µChameleon 2
Manuel Utilisateur

Rev 4.0

1. Présentation générale 4
1.1. Fonctionnalités principales 4
1.2. Communication USB 4
1.3. L'interpréteur de commandes 5
1.4. Mises à jour du firmware 5
1.5. Connecteurs externes 6
1. Pour débuter 7
1.6. Installation des drivers 7
1.6.1. Présentation 7
1.6.2. Détection automatique 8
1.6.3. La led d'activité 8
1.6.4. Entrées/Sorties digitales 8
1.6.5. Saisie de commande en direct 8
2. Liste des commandes 9
1.7. Communication: les bases 9
1.7.1. Ouvrir la communication 9
1.7.2. Vérifier que le µChameleon répond 9
1.7.3. Vérifier la version du firmware 10
1.8. Led d'activité 10
1.8.1. Allumer ou éteindre la led 10
1.8.2. Appliquer une séquence (chenillard) 10
1.9. Entrées-sorties digitales 11
1.9.1. Choisir la direction 11
1.9.2. Lire l'état 11
1.9.3. Imposer l'état 11
1.9.4. L'activation du pull-up 11
1.9.5. Monitoring de l'activité d'une pin 12
1.10. Entrées analogiques 13
1.10.1. Lire la tension d'entrée 13
1.11. Sortie analogiques – PWM – Génération de fréquence 13
1.11.1. Applications du PWM 13
1.11.2. Résumé des commandes pwm 14
1.11.3. Démarrer ou arrêter le pwm 14
1.11.4. Fréquence de sortie pwm 14
1.11.5. Rapport cyclique pwm 15
1.11.6. Polarité pwm 15
1.11.7. Prédiviseur pwm 16
1.11.8. Compteur pwm 16
1.12. L'instruction wait : tout est dans le timing 17
1.12.1. Créer une référence de temps zero 17
1.12.2. Attendre un temps spécifié 17
1.12.3. Exemple d'utilisation du wait 17
1.13. Mesures de fréquences et de durées d'impulsions 18
1.13.1. Mesurer la durée d'un cycle complet 18
1.13.2. Mesurer la durée d'une impulsion 18

1.13.3. Changer la polarité de déclenchement 19
1.14. SPI – Serial Peripheral Interface 20
1.14.1. Introduction 20
1.14.2. Activer et désactiver le SPI 20
1.14.3. Changer la vitesse d'horloge 20
1.14.4. Programmer la phase et la polarité 21
1.14.5. Envoi de données 21
1.14.6. Réception de données 21
1.14.7. Programmer l'octet factice 22
1.14.8. Emission-réception simultanées 22
1.15. Support de l'UART 23
1.15.1. Introduction 23
1.15.2. Initialisation 23
1.15.3. Envoi de données 23
1.15.4. Réception de données 24
1.15.5. Notification Asynchrone 24
1.16. Variables et arithmétique 25
1.16.1. Dim 25
1.16.2. Let 25
1.16.3. Increment et decrement 26
1.16.4. Print 26
1.16.5. La variable spéciale '?' 26
1.16.6. Effacer les variables 26
1.17. Exécution conditionnelle 27
1.17.1. If … then et opérateurs relationnels 27
1.18. Gestionnaires d'événements 28
1.18.1. Introduction 28
1.18.2. L'évènement reset 28
1.18.3. L'évènement background 28
1.18.4. Background : choisir la source d'horloge 28
1.18.5. Définir un gestionnaire d'évènement 29
2. Informations sur le matériel (hardware) 30
2.1. Entrées / Sorties 30
2.2. Circuit d'alimentation 30
2.2.1. Présentation 30
2.2.2. Protections du circuit d'alimentation 31
2.2.3. Utilisation d'un bloc secteur 32
2.2.4. Considérations thermiques 32

1. Présentation générale

1.1. Fonctionnalités principales

- Communication USB rapide: 1Moctets/s
- Pilotes fournis pour Windows 98/Me/2000/XP, Linux, MacOS
- Communication en Classe USB CDC
- 8 entrés analogiques avec 12 bits de résolution
- 18 Entrées-Sorties numériques d'usage général
- 4 timers avec sorties PWM (modulation de largeur d'impulsion)
- 4 entrées de mesure de fréquence ou durée d'impulsions
- Interpréteur de commandes embarqué
- Fonctionnement autonome
- Alimentation par l'USB, ou bloc secteur (commutation automatique)
- Borniers à vis pour connection rapide aux circuits extérieurs
- Taille compacte

1.2. Communication USB

Grâce à l'utilisation du standard USB CDC (Communication Device Class) votre µChameleon est reconnu automatiquement par la plupart des systèmes d'exploitation comme périphérique de communication série.
Le port série virtuel ainsi crée, se comporte exactement comme un port série standard. Le µChameleon apparaît dans le panneau de contrôle sous la rubrique « Ports COM et LPT ». Tous les logiciels existants, ou environnements de programmation qui peuvent envoyer des chaînes de caractères sur un port série sont capables de dialoguer avec un µChameleon. Il est même possible d'effectuer des tests simples avec votre émulateur de terminal favori, en tapant les commandes au clavier.

