Introduction à la programmation sur le microcontrôleur Propeller

Avant de commencer...

Discussion sur les différents types de processeur utilisés actuellement

Microprocesseur

- Intel, AMD, Freescale
- Nécessite de la mémoire et des périphériques
- De plus en plus : multi-processeur
- Fonctionne avec un système d'exploitation
- Peu utilisé dans les systèmes embarqués
- Tendance vers des processeurs faible puissance (Atom) ayant des performances similaires à certains microcontrôleurs

Microcontrôleur

- ARM, Atmel, Texas Instrument, NXP
- Mémoires et périphériques embarqués
- Microcontrôleurs de plus en plus puissant (cellulaire intelligent, mini-portable)
- Faible puissance et faible coût
- Programmation principalement avec compilateurs C et langage assembleur
- Nécessite des connaissances “Hardware”

FPGA

- Xilinx, Lattice, Altera, Atmel
- Très flexible et performant
- Programmation en langage HDL (Hardware Description Language) i.e. VHDL et Verilog
- Possibilité d'embarquer des processeurs “Soft”
- Courbe d'apprentissage abrupte
- Temps de développement longs

Parallax Propeller (P8X32A)

- Contient 8 processeurs 32 bits identiques (Cog)
- Deux langages officiels : Spin et ASM
- Outils de développement gratuits
- Ne nécessite pas de programmeur
- Bien documenté et bon support (Forums)
- Grande base de pilote existante

Inconvéviants

- Pas de débogueur pas-à-pas
- Apprendre un langage non standard
- Langage C non supporté nativement
- Pas d'interruptions
- Pas de périphériques
- Mémoire limitée
- Coût relativement élevé

