

INTERPRÉTATION DES CODES RC5

La gestion des codes RC5 a déjà fait

l'objet de descriptions dans les

colonnes d'Electronique Radio-Plans.

Une première approche avait été

exposée par G. de Dieuleveult dans le

ERP 523 suite aux articles théoriques

de D. Paret.

Les solutions habituelles utilisent un ou deux timers pour mesurer les différents

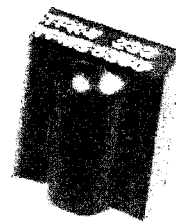
temps nécessaires au décodage. Malheureusement, les timers sont

souvent utilisés à d'autres tâches, le timer 1 servant généralement de base

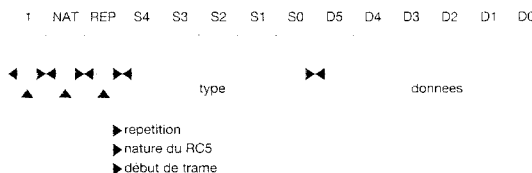
de temps à l'interface série, le timer 0 étant lui utilisé pour gérer les différents

temps système via une routine d'interruption.

L'idée est d'utiliser cette routine d'interruption périodique pour scruter la ligne de réception du code RC5 sans nécessiter l'utilisation exclusive du timer. Nous supposons par contre qu'un timer génère une interruption périodique au moins toutes les 250µs. Nous rappellerons à la **figure 1** la forme des signaux utilisés dans le codage RC5 et en **figure 2** la composition de la trame RC5. Un premier bit, toujours à 1 est le signal de début de trame. Il est suivi d'un bit indiquant la nature du RC5, puis du bit de répétition. Ce bit change d'état à chaque émission, il permet donc de faire la distinction entre la répétition (deux fois consécutives, du même code et la réception du même code séparé par un "trou" dû à une mauvaise réception. Les cinq bits qui suivent déterminent le type d'appareil auquel est destinée la télécommande (téléviseur, magnétoscope, CDI, etc). Suivent enfin les six bits de commande proprement dits.



■ Figure 1 : définition des "0" et "1" logiques en RC5.



■ Figure 2 : constitution d'une trame.

(1) 1 0 0 0 0 1 1 0 1 0 1 0 0

les barres indiquent les créneaux mesurés

■ Figure 3 : représentation physique d'une trame quelconque avec interprétation des bits consécutifs. Les barres indiquent les créneaux mesurés.



On remarquera que le bit de "nature de RC5" constitue en fait un septième bit de donnée, certaines télécommandes générant à la fois des codes standards et étendus. (Le code RC5 initial utilisait 2 bits de start, et ce bit n'était donc pas utilisable)

Si on observe les signaux, on voit qu'il y a une succession de créneaux larges ou étroits. Les créneaux sont larges dans le cas de deux bits successifs différents (1 puis 0 ou 0 puis 1), ils sont étroits si les deux bits successifs sont de même valeur.

L'algorithme permettant le décodage pourra s'exprimer de la manière suivante :

- mesurer la largeur d'un créneau (sans tenir compte de sa polarité)
- si c'est un créneau court, le bit suivant est le même que le bit courant, si c'est un créneau long, le bit suivant est l'inverse du bit précédent.

De plus on appliquera la règle suivante :

- si c'est un créneau court, on ne mesure pas le créneau suivant.

On appliquera cet algorithme jusqu'au dernier bit.

Pour aider le décodage, on sait de plus que le premier bit est à un.

On pourra aussi éliminer les parasites en écartant les trames dans lesquelles il y a des créneaux manifestement trop courts ou trop longs. La **figure 3** montre les créneaux mesurés et la valeur déduite.

Cet algorithme est programmé sur un microprocesseur de la famille 80C51. Le listing suivant montre la routine d'interruption ainsi que la définition des variables.

Le listing complet ainsi qu'un petit exécutable sortant le code décodé sur les ports 1 et 3 du microprocesseur est disponible sur le serveur ERP.

Le hardware est réduit à sa plus simple expression : un circuit de chez Telefunken inclut tout à la fois le phototransistor de réception, le filtrage et la mise en forme des signaux. Il existe plusieurs modèles suivant la fréquence de modulation du signal infrarouge, souvent celui-ci est modulé à 36kHz, c'est donc le modèle TFMS 5360 (disponible chez OMNITECH-SERTRONIQUE) que nous avons utilisé. Ces circuits existent aussi chez SIEMENS. Ce circuit possède trois broches, deux d'alimentation (0, +5V), et une sortie de signal directement compatible avec un microprocesseur, connectée ici sur le port P3.5.

