Laboratori de Telemàtica 1
Memòries de sessions I-V

Lluís Batlle
Edgar Pérez
Guillem Pérez
David Sabaté

21 de febrer de 2003

1 Sessió I

Els terminals són emulats en el PC mitjançant un programa que fa que la pantalla i el teclat mostrin i enviin les dades que s'envien o reben pel port sèrie de l'ordinador. Aquests terminals admeten múltiples paràmetres de configuració, ja siguin relatius a la configuració física (protocol RS-232), o a la d'interpretació dels caràcters de control enviats per la línia sèrie.

1.1 Connexió dels ports sèrie dels PCs

Per a connexionar dos PCs mitjançant el port sèrie, necessitem un cable *null-modem creuat*. Aquest com a mínim creua les línies TX-RX i CTS-RTS, per a tenir enviament-recepció i control de fluxe hardware típic. Amb altres senyals de la línia (DTR-DSR) també es pot fer control de fluxe de hardware, però el software que utilitzem d'emulació de terminal no ho permet.

1.2 Configuració dels terminals

Per fer aquesta pràctica, hem de configurar els terminals de la següent manera:

- Emulació de terminal VT-100: És relatiu a com s'envien els caràcters de control, com *backspace*, o protocol per a moure el cursor de la pantalla, fer scroll, ...
- Eco activat: Mostra per la pantalla el que s'envia per la línia sèrie. Si no tenim aquesta opció activada les tecles que apretem al teclat s'envien directament per la línia sèrie, sense apareixer a la pantalla (que es limita a mostrar el que es rep).
- Transformació CR→ CRLF a l'enviament: La tecla INTRO s'interpreta com un CR, però CR significa *retorn de carro*. Això, a la pantalla, fa que el cursor es situi a l'inici de la línia. Però normalment quan premem INTRO volem dir que el cursor es situi a l'inici de línia, i a més baixi a la línia següent. Aquesta última acció l'efectua el caràcter de control LF. Així, volem que quan premem INTRO, realment s'envii CRLF perquè a la pantalla de l'altre terminal faci un CRLF (igualment que a la nostra, per l'*eco local*).
- Velocitat de transmissió: La mateixa als dos terminals. Perquè les *UART*s tinguin la mateixa velocitat d'enviament que d'interpretació de la línia sèrie.
- Trama: La mateixa als dos bàndols. Necessitem 8 bits de dades si volem enviar caràcters de l'alfabet *Extended ASCII* (caràcters > 127). La paritat ens serveix per a comprovar la rebuda correcta dels caràcters (control d'errors), i el número de bits de stop a 1, per indicar amb un sol bit el final de trama.
- Control de fluxe: Perquè s'enviin dades només quan el terminal receptor estigui preparat per rebre-les. El fixem a hardware als dos terminals, perquè tenim les línies CTS-RTS connectades creuades i les *UART*s ho permeten.

A partir d'aquí, tenim comunicació còmode entre els dos terminals.

1.3 Transferència de dades

Els dos terminals mostren tot el texte que s'envia per les línies (mateixa sortida per pantalla), fent avanços de línia quan cal.

Modificant paràmetres de configuració dels dos terminals (número de bits de dades i paritat), la comunicació no és correcte. Els terminals treballen mostrant caràcters de 8 bits. Quan es reben caràcters de 7 bits, el considerem com un de 8 bits amb el bit de més pes a 0. Els bytes de dades s'envien dins la trama de menys a més pes. Per exemple, tractarem el cas d'enviar una `a` (Bin: `01100001`).

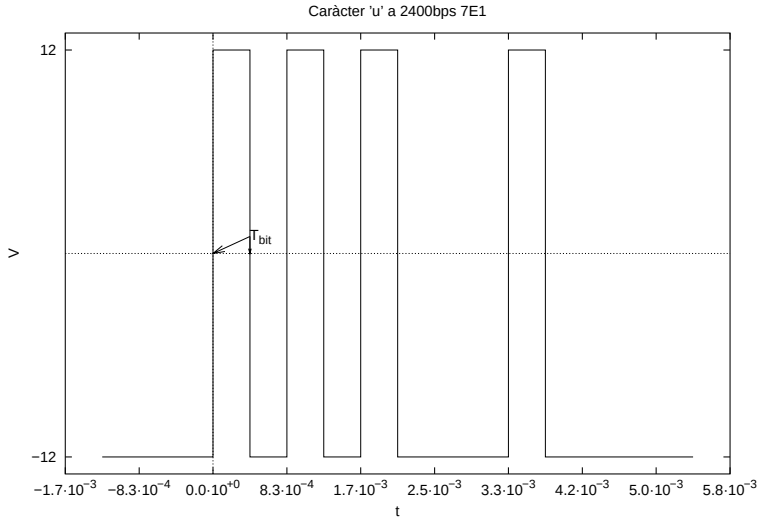
Sense paritat, si enviem amb 8 bits i rebem amb 7, s'enviarà la cadena `0100001101`, essent el primer bit el de *start* i l'últim el de *stop*. El receptor interpretarà que el penúltim bit és el de *stop* (tot i ser un `0`, perquè el software ignora els errors de la *UART* - si és que aquesta els reporta), i el caràcter rebut serà també una `a`, perquè el `0` que s'havia perdut en l'interpretació del bit de *stop* serà assumit al estar en mode de 7 bits.