Architecture du Propeller

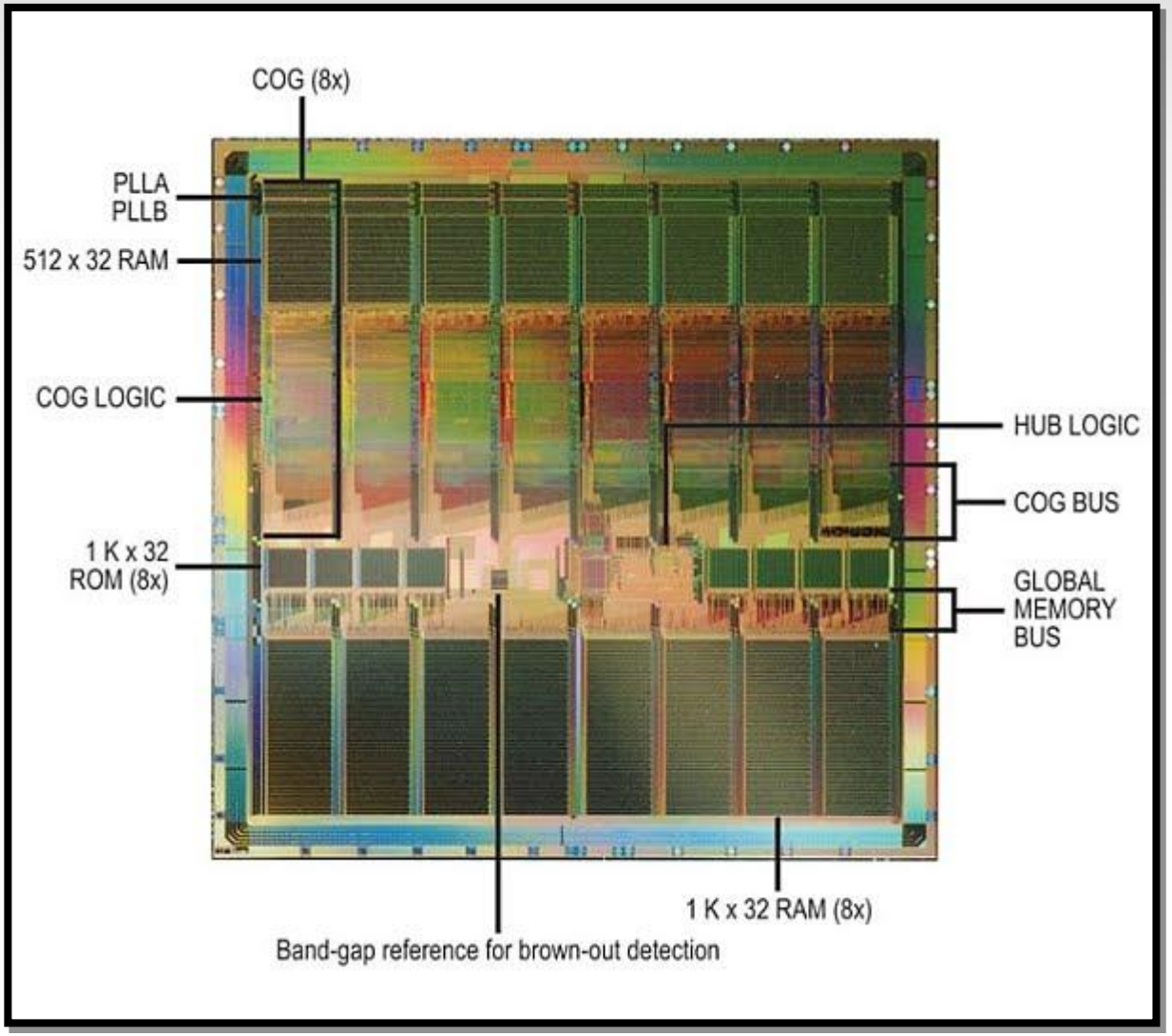