Pour certains systèmes un peu anciens, il peut être nécessaire d'indiquer quel driver existant utiliser, ce qui se fait par un simple fichier texte de type .inf.

1.3. L'interpréteur de commandes

Le logiciel embarqué (firmware) du µChameleon possède un interpréteur de commandes capable d'analyser des ordres en texte pur qui donnent accès à ses ressources matérielles. Le jeu de commandes a été optimisé pour faire face aux applications d'interfaçage avec le monde réel, et sont très simple à mémoriser et à mettre en œuvre, sans que l'utilisateur soit confronté aux détails complexes habituellement rencontrés dans ces applications. Vous pourrez donc vous concentrer sur votre application proprement dite. Fini les interfaces de programmation complexes, avec leurs nombreux paramètres impossibles à mémoriser, et leurs variantes subtiles dont on ne sait laquelle choisir.

1.4. Mises à jour du firmware

Nous travaillons constamment à l'amélioration des capacités du µChameleon, et nous fournissons régulièrement de nouvelles fonctionnalités en fonction des désirs de nos clients. Vous pourrez télécharger ces mises à jour gratuitement sur notre site Internet, et avec notre application sur PC, le « Firmware Upgrader », quelques clics suffisent pour disposer de nouvelles possibilités. En plus, notre système de boot protégé, qui ne peut être effacé accidentellement, implique que vous ne pouvez pas vous retrouver bloqué avec un µChameleon inutilisable. Si quelque chose se passait mal pendant la mise à jour, comme une coupure secteur, ne vous inquiétez pas, il suffit de recommencer.

1.5. Connecteurs externes

Les connecteurs donnent accès à 18 pins d'entrées-sorties. Elles sont toutes individuellement programmables comme entrées ou sorties numériques, et leur état peut être écrit ou lu. De plus, certaines d'entre elles ont des fonctions spécialisées, comme des entrées analogiques, des sorties analogiques, de la génération de fréquence, de la modulation de largeur d'impulsion…
Les numéros de pins ainsi que leurs spécialisations sont indiquées sur l'étiquette du boitier du µChameleon, fournissant une référence rapide et pratique au moment ou vous branchez vos circuits. Voici un résumé des pins d'entrées-sorties avec les spécialisations éventuelles :

Fonctionnalités	Numéros de pins
E/S digitales	toutes
Pull-up sur entrées	9 à 16
Entrées analogiques	1 à 8
Sorties analogiques	9 à 12
Sorties pwm (mli)	9 à 12
Timers	9 à 12
SPI (port série synchrone)	13 à 16
UART (rs-232)	17 et 18

1. Pour débuter

1.6. Installation des drivers

La première fois que vous connecterez votre µChameleon à votre ordinateur, le système d'exploitation installera automatiquement le driver adéquat.
Sur certaines versions un peu anciennes, une boite de dialogue apparaitra, et vous proposera l'installation du driver, ce qui se fera grâce à un simple ficher text .inf présent sur le CD fourni.

1.6.1. Présentation

Sur le CD d'installation qui vous est fourni avec votre µChamelon, vous trouverez une application nommée « µChameleon control », qui va vous permettre d'effectuer avec une interface graphique la plupart des opérations dont est capable votre µChameleon, ainsi que de vérifier qu'il est bien reconnu par votre ordinateur. Si vous en connectez plusieurs, vous pourrez également sélectionner sur lequel les opérations vont s'effectuer.

Ce qui est intéressant, c'est que chaque action effectuée dans l'interface graphique va générer une ou plusieurs chaînes de commandes texte qui seront envoyées au µChameleon, et que celles-ci vont s'afficher à l'écran, avec la réponse dans les cas ou une réponse est prévue. Du coup, cela vous permet en même temps d'apprendre le jeu de commandes et, très rapidement, de comprendre comment s'en servir pour la plupart des taches que vous aurez envie d'accomplir dans vos propres applications.

De plus, une boite de saisie vous permet d'entrez vous-même des commandes texte, qui sont envoyées au µChameleon, et dont vous constatez immédiatement l'effet, ainsi qu'un affichage de la réponse éventuelle.

Pour installer ce logiciel, il vous suffit de lancer le programme "setup.exe" dans le répertoire « Utilities\µChameleon Control » du CD d'installation.

1.6.2. Détection automatique

Au lancement de l'application, le cadre nommé "Device selection" vous montre automatiquement le ou les µChameleon connectés à votre ordinateur, et vous permet de choisir auquel vous souhaitez parler. Bien sur, s'il n'y en a qu'un seul de connecté, il sera pris pas défaut.