Deux remarques sur ce hard ultra-simple :

- Le signal de sortie est inversé, il faut donc en tenir compte si vous utilisez le programme avec un hardware différent.

- Il faut bien découpler le circuit de Telefunken pour ne pas avoir d'impulsions parasites en sortie. Aussi, utilisez une capacité céramique aux bornes de ses broches d'alimentation.

Ces précautions étant prises, on obtient facilement dans une pièce normalement éclairée des portées d'une dizaine de mètres avec des télécommandes du commerce.

```

RCS_PIN EQU 0B5H ; entrée du code RCS (inverse) P3.5
RCS_MSG: RSEG SEGMENT_DATA ; datas utilisées
RC5_SHF: DS 2 ; message reçu
RC5_CPT: DS 1 ; registre de reception
RSEG SEGMENT_BIT ; bits utilisés
RC5_STA: DBIT 1 ; attend un start
RC5_BIT: DBIT 1 ; numero du bit en cours
RC5_SKP: DBIT 1 ; saute le bit
RC5_NEW: DBIT 1 ; nouveau code
RC5_OUT: DBIT 1 ; saute jusqu'a ce que ligne stable
; définition arbitraire de la position du MSB et du LSB des donnees 16 bits
MSB EQU 0
LSB EQU 1

```

; Gestion de codes RCS par interruption. L'interruption est toutes les 208µs,
; un bit dure 1.778 ms, un demi bit 889 µs, soit le temps de 4.27 interruptions
; La routine d'interruption verra donc la ligne RCS haute ou basse pendant au
; maximum 3 à 5 it pour l'état le plus court, de 7 à 9 it pour le plus long.
; la durée entre deux messages est environ de 80 ms

```

ITEMPS: PUSH PSW
        PUSH ACC
RC5DEC: JB RCS_STA,RC5SKP ; attend un bit de start
; teste si le temps entre deux bits est dépassé
        DJNZ RC5_CPT,RC5SKP ; pas de watch dog, continue
; le temps entre deux bits est dépassé, prépare à recevoir une nouvelle trame
        CLR RC5_OUT
        MOV RC5_SHF+LSB,#00001000B
        MOV RC5_SHF+MSB,#0
        SETB RC5_STA ; saute l'état avant le start
        SETB RC5_BIT ; la ligne au repos est haute
        SETB RC5_SKP ; et skip à 1
        SJMP RC5SK2
; sort directement tant que RCS_PIN est identique a RCS_BIT
RC5SKP: JB RCS_PIN,RC5SK1 ; saute si ligne à 1
        JNB RC5_BIT,RC5SEND ; pas de changement, saute
        SJMP RC5SK0
RC5SK1: JB RC5_BIT,RC5SEND ; pas de changement, saute
RC5SK0: JB RC5_OUT,RC5SK2 ; change le bit courant de signe
        CLR RC5_STA ; si start, attend encore
        JBC RC5_SKP,RC5SK2 ; si skip armé, desarme le bit
        MOV A,RC5_CPT ; prend le compteur de tics
        SETB RC5_OUT
        ADD A,#5
        JC RC5SK3
; long, ne saute pas le suivant, et pousse RCS_BIT
        ADD A,#4
        JNC RC5SK2 ; si trop long, break
        MOV C,RC5_BIT
        CPL C
        SJMP RCSACC
; court, saute le suivant et pousse le complement de RCS_BIT
RC5SK3: ADD A,#3
        JC RC5SK2 ; si trop court, break
        MOV C,RC5_BIT
        SETB RC5_SKP
RCSACC: MOV A,RC5_SHF+LSB ; pousse la retenue dans RC5_SHF (16 bits)
        RLC A
        MOV RC5_SHF+LSB,A
        MOV A,RC5_SHF+MSB
        RLC A
        MOV RC5_SHF+MSB,A
        MOV RC5_OUT,C ; si c'est le dernier, saute apres
        JNC RC5SK2 ; ce n'est pas le dernier, continue
; dernier bit poussé, teste si code identique au précédent
        XRL A,RC5_MSG+MSB
        JNZ RCSNEW ; différent, nouveau code
        MOV A,RC5_SHF+LSB
        XRL A,RC5_MSG+LSB
        JZ RC5SK2 ; identique, resete l'automate au prochain wdog
RCSNEW: SETB RC5_NEW
        MOV RC5_MSG+MSB,RC5_SHF+MSB
        MOV RC5_MSG+LSB,RC5_SHF+LSB
RC5SK2: MOV RC5_CPT,#0 ; met à 0 le compteur de tics
RC5SEND: POP ACC
        POP PSW
        RETI

```