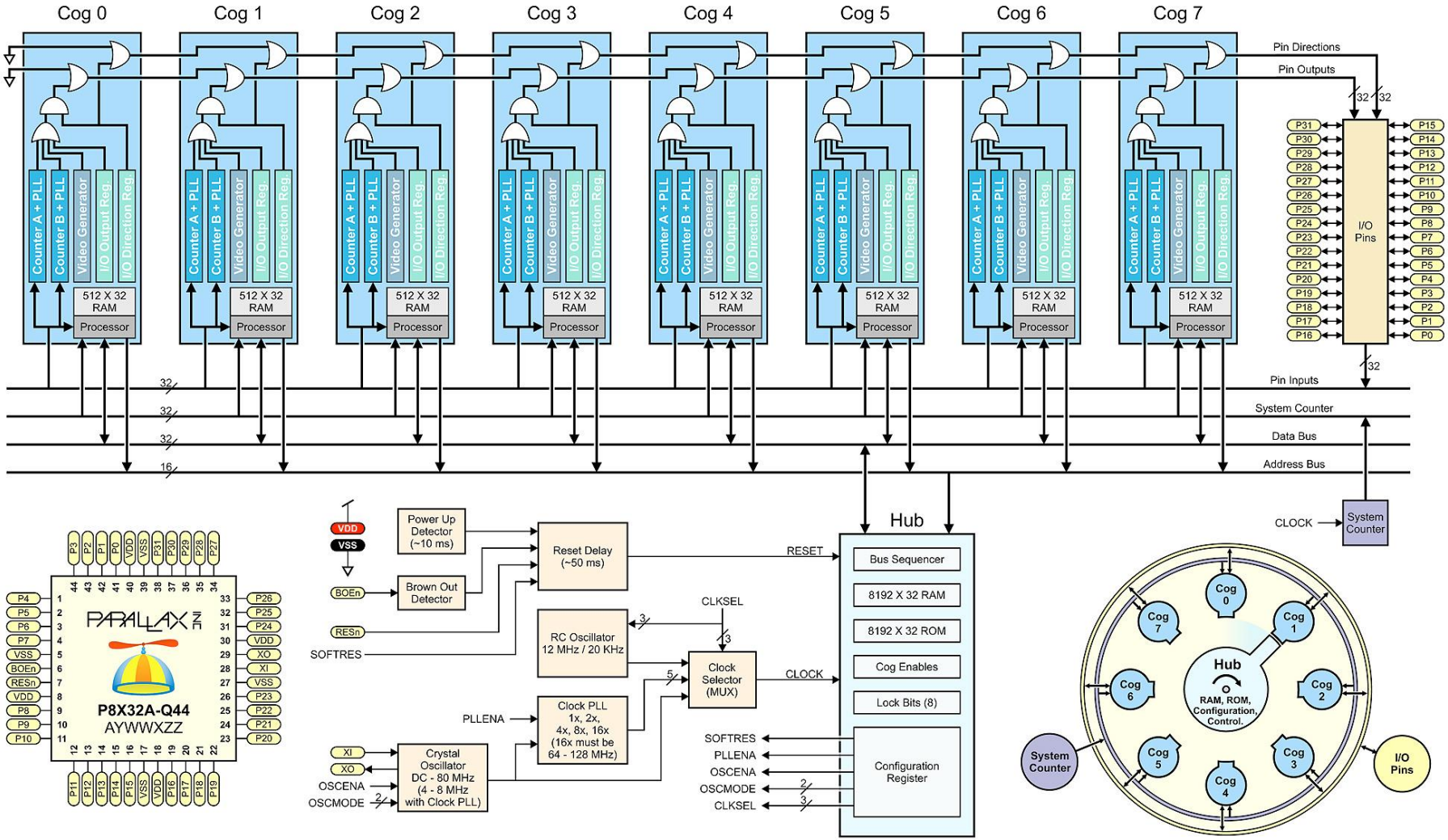