1.6.3. La led d'activité

Deux boutons, marqués "led on" et "led off" vous permettent d'allumer et d'éteindre la led d'activité située à coté du connecteur usb du µChameleon. C'est l'une des choses les plus simples que vous puissiez faire avec. Il est aussi possible d'utiliser au nombre sur un octet comme une séquence de type chenillard, qui permet de signaler simplement certains états particuliers sans avoir à regarder l'écran de l'ordinateur. Essayez donc la commande suivante (au clavier) : « led pattern 5 » (saisissez la et appuyer sur enter). Vous devriez voir la led presque tout le temps éteinte, avec voir deux flashes brefs et rapprochés. (la valeur 5 correspond au motif binaire 00000101). Pour revenir à l'état par défaut, vous pouvez taper : « led pattern 254 ».

1.6.4. Entrées/Sorties digitales

L'onglet correspondant affiche une représentation du µChameleon, ou chaque pin d'I/O possède deux objets clickables. L'un d'eux est une boite carrée contenant le lettre I (Input = Entrée) ou O (Output = Sortie) qui permet de fixer la direction de la pin correspondante. L'autre ressemble à une led, et montre l 'état de la pin. Quand la pin est configurée en entrée, la led sera vert foncé pour un état bas, et vert clair pour un état haut, reproduisant l'aspect d'une led éteinte ou allumée. Quand la pin est configurée en sortie, la led sera représentée en rouge sombre à l'état bas, ou rouge clair à l'état haut.

1.6.5. Saisie de commande en direct

Le boite de saisie vous permet de taper directement au clavier des commandes et de les envoyer immédiatement à votre µChameleon. La commande, ainsi que la réponse éventuelle, s'affichent dans la région de logging, afin de garder une trace. Des cases à cocher permettent de choisir le type de fin de chaîne utilisé.

2. Liste des commandes

Cette section décrit l'ensemble des commandes supportées par le logiciel embarqué du µChameleon. Il est possible de tester l'effet de toutes ces commandes grâce a l'application « µChameleon control ». Toute action dans l'interface graphique génère également l'affichage de la commande texte (dans la boite de log) telle qu'elle a été envoyée. Il est également possible de saisir soi-même une commande au clavier et de l'envoyer.

Note : Dans le reste de ce manuel, lorsque l'on parle d'envoyer une commande, cela implique d'envoyer la chaîne de caractères, suivie du caractère LineFeed (Lf), ou RetourCharriot (Cr), ou de la combinaison RetourCharriot+LineFeed (CrLf).

Note: Les commandes à envoyer, sont présentées en italique, comme ceci: *led on,* de même que les réponses.

1.7. Communication: les bases

Votre µChameleon est vu par les application grâce à un 'Port Série Virtuel', ce qui signifie que tout se passe exactement comme si vous communiquiez avec un port série standard. La plupart des environnements de programmation supportent cette méthode, souvent à travers des objets tout faits, ou à travers les appels systemes standard d'accès aux fichiers. Les exemples de code source que vous trouverez sur notre site ont tous étés développés en 'Visual Basic', la transposition à d'autres langages est généralement assez directe.

1.7.1. Ouvrir la communication

Avant d'envoyer effectivement des commandes, il est nécessaire d'ouvrir le port de communication, et la manière exacte pour le faire dépendra de votre environnement de programmation, mais pour ceux qui sont familier de Visual Basic, cela donnera simplement :

```
MSComm1.PortOpen = True
```

1.7.2. Vérifier que le µChameleon répond

Bien que cela ne soit pas toujours nécessaire, il est possible de vérifier que le µChameleon est bien à l'écoute de vos commandes et en mesure de répondre, en envoyant la commande *id* (pour identification), à laquelle il répondra par ces deux mots : *id µChameleon2*.

1.7.3. Vérifier la version du firmware

Il est possible de vérifier quelle version du firmware est actuellement présent dans le µChameleon avec la command suivante :

firmware -> firmware 4.0

1.8. Led d'activité

A coté du connecteur USB du µChameleon, se trouve une led qui s'allume dès la mise sous tension, avec un petit clignotement qui indique que le firmware a démarré, et attend vos ordres. Il est possible d'agir sur cette led par logiciel.

1.8.1. Allumer ou éteindre la led

Allumer la led:
led on
led 1

Eteindre la led:
led off
led 0

1.8.2. Appliquer une séquence (chenillard)

led pattern <n>

La led va être pilotée selon une séquence de 8 états, on (allumée) ou off (éteinte), chaque état correspondant à un bit de l'octet de paramètre <n>. Par exemple, si vous voulez que la led soit la majeure partie du temps éteinte, avec 3 petits flashes, la valeur du paramètre sera: 21 (1 + 4 + 16).

