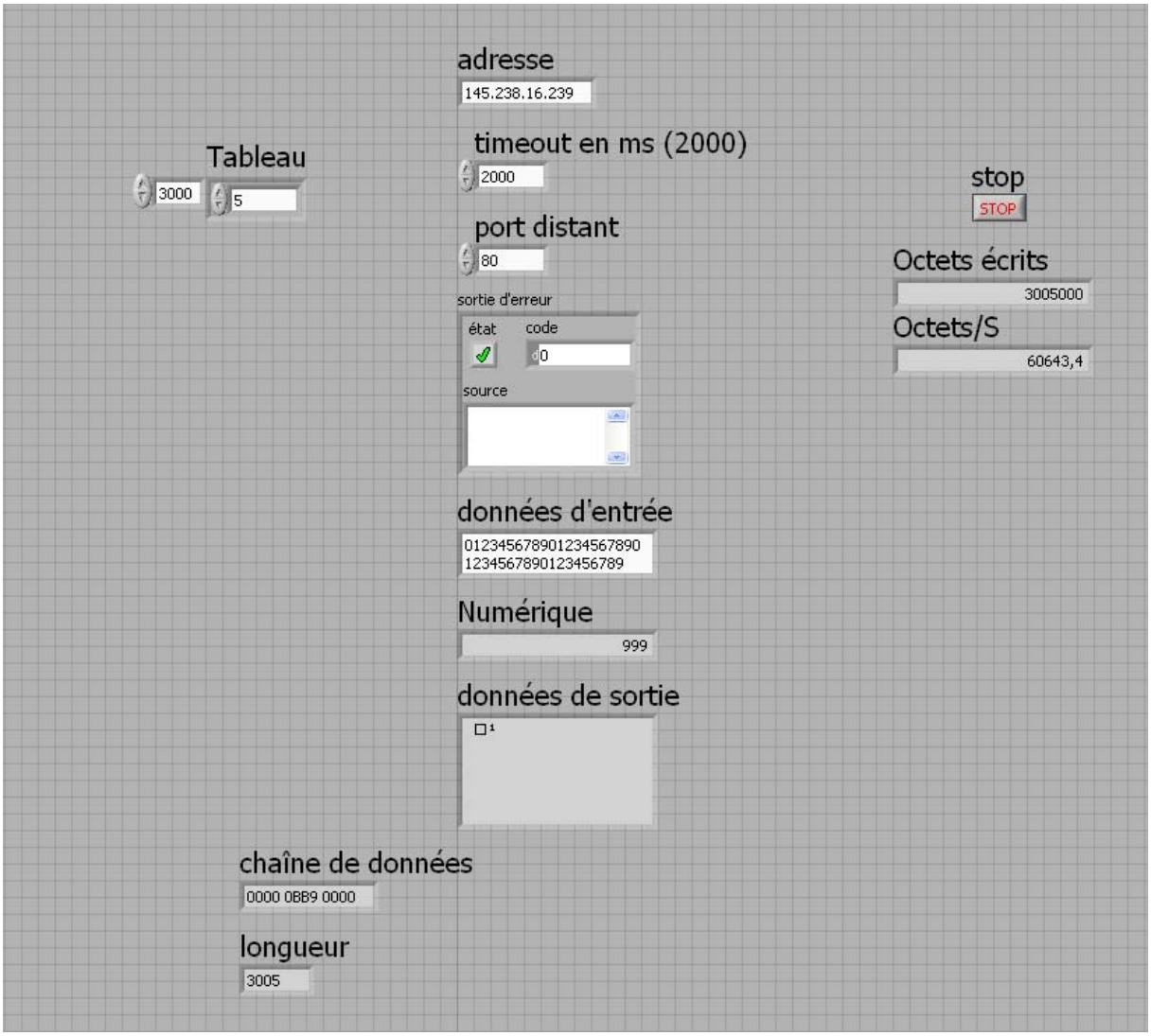

Figure 1 : Face avant

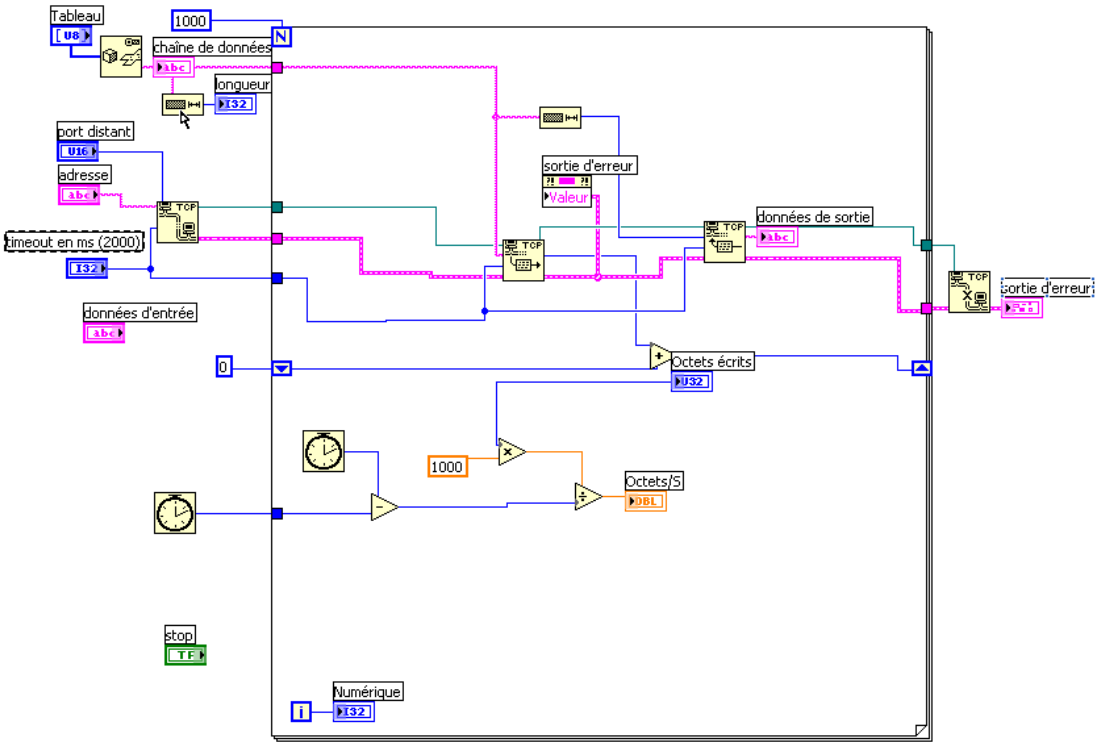

Figure 2 : Diagramme

Résultats

Taille du paquet	Taux de transfert (envoi + réception)
24	15800
54	33200
84	47800
94	52400
104	10600
204	20400
504	49800
1004	91600
2004	95400
3004	121000
4004	115000

Analyse

Au dessus de 100 octets par paquet on entre dans un autre mode de transfert et les taux baissent de nouveau pour augmenter ensuite. Il semble que la taille optimale pour un paquet soit de 3000 octets. Les performances sont alors de 115000 octets /seconde, soit, 1,2 Mbits/sec
La limitation théorique est la bande passante côté microcontrôleur qui est de 10 Mbits/sec. Donc, le taux de transfert est tout à fait respectable pour un tel dispositif.

Conclusion

Voilà un type de communication qui mérite d'être utilisé, non seulement pour les envois de TC et réception de TM mais aussi pour les gros paquets de données. La mise en œuvre côté microcontrôleur est simple, grâce aux exemples, et côté PC, on trouvera sans aucune difficulté, des programmes simples pour envoyer des données et en recevoir en TCP, que ce soit sous LabVIEW comme présenté ici, mais aussi en Borland C++ Builder, Visual Basic, ou autre langage qui met à disposition des librairies Internet.