De l'altre manera, sense paritat, si enviem amb 7 bits i rebem amb 7, s'enviarà la cadena `010000111`. El bit de *stop* a l'enviament serà interpretat al receptor com a bit de dades, i el que realment veurem en pantalla és un caràcter de l'*Extended ASCII* (que depen de la pàgina de codis amb què estem treballant), ja que el byte interpretat té el bit de més pes a `1`: `11100001`.

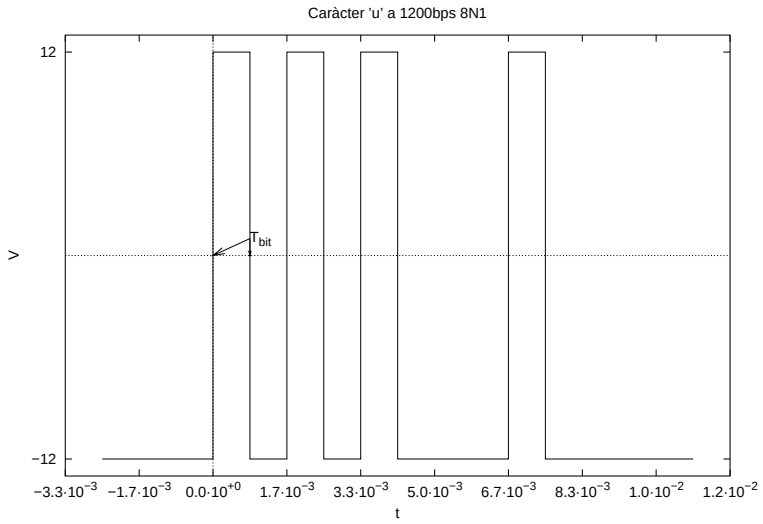
2 Sessió II

Aquí estudiarem els senyals que s'envien per la línia sèrie amb l'oscil·loscopi.

Enviant el caràcter `u` (Bin: `01110101`, tenim la següent forma d'ona a 2400bps, i `7O1`:



Enviant-lo a 1200bps, `8N1`:

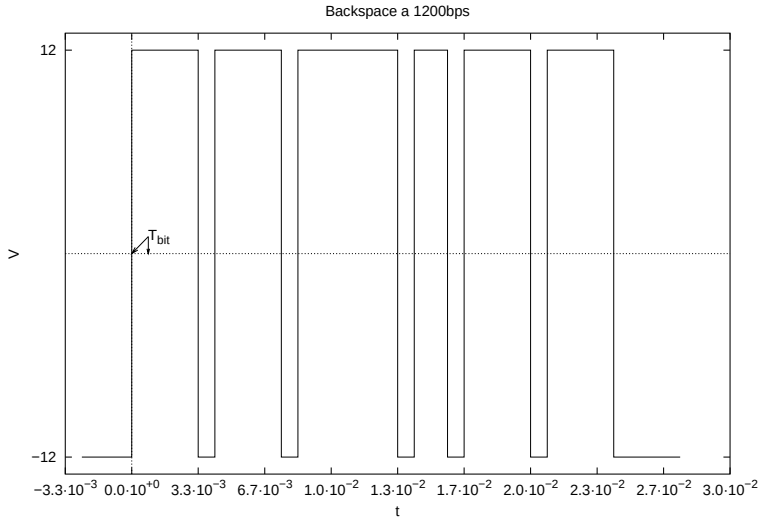


Els dos tenen la mateixa forma d'ona perquè en el primer cas, tenim que el bit de paritat és un 0. I en el segon cas, tenim també un 0 en la mateixa posició, però de dades. Immediatament tenim el bit de stop en els dos casos, perquè en el segon no hi ha paritat.