Essayez donc: *led pattern 21*

1.9. Entrées-sorties digitales

1.9.1. Choisir la direction

Mettre la pin n en entrée:

pin <n> input
pin <n> in

Mettre la pin n en sortie:

pin <n> output
pin <n> out

1.9.2. Lire l'état

Lire l'état de la pin n:

pin <n> state -> pin <n> 0 | 1

1.9.3. Imposer l'état

Mettre la pin n à l'état haut:

pin <n> high
pin <n> hi

Mettre la pin n à l'état bas:

pin <n> low
pin <n> lo

1.9.4. L'activation du pull-up

Activer le pull-up sur la pin n:

pin <n> pullup on
pin <n> pullup 1

Désactiver le pull-up sur la pin n:

pin <n> pullup off
pin <n> pullup 0

Note: La valeur typique de résistance de pull-up est d'environ 47kOhms.

1.9.5. Monitoring de l'activité d'une pin

Cette méthode alternative, également appelé 'notification asynchrone', permet de savoir qu'une pin a changé d'état sans avoir à mettre en place un timer qui va régulièrement demander l'état de la pin.

Activer le monitoring:

pin <n> monitor on | 1
pin <n> mon on | 1

Désactiver le monitoring :

pin <n> monitor off | 0
pin <n> mon off | 0

Quand le monitoring est actif, le µChameleon va constamment surveiller les transitions sur la pin sélectionnée, et envoyer, à chaque fois que l'état change, la même chaîne de caractères que celle qui serait renvoyée si on avait demandé l'état explicitement.

Par exemple, si vous voulez surveillez la 3, vous envoyez:
pin 3 monitor on
et à chaque transition bas vers haut détectée, vous recevrez:
pin 3 1
ou à chaque transition haut vers bas détectée, vous recevrez:
pin 3 0

Le monitoring est supporté sur l'ensemble des 18 pins, et peut être actif simultanément sur n'import quelle combinaison d'entre elles.

1.10. Entrées analogiques

1.10.1. Lire la tension d'entrée

Lire la tension sur la pin n:

adc <n> -> *adc <n> <v>*

Les pins 1 à 8 supportent la conversion analogique vers digital d'une tension 0-5volts avec 12 bits de résolution. Le firmware répond en répétant la partie *adc <n>,* ce qui facilite l'analyse de la réponse, en particulier lorsque plusieurs réponses sont attendues et qu'il pourrait y avoir ambiguïté. La valeur de n sera comprise dans l'intervalle [0;4095] avec 0 pour 0volts et 4095 pour 5volts.

1.11. Sortie analogiques – PWM – Génération de fréquence

1.11.1. Applications du PWM

Les 4 timers de votre µChameleon sont capables de générer des signaux dont on peut contrôler la fréquence et le rapport cyclique, et peuvent être utilisés à différentes fins, notamment la génération de tensions analogiques et de formes d'ondes.

Un simple filtre R-C permet de générer une tension stable, et en modifiant par programme la valeur correspondante à intervalles réguliers, on peut aussi générer des formes d'ondes arbitraires.

Il est donc très simple de générer des tensions de contrôle, par exemple pour réaliser une alimentation programmable, ou piloter un vco, bref, tout ce qui peut être contrôlé en tension.

Il est aussi possible d'utiliser les sorties pwm sans filtrage, par exemple pour contrôler la luminosité d'une led, ou de piloter directement un transistor ou un pont de puissance, afin de contrôler la vitesse et la direction d'un moteur à courant continu.

1.11.2. Résumé des commandes pwm

Voici un résumé des commandes disponibles:

pwm <n> on
pwm <n> off
pwm <n> period
pwm <n> width
pwm <n> polarity
pwm <n> prescaler
pwm <n> counter

Les sections suivantes fournissent des détails concernant l'utilisation de ces commandes.

Note: ces fonctions sont disponibles sur les pins 9-10-11-12.

1.11.3. Démarrer ou arrêter le pwm

Démarrer le pwm sur la pin n:

pwm <n> on

Arrêter le pwm sur la pin n:

pwm <n> off

Note: ces commandes peuvent n'être envoyées qu'une seule fois au début et à la fin de l'application, toutefois cela peut être une bonne idée de programmer les divers paramètres de fréquence et de rapport cyclique avant d'activer effectivement le pwm.

1.11.4. Fréquence de sortie pwm

Régler la période du signal sur la pin n:

pwm <n> period <p>
pwm <n> per <p>

Le paramètre de période s'exprime en unités de l'horloge maître, sur un intervalle [0;65535]. La fréquence nominale est 24MHz. Par exemple, *pwm 9 period 1000* va programmer le timer de la pin 9 pour générer une fréquence de 24kHz.

Note: ceci est vrai à moins que vous n'ayez changé la valeur du prédiviseur d'horloge du timer – voir la section 1.12.7.