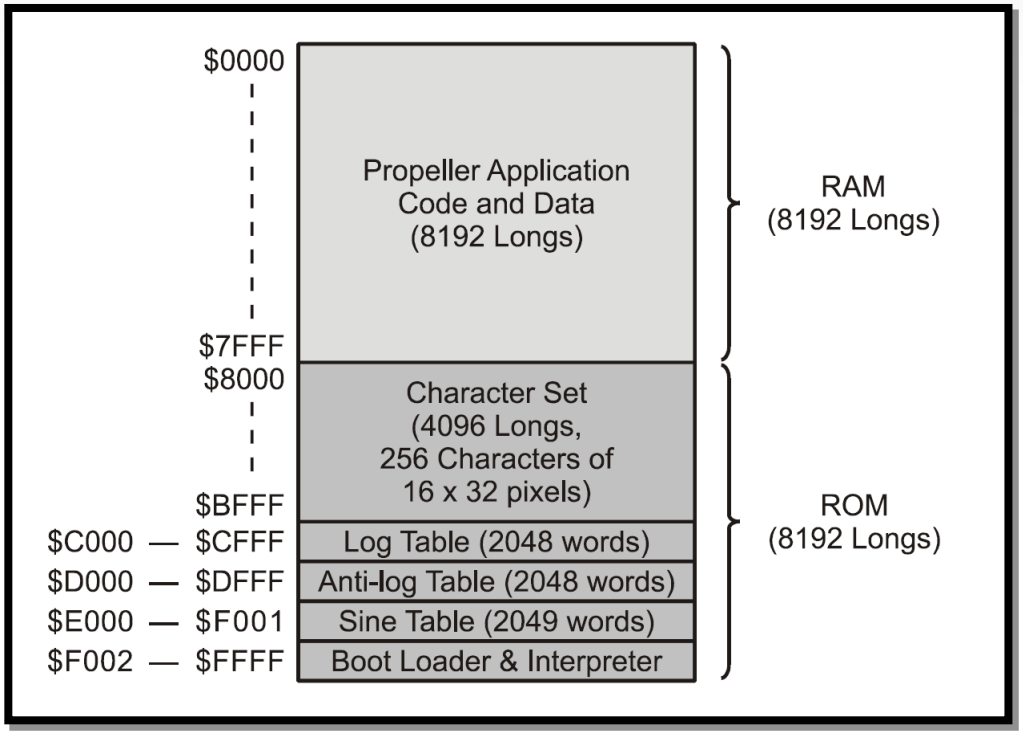
Schéma fonctionnel



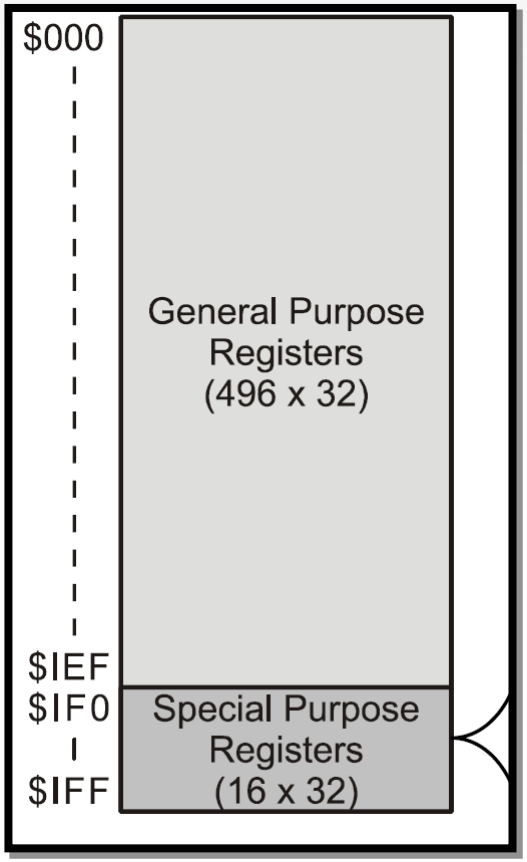
Hub and Cog Interaction

Mémoires

- Main RAM



- Cog RAM



Registres spéciaux

Address	Name	Type	Description
$1F0	PAR	Read-Only[1]	Boot Parameter
$1F1	CNT	Read-Only[1]	System Counter
$1F2	INA	Read-Only[1]	Input States for P31 - P0
$1F3	INB	Read-Only[1]	Input States for P63- P32[2]
$1F4	OUTA	Read/Write	Output States for P31 - P0
$1F5	OUTB	Read/Write	Output States for P63 – P32[2]
$1F6	DIRA	Read/Write	Direction States for P31 - P0
$1F7	DIRB	Read/Write	Direction States for P63 - P32[2]
$1F8	CTRA	Read/Write	Counter A Control
$1F9	CTRB	Read/Write	Counter B Control
$1FA	FRQA	Read/Write	Counter A Frequency
$1FB	FRQB	Read/Write	Counter B Frequency
$1FC	PHSA	Read/Write	Counter A Phase:
$1FD	PHSB	Read/Write	Counter B Phase
$1FE	VCFG	Read/Write	Video Configuration
$1FF	VSCL	Read/Write	Video Scale