2.1 Forma d'ona de *backspace*

El *backspace* té una forma d'ona especial perquè quan el premem no s'envia un sol caràcter, sino 3. Primer s'envia `BS`, llavors un espai, i de nou `BS`. Així es borra el caràcter anterior a la posició del cursor, a més del desplaçament de cursor.

La seva forma d'ona a l'oscil·loscopi, enviat a 1200bps a `8N1` és la següent, amb tics a cada 4 bits:



2.2 Control de fluxe

Hem comprovat el control de fluxe per software (XON, XOFF), tan manualment com enviant fitxers de text. Com a detall, només dir que quan s'envia un XOFF, encara s'envien uns quants caràcters més. Això és degut a que el que deixa d'enviar és el programa (i no la *UART*); quan deixa d'enviar el programa, el *buffer* de la *UART* encara té dades, i per tant les envia.

En el cas del control de fluxe per hardware hem vist que el software del terminal no funciona bé. Quan l'emissor baixa CTS, continua fent com si enviés les dades, però sense enviar-les per la línia. O sigui que totes les dades que s'haurien d'enviar en el temps en què la comunicació està aturada (entre desactivació i activació de CTS) es perden.

Per provar mètodes de control de fluxe, el millor és enviar fitxers a través de la línia, a poder ser grans i que necessitin procés al receptor. Així aquest ha de informar a l'emissor que no envii més dades.

3 Sessió III

Hem vist que al iniciar un enllaç X-Modem el podem fer de dues maneres: que es faci control d'errors per Checksum o per CRC. Si l'emissor envia la primera trama després de que el receptor envii una C, tenim mode CRC. I si ho fem després de que envii un NAK, tenim mode Checksum. En principi, per solicitar l'enviament, el receptor envia 3 Cs, i llavors NAKs.

Les trames són com les descrites al *guió de pràctiques*, igual que el tamany del paquet.

3.1 Temps de trames

La durada de les trames és:

- Trama amb CRC: $t_{\text{trama-CRC}} = 4389\,\text{ms}$
- Trama amb checksum: $t_{\text{trama-CS}} = 4336\,\text{ms}$

Els temps entre senyals són:

- Entre fi de trama i reconeixement: $t_{\text{CRC-ACK}} = 170\,\text{ms}$
- Entre reconeixement i nova trama: $t_{\text{ACK-SOH}} = 169\,\text{ms}$

Mentre feiem les mesures de temps amb l'analitzador de protocols ens vam trobar que en un moment ens faltava un byte del CRC. Era perquè estava configurat amb *Supress Idle*, que el que fa és no mostrar el byte `FF`. Per veure'l, vam posar que mostrés tots els caràcters.

3.2 Temps de fases

L'enllaç es fa en tres fases: *fase d'establiment*, *fase de manteniment* i *fase d'alliberament*. La primera és variable, segons el temps que tarda l'emissor a enviar les dades des de que el receptor a començat a donar senyals C/NAK. En el nostre cas vam tenir una fase d'establiment de 1151 ms, amb la qual cosa vam tenir enviament amb CRC.

La *fase de manteniment* també és de temps variable, depenent dels errors (reenviament de trames), i en menor mesura del temps de CPU al processar les trames. En el nostre cas, va ser de 61305 ms.

La *fase d'alliberament* és pràcticament constant ja que només depen del temps de procés de la CPU. Aquesta és des del EOT fins a la confirmació d'alliberament per part del receptor, i en el nostre cas és de 169 ms

3.3 Control d'errors

Els sistemes de control de CRC i Checksum generen 2 o 1 bit respectivament, que són resultats de funcions aplicades a les dades enviades. L'emissor calcula aquests valors i els envia. Llavors, el receptor els torna a calcular per les dades que ha rebut, i ho compara amb els codis de control d'errors rebuts. Si són iguals, retorna un ACK indicant que la recepció ha estat correcte, i si no un NACK, que obliga al emissor a reenviar la última trama.

Per provar el mecanisme de control d'errors, farem talls a les línies d'emissió i recepció. Si obrim la línia d'emissió i la tornem tancar ràpidament, es demana reenviament de trama perquè el receptor no rep el número de bytes esperat (el de la trama). A més, algun cop es perd el sincronisme de la línia sèrie (trames *bit-start* i *bit-stop*). En aquest últim cas, es perd totalment la resta de trama, ja que el receptor interpreta bytes estranys del que veu a la línia.