1.11.5. Rapport cyclique pwm

Fixer le rapport cyclique de la pin n:

pwm <n> width <w>
pwm <n> wid <w>

Le paramètre width (largeur) s'exprime en unités de l'horloge maître, sur un intervalle [0;65535]. Par exemple, si vous voulez générer un signal carré à 10kHz avec un rapport cyclique de 30%, vous allez envoyer:

pwm 9 period 2400
pwm 9 width 720

Par ailleurs, si vous souhaitez générer un signal de fréquence variable, mais avec un rapport cyclique constant de 50%, en supposant que la période nécessaire se trouve dans *mavariable*, vous allez envoyer:

pwm 9 period mavariable
pwm 9 width mavariable /2

Note: ceci est légèrement simplifié, car il s'agit d'envoyer la chaîne représentant la valeur contenue dans la variable, en fonction du langage de programmation utilisé, et non le texte 'mavariable' lui-même, bien entendu. De plus, il ne faut pas oublier d'activer le pwm si c'est la première fois qu'on envoie la commande.

1.11.6. Polarité pwm

Il est possible de contrôler la polarité du signal logique sur chaque canal pwm, ce qui va affecter la signification du paramètre *width*. Un rapport cyclique de 30% devient maintenant 70%.

Polarité normale: (par defaut)

pwm <n> polarity 0
pwm <n> pol 0

Polarité inversée:

pwm <n> polarity 1
pwm <n> pol 1

Note: Les canaux pwm 9 et 10, ainsi que 11 et 12 partagent leur horloge (paramètre period), ce qui rend très simple la génération de signaux complémentaires, en programmant les deux canaux de manière identique et en changeant simplement la polarité de l'un des deux.

1.11.7. Prédiviseur pwm

Les paramètres de période et de largeur d'impulsion sont tous deux exprimés en unités de conteur, avec une fréquence par défaut de 24MHz, soit une résolution d'environ 42 nanosecondes (41.667ns). Cela signifie que la fréquence la plus basse par défaut est de 366Hz avec prescaler à 1. Il est possible de diviser la fréquence maître de départ par les valeurs suivantes: 1,2,4,8,16,32,64,128. (valeur par défaut : 1).

pwm <n> prescaler <p>
pwm <n> pre <p>

1.11.8. Compteur pwm

Il est possible de connaître la valeur en cours du compteur d'un canal pwm avec la commande suivante :

pwm <n> counter -> pwm <n> counter <c>
pwm <n> cnt

1.12. L'instruction wait : tout est dans le timing

Lorsque l'on interagit avec le monde physique, il est souvent nécessaire de le faire avec un timing précis, en maitrisant le temps auquel certaines actions seront effectués. Il est possible, avec cette instruction, d'effectuer des actions classiques, comme mettre une pin de sortie à l'état haut ou bas, en controllant l'intervalle de temps séparant ces actions.
Cela se fait en définissant en premier lieu un temps zéro, qui servira de référence par la suite. Dans un deuxieme temps, en enverra une suite de commandes qui alterneront les commandes d'entrées-sorties avec des commande wait.
Il faut préciser que le fonctionnement de l'instruction wait diffère d'une pause en ce sens qu'elle ne dépend pas du temps d'execution des autres instructions d'une séquence. En effet, le paramètre de temps d'une instruction wait fait toujours référence au temps écoulé par rapport à l'instruction wait précédente.

1.12.1. Créer une référence de temps zero

L'instruction suivante permet de définir un temps zéro qui servira de référence pour les instructions suivantes, ainsi que le canal de timer qui y sera associé :

wait init <canal>

1.12.2. Attendre un temps spécifié

L'instruction suivante permet d'attendre qu'un certain temps se soit écoulé, par rapport à l'instruction wait précédente :

wait time <temps>

1.12.3. Exemple d'utilisation du wait

Dans le logiciel de test 'µChameleon Control', on chargera le fichier de démo nommé 'wait pulses.txt'. La séquence d'instructions qui s'y trouve produit trois impulsions d'une durée de 1ms, séparées elles-mêmes d'intervalles de 1ms (donc une impulsion toutes les 2ms).

1.13. Mesures de fréquences et de durées d'impulsions

Les 4 timers du µChameleon ont la possibilité d'effectuer des mesures de fréquence ou de durée d'impulsion sur des signaux appliqués à leurs entrées (pins 9 à 12).
La logique de détection de front peut être configurée de sorte à être activée sur front montant (passage de l'état bas à l'état haut – valeur par défaut) ou descendant (état haut vers état bas).
La résolution par défaut de ces mesures est de 42ns (horloge à 24MHz), et comme cette valeur dépend des prescalers des timers, elle peut être modifiée à l'aide de la commande 'pwm prescaler'.

1.13.1. Mesurer la durée d'un cycle complet

L'instruction suivante va attendre une transition sur le signal d'entrée correspondant, puis utilisera le hardware du timer pour compter le nombre de cycles écoulés, en attendant la prochaine transition de même nature. A ce moment, le nombre de cycles de timer écoulé est reporté.

timer <pin> capture cycle

Le µChameleon répond par :

timer <pin> capture cycle <compte>

1.13.2. Mesurer la durée d'une impulsion

L'instruction suivante va attendre une transition sur le signal d'entrée correspondant, puis utilisera le hardware du timer pour compter le nombre de cycles écoulés, en attendant la prochaine transition de signe opposé. A ce moment, le nombre de cycles de timer écoulé est reporté.

timer <pin> capture pulse

Le µChameleon répond par :

timer <pin> capture pulse <compte>

1.13.3. Changer la polarité de déclenchement

L'instruction suivante permet de modifier la polarité du front servant au déclenchement de la fonction de capture de durée de cycle ou d'impulsion.

timer <pin> capture edge <polarité>

Pour le paramètre de polarité, 0 correspond à un front descandant, 1 correspond à un front montant (valeur par défaut).

1.14. SPI – Serial Peripheral Interface

1.14.1. Introduction

Le SPI du µChameleon implémente une liaison série à 3 fils synchrone qui est très répendue, en particulier dans de nombreux circuits périphériques. Le µChameleon est toujours en mode maître lors de l'utilisation de ce port de communication.

Les signaux du port SPI sont affectés de la manière suivante:

pin 13 : SCK (Serial Clock)
pin 14 : MOSI (Master Out Slave In – TX)
pin 15 : MISO (Master In Slave Out – RX)

1.14.2. Activer et désactiver le SPI

Avant de pouvoir utiliser le SPI, il faut l'activer car les pins sont partagées avec les pins d'E/S normales 13, 14 et 15.

Syntaxe: *spi on*
spi off

Désactiver le SPI permet de retrouver l'usages des pins correspondante comme entrées-sorties d'usage général.

1.14.3. Changer la vitesse d'horloge

L'horloge du SPI peut être modifée par un circuit prescaler, qui divise l'horloge maître (24MHz) du µChameleon, par les valeurs of 2, 4, 8, 16, 32, 64, 128, ou 256.
La valeur par défaut est de 2, ce qui donne une transmission à 12Mbits/s.

Syntaxe pour la programmation du prescaler:

spi prescaler <p>
spi pre <p>

Example:

spi prescaler 32 (fournit une horloge à 750kHz)

1.14.4. Programmer la phase et la polarité

Afin de s'adapter à la grande variété de périphériques existants, il est possible de programmer la phase et la polarité du SPI avec les commandes suivantes :

Polarité normale (normalement bas, actif au niveau haut – par défaut)
spi polarity 0
spi pol 0

Polarité inversée (normalement haut, actif au niveau bas)
spi polarity 1
spi pol 1

Phase normale (front d'horloge au milieu du bit de donnée – par défaut)
spi phase 0
spi pha 0

Phase inversée (horloge et donnée changent simultanément)
spi phase 1
spi pha 1

1.14.5. Envoi de données

La transmission d'une donnée est effectuée avec la commande suivante:

spi out <octet>

Dès reception de la commande, le SPI commence à décaler l'octet sur la pin MOSI et le signal d'horloge SCK synchronise la sortie.

1.14.6. Réception de données

La réception d'un octet d'effectue avec la commande suivante :

spi in -> spi in <octet>

Dès que la commande est recue, le SPI démarre un cycle, effectuant des impulsions sur la sortie SCK, et recevant sur MISO les bits un à un. Après 8 cycles, le µChameleon renvoie l'octet correspondant à cette réception.

1.14.7. Programmer l'octet factice

Du fait du fonctionnement convenu sur les liaisons de type SPI, lors de la reception d'un octet sur, la sortie MOSI reste au niveau bas, ce qui correspond à l'envoi d'un octet zéro.
Il est possible de changer ce comportement par défaut avec la commande suivante :

spi dummy <octet>

1.14.8. Emission-réception simultanées

Il est possible de combiner la transmission et la reception de deux octets dans le même cycle avec la commande suivante.

spi swap <octet émis> -> *spi in <octet recu>*

Lorsque le µChameleon recoit la commande SPI de swap (échange), un cycle SPI d 'émission commance, et pour chaque bit émis sur la broche MOSI, un bit est recu sur la broche MISO, le décalage étant simultané.

1.15. Support de l'UART

1.15.1. Introduction

Le µChameleon dispose d'un port série de type UART, qui est similaire aux ports série standards que l'on trouve sur un grand nombre de machines informatique, comme le PC, ou certains instruments de mesure.
Les signaux de l'UART du µChameleon sont des niveaux logiques 0-5Volts, il peut donc être nécessaire de les adapter avec une électronique apropriée si l'on veut dialoguer avec une liaison RS-232. Le firmware 4.0 supporte uniquement le contrôle de flux matériel (DTR/CTS).

1.15.2. Initialisation

L'Uart est initialisée après un reset à une communication par défaut à 9600 bauds, 8 bits, 1 bit de stop, pas de parité, et contrôle de flux sur les pins 16 (sortie-RTS) et 15 (entrée-CTS). TX est la pin 18, RX la pin 17.