Si després d'obrir la línia de transmissió no la tanquem, el receptor va enviant NAKs esperant reenviament del paquet (un cop passat el *timeout* de l'espera de bytes de la trama tallada). El temps entre que tallem la línia i el primer NAK depen de l'instant en què tallem la comunicació, i en el nostre cas és de 4150 ms. Si el receptor continua sense rebre res, envia un total de 19 NAKs, i llavors el software dóna error. El temps total dels 19 NAKs és de $11.99\,\text{s} \times 19 = 227.81\,\text{s}$.

Si obrim la línia de recepció i no la tanquem, l'emissor no pot rebre els ACKs. Llavors, el receptor envia NAKs perquè passa massa temps entre el ACK i la següent trama (que no rep perquè l'emissor no l'envia). Entre l'últim ACK, i el següent NAK hi ha un temps de 11975 ms. Quan hi ha un tall de recepció, el receptor arriba a enviar 5 NAKs abans de que l'emissor cancel·li l'enviament. Desde el primer NAK i la cancel·lació tenim un temps de 60 s.

El temps entre que vam confirmar al software que hi havia un error i l'enviament es va cancel·lar és de 300 ms.

4 Sessió IV

El protocol HDLC permet comunicar més de dos dispositius sobre una línia sèrie, ja que les seves trames permeten adreçament. També disposa de control d'errors, i a més és orientat a connexió. Per provar el protocol HDLC, utilitzem un programa que permet experimentar amb el protocol. Dóna llibertat per fer trames, i a més monitoritza l'estat dels comptadors de la finestra i de l'estat de comunicació (amb connexió, sense, ...).

4.1 Configuració del software

- Direcció: a un li assignem la `01` i a l'altre la `03`.
- Mida de la finestra: 7

- Mode de resposta automàtica: tria el comportament del programa. Si està activat, respon automàticament a les peticions de connexió, desconnexió i reenviament de paquets. L'activarem segons convingui a cada pràctica.

4.2 Forma de les trames

Cada trama genèrica té uns camps comuns:

- Flags: Serveixen per marcar l'inici o final de trama
- Direcció: Indica a qui va dirigit el paquet, o qui respon una confirmació (mode comando o resposta). Si un terminal envia una trama amb la seva pròpia direcció està responent.
- Control: Indica quin tipus de trama és.
- Bit P/F: Si aquest bit està a `1`, esperem resposta immediata a la comanda. Sense tenir la resposta, no continuarem enviant res. Les trames de resposta han de tenir aquest bit igual que el de la comanda a que responen.
- SVT o FCS: CRC que serveix per al control d'errors, com al X-Modem.

Les *trames d'informació* tenen més camps:

- $N(s)$: Indica quin número de trama s'envia
- $N(r)$: Indica quin número de trama s'espera rebre

Les *trames de supervisió* tenen codis de supervisió en comptes de $N(r)$, i les *trames no numerades* tenen codis no numerats al lloc de $N(s)$ i $N(r)$.

4.3 Establiment de connexió

Per establir una connexió, primer un ha d'enviar un SABM i l'altre l'ha d'acceptar amb un UA. Podem enviar les trames manualment. En el nostre cas, les trames són:

- SABM: `033F`
- UA: `0373`

Un cop enviat el SABM estem en estat de *inici de connexió*, i si l'altre la confirma, passem a *connexió*. Per a refusar el SABM, l'altre hauria de contestar amb un DM.

Quan hi ha una col·lisió de SABM (els dos l'han enviat alhora) tots dos han de contestar l'UA, perquè sino un d'ells es quedaria en estat de *inici de connexió*.

4.4 Desconnexió

Quan estem en *connexió* i volem desconnectar, hem d'enviar un DISC. L'altre dispositiu ens contestarà amb un DM (o UA) per desconnectar. El DISC situa el que l'ha enviat en estat de *inici de desconnexió*. Quan aquest rep un UA, es desconnecta. L'altre, quan confirma un DISC (rebut en estat de *connexió*) amb un UA, es desconnecta.

És recomanable desconnectar amb DM per això, per solucionar el problema de dos DISC simultanis. Quan es respon amb un DM, el que respon es desconnecta automàticament (estigui en l'estat que estigui). Si hi ha una col·lisió de DISC, els dos estaran amb estat de *inici de desconnexió* (els dos esperant un UA). Quan un respongui un UA, l'altre es desconnectarà i ell no (perquè encara esperarà l'UA de l'altre). Com que estaran desconnectats, l'altre mai respondrà el segon UA (així es justifica el DM).