Afin d'activer l'uart, ainsi que les signaux de contrôle de flux, la commande suivante sera utilisée :

uart on

Afin de désactiver l'uart, et de retrouver l'usage des broches correspondantes pour un usage générique, on utilisera la commande suivante:

uart off

1.15.3. Envoi de données

Des données peuvent être envoyées avec la commande suivante:

uart send <n> <…données…>

ou <n> est une chaine de caractères représentant le nombre d'octets à envoyer, suivi d'un espace, puis des octets devant être effectivement envoyés. La présence de ce nombre permet l'envoi de données quelconque, c'est à dire pas seulement de textes ou caractères standard, mais de n'importe quelles valeurs binaires sur 8 bits.

Example: *uart send 5 hello*

Note: cette commande ne nécessite pas de délimiteur à la fin.

1.15.4. Réception de données

Pour recevoir des données, on utilisera la commande suivante:

uart receive

Si le µChameleon n'a pas revu de données dans son buffer interne, il répondra simplement par:

uart receive 0

Le reste du temps, le µChameleon formera sa réponse en utilisant les mots-clés 'uart receive', suivi d'un espace, puis d'un nombre indiquant la quantité d'octets prêts à être lus, et enfin les octets recus eux-mêmes.
Le format utilisé est semblable à celui utilisé pour l 'émission.
Les données peuvent contenir du text lisible ou des données binaires.

Example:

uart receive 5 hello

Note: un caratère de terminaison sera envoyé après la commande, mais la valeur indiquant le nombre d'octets à recevoir n'inclut que les données elle-mêmes, afin de garantir la possibilité d'une récetion exacte et fidèle du flux d'octets recus (pas de filtrage).

1.15.5. Notification Asynchrone

Il est parfois souhaitable d'être informé de la réception de caractères, sans avoir à effectuer de requètes de manière répétitive et régulière à l'aide de la commande 'uart receive'.

L'activation de cette fonctionnalité s'effectue avec la commande suivante :

uart notify <octets>

A chaque fois que le buffer interne à accumulé une reception d'au moins un certain nombre d'octets (paramètre <octets>), le µChameleon procède automatiquement à l'envoi des données en utilisant la syntaxe de récéption de données décrite dans la section précédente.

Si nécessaire, il est possible de revenir à la réception par demandes explicites avec la commande suivante :

uart notify 0

1.16. Variables et arithmétique

1.16.1. Dim

L'instruction dim crée une nouvelle variable, mémorise son nom et son type, et lui alloue de la mémoire. Le nom de la variable ainsi que son type sont stockés de manière permanente dans la mémoire flash, en revanche la valeur est perdue lors de la mise hors tension.

Syntaxe: dim <mavariable> as <type>

Exemple: *dim mavar as int*

Notes: Il est possible de définir jusqu'à 32 variables. Les noms de variables peuvent avoir jusqu'à 12 caractères. Seul le type 'int' (entier 32 bits signé) est défini dans le firmware 4.0.

1.16.2. Let

L'instruction let affecte une valeur à une variable, le coté droit du signe égal étant une variable ou constante, ou une opération arithmétique combinant deux variables ou constantes.

Syntaxe: let <variable> = <valeur>
let <variable> = <valeur> <opérateur> <valeur>

Exemples: *let x = 0*
let compteur = compteur + 3
*let vitesse = vitesse * accel*

Opérateurs disponibles:

opérateur	calcul effectué
+	addition
-	soustraction
*	multiplication
/	division
%	modulo

1.16.3. Increment et decrement

Au lieu d'écrire: *let var = var + 1*
On peut écrire: *increment var*
Ou: *incr var*

Au lieu d'écrire: *let var = var - 1*
On peut écrire: *decrement var*
Ou: *decr var*

Cela est généralement plus pratique, génère une code plus compact, et aussi plus rapide.

1.16.4. Print

Il est possible de consulter la valeur d'une variable avec la commande print.

Syntaxe: *print <mavariable>*

Cette commande retourne la valeur de la variable vers le port de communication USB.

1.16.5. La variable spéciale '?'

Certaines commandes, comme 'adc' ou 'pin', ont leur dernière valeur stockée dans une variable spéciale qui peut être consultée grâce au ' ?'.

Exemple: *adc 1*
let voltage = ?

la variable 'voltage' (nous supposons qu'elle a été crée précédemment avec un instruction 'dim') contient désormais le résultat de la dernière conversion analogique vers numérique.

Note: le signe '?' peut être utilisé dans toutes les commandes en lieu et place d'une variable ou d'une constante.

1.16.6. Effacer les variables

Il est possible d'effacer toutes les variables, leurs définitions, et de libérer les emplacements mémoire qu'elles occupent avec la commande erase :

Syntaxe: *erase dims*

1.17. Exécution conditionnelle

1.17.1. If … then et opérateurs relationnels

L'instruction if permet l'exécution conditionnelle d'autres commandes en fonction du résultat de comparaisons entre valeurs.

La forme générale est la suivante:

if <valeur A> <opérateur> <valeur B> then <command>

Les valeurs peuvent être des variables ou des constantes. L'opérateur peut porter sur une des comparaisons suivantes:

opérateur	comparaison effectuée
=	égal à
<>	non égal à
>	strictement supérieur à
>=	supérieur ou égal à
<	strictement inférieur à
<=	inférieur ou égal à

Exemples:
if compteur >= 1024 then let compteur = 0
if tension < 128 then pin 3 low
if alerte = 1 then led pattern 5

1.18. Gestionnaires d'événements

1.18.1. Introduction

Les gestionnaires d'événements permettent d'exécuter une suite d'instructions lorsqu'un événement spécifique se produit.
Ils sont comparables à la notion de fonction ou de procédure de la plupart des langages de programmation, à ceci près qu'ils ne reçoivent pas de paramètres.

Il existe actuellement deux évènements définis: 'reset' et 'background'.

1.18.2. L'évènement reset

Comme son nom l'indique, l'évènement reset se produit à chaque fois que le µChameleon démarre, c'est-à-dire à réception de la commande 'reset', suite à l'enfoncement du bouton de reset, ou lorsque l'on vient de le mettre sous tension.

1.18.3. L'évènement background

L'évènement background (qui signifie arrière-plan ou tâche de fond) est généré automatiquement de manière périodique par un oscillateur interne fonctionnant à une fréquence d'environ 20Hz. Il peut être actif ou inactif (la valeur par défaut après un reset est inactif).

Syntaxe pour activer le générateur interne:
background on
back on

Syntaxe pour désactiver le générateur interne:
background off
back off

1.18.4. Background : choisir la source d'horloge

Il est possible d'utiliser la fréquence produite par un des 4 timers pwm afin de servir d'horloge pour l'événement 'background', au lieu de l'horloge interne à 20Hz par défaut, avec la commande suivante :

background clock <n>
back clk <n>

On aura au préalable configure un canal pwm pour produire la fréquence requise à l'aide des commandes 'pwm' appropriées.

1.18.5. Définir un gestionnaire d'évènement

Syntaxe pour définir un gestionnaire d'évènement:

onevent <nom de l'évènement>
<instruction ligne 1>
<instruction ligne 2>
.
.
.
<instruction ligne n>
endevent

Exemple:

onevent reset
pin 2 output
background on
endevent

onevent background
adc 1
if ? > 135 then pin 2 low
if ? < 126 then pin 2 high
endevent

Note: L'exemple précédent montre à quel point il est facile d'implémenter un contrôleur de température avec hystérésis.

2. Informations sur le matériel (hardware)

2.1. Entrées / Sorties

Les entrées de type CMOS présentent une impédance très élevée (typiquement supérieure à 10Mohms.)
Toutes les E/S possèdent une limitation de courant. Ceci permet une connexion directe à des LEDs, des opto-coupleurs, des transistors de puissance, des relais miniatures, des buzzers piezo et des petits haut parleurs… avec un courant de sortie typique de 20mA.

2.2. Circuit d'alimentation

2.2.1. Présentation

- Alimentation à partir de l'USB
- Alimentation à partir d'un bloc secteur
- Alimentation à partir d'un +5 Volts extérieur
- Commutation automatique des sources d'alimentation
- Fournit l'alimentation à vos circuits extérieurs
- Multiples dispositifs de protection assurant une haute fiabilité

Le circuit d'alimentation du µChameleon est extrêmement flexible, et a été conçu pour répondre aux exigences du plus grand nombre d'applications possible.
Tout d'abord, il peut s'alimenter directement à partir du port USB de votre ordinateur, ce qui est la configuration par défaut que la plupart des utilisateurs choisissent. Dans certaines situations, toutefois, par exemple lorsqu'on utilise un hub qui n'est pas capable de fournir assez de courant car ne disposant pas de sa propre alimentation, ou bien pour économiser la batterie d'un ordinateur portable, ou bien si l'on veut une tension d'alimentation très stable pour les applications analogiques, le µChameleon possède son propre régulateur linéaire intégré.

2.2.2. Protections du circuit d'alimentation

Le circuit d'alimentation est protégé contre les situations suivantes:

- Inversion de polarité du bloc secteur
- Court circuit sur les borniers d'alimentation de la carte
- Consommation excessive supérieure à 500mA (protège votre pc)
- Température excessive du régulateur 5Volts linéaire

2.2.3. Utilisation d'un bloc secteur

Il suffit de connecter n'importe quel bloc secteur, capable de fournir une tension de sortie comprise entre 9Volts et 12Volts, au connecteur prévu cet effet à coté de la prise usb. La tension externe sera régulée à 5Volts par le régulateur interne, et le µChameleon se commutera automatiquement sur cette source de tension propre et stable.

2.2.4. Considérations thermiques

Quand on alimente par transformateur externe, la région proche du connecteur d'alimentation peut chauffer légèrement. Ceci est tout à fait normal. Toutefois, il est suggéré de ne pas dépasser une tension externe de 16Volts, en particulier en cas de forte consommation, car cela augmente la chaleur que doit dissiper le régulateur interne.