4.5 Comptadors de finestra

La capa de protocol HDLC de cada dispositiu té dues variables contadores de la posició dins la finestra: $V(s)$ i $V(r)$. Aquestes contenen el número de la pròxima trama que s'enviarà, i el número de la trama que esperem rebre, respectivament. En rebre una nova trama, es comparen els valors de $N(r)$ rebut i el $V(s)$ propi, i el $N(s)$ rebut amb el $V(r)$ propi. Si són correctes (igual que el CRC del paquet), s'incrementa $V(r)$. Si enviem una nova trama, s'envia amb $N(s) = V(s)$ i $N(r) = V(r)$.

Si $N(s)$ i $V(r)$ no coincideixen, es demana reenviament de trames. Si són $N(r)$ i $V(s)$ els que no coincideixen, no es considera error perquè pot ser degut al temps de procés de les trames del receptor. Només ens confirma que ha rebut correctament fins al seu $N(r)$, i esperem la confirmació de la resta més endavant.

4.6 Moviment de finestra

La finestra, tot i tenir 3 bits pels comptadors, no és de 8 sino de 7 trames. Això és perquè quan haguéssim enviat 8 trames, el receptor ens hauria de confirmar que ha rebut bé fins la 8ena. Això només ho podria fer posant el camp $N(r) = 0$, perquè la pròxima trama que espera és 0. Però això és ambigu, perquè també podria significar que espera TOTES les trames que hem enviat, des de la 0. En canvi, si la finestra és fins a 7, la última confirmació és amb $N(r) = 111_{(2} = 7$.

Un cop enviem la última trama hem de posar el *bit de poll* (P/F) a 1 perquè no podem enviar més trames fins que ens en confirmin la recepció. Això suposa una pèrdua d'eficiència. Per tant, el més normal és aprofitar-se del *full duplex* de la línia i anar enviant confirmacions de paquets tot i no tenir la finestra de recepció plena. Així no s'ha d'arribar mai a l'estat final d'espera perquè la finestra es va desplaçant com un buffer circular.

Les confirmacions de recepció es fan a través del $N(r)$ dels paquets enviats pel receptor. Si el receptor no està enviant informació, ho fa enviant trames RR.

5 Sessió V

5.1 Control de fluxe

Aquest s'efectua mitjançant les trames RR (*Receiver Ready*) i RNR (*Receiver Not Ready*). La primera significa que podem continuar enviant trames al receptor, i la segona que hem de parar d'enviar trames perquè el receptor no està disponible. La trama RR es fa servir també per confirmar les trames rebudes correctament mitjançant el seu $N(r)$.

5.2 Control d'errors de transferència d'informació

Quan es rep una trama amb CRC incorrecte, en podem demanar el reenviament mitjançant una trama REJ, amb el $N(r)$ de la primera trama que volem que sigui reenviada. Llavors, a l'emissor li toca tornar a enviar les trames que li demanen (des de l'$N(r)$ rebut fins al seu $V(s)$).

Quan s'envia un REJ d'un número de trama que ja estava confirmat com a correcte, hi ha un error de protocol. Aquests s'expliquen al pròxim apartat.

5.3 Control d'errors de protocol

La trama FRMR indica que s'ha rebut una trama incorrecta (amb desacord amb el protocol). Això passa quan hi ha hagut dessincronització entre emissor i receptor (degut normalment a errors de la implementació del protocol). Aquesta trama permet especificar quina és la trama incorrecta que s'ha rebut. S'aprofita la posició del $N(s)$ del camp de control per posar-hi un ID que identifiqui la trama incorrecta. La informació del FRMR consta de 3 bytes:

1. Camp de control de la trama que ha provocat l'error
2. Conté el $V(r)$ i el $V(s)$ anteriors a l'enviament del FRMR, a més de si la trama rebutjada era de comanda o de resposta
3. Indica quina és la causa de l'excepció

En general, la rebuda del FRMR indica que hi ha un error al $V(s)$ o $V(r)$ del receptor o l'emissor, i per tant es soluciona tornant a iniciar la connexió (ressetejant els comptadors de finestra a 0). Una vegada reiniciades les finiestres, es reenvien totes les trames que s'havien enviat amb la finestra d'abans del reseteig.