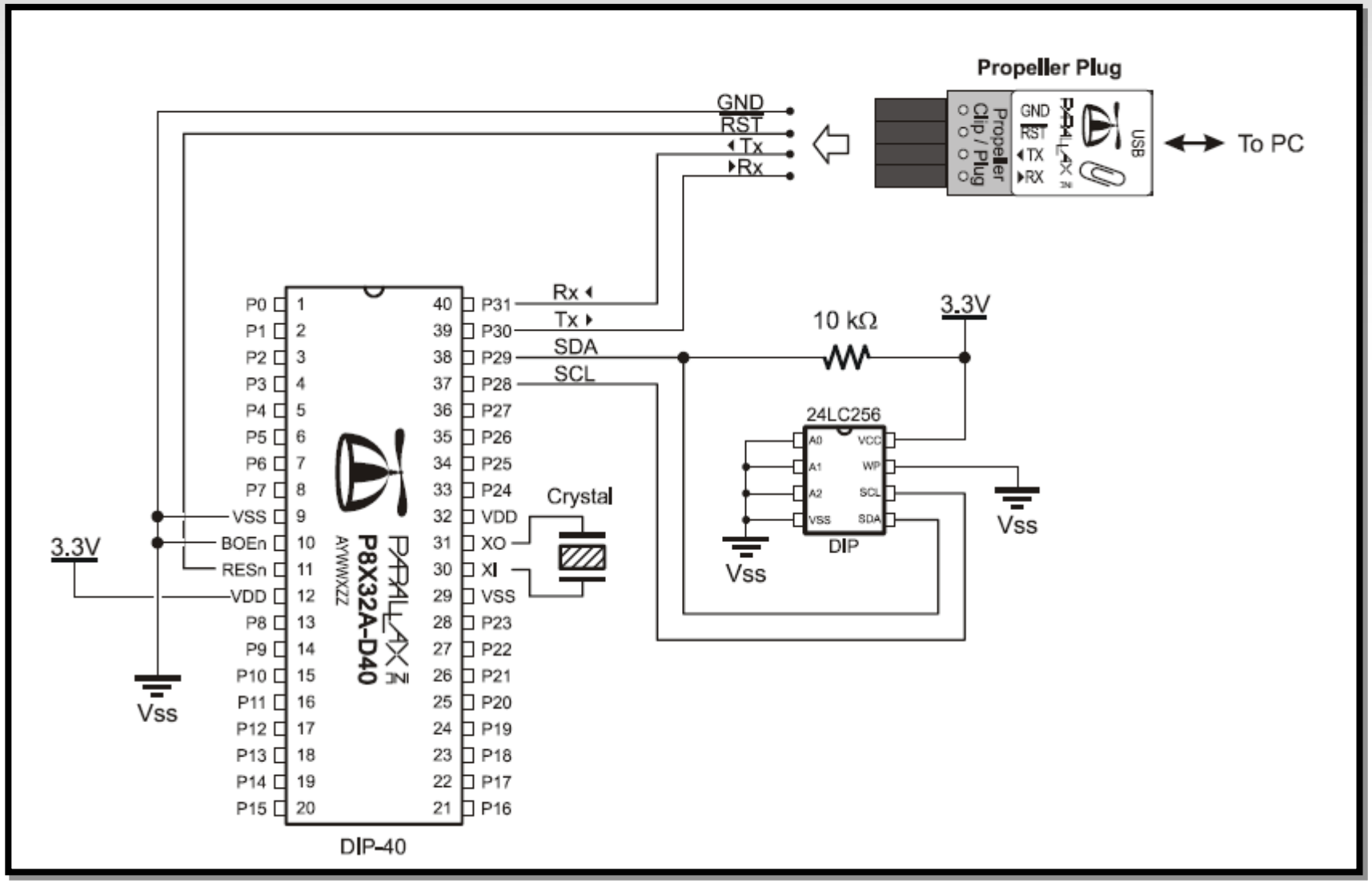
Périphériques

- 2 compteurs par Cog
- 1 générateur vidéo par Cog
- 1 compteur global “Free running”
- 1 oscillateur interne RC (12MHz – 20 kHz)
- 8 sémaphores binaires

Propeller Tool

- Langages Spin et ASM
- Permet de programmer le Propeller (RAM et EEPROM)
- Simple d'utilisation
- Contient un terminal série
- Contient les différents documents de références

Branchement minimal fonctionnel



Partie pratique

Introduction au langage Spin

Démo 1

- Identification du Propeller
- Première fonction (Bloc PUB)
- Entrées/Sorties (**`dira`**, **`outa`**, **`ina`**)
- Conditions : **`if`**, **`else`**
- Boucle infinie : **`repeat`**
- Programmation RAM vs EEPROM
- Visualisation de la mémoire

Démo 2

- Valeurs fixes et configurations (Bloc `CON`)
- Délais avec **`waitcnt`**
- Compteur global **`cnt`**
- Fréquence actuelle **`clkfreq`**
- Boucles avec indentation

Démo 3

- Commentaires
- Paramètres de fonctions
- Post-set/clear : **~~/~**
- Inversion binaire : **!**

Démo 4

- Variables globales (Bloc `VAR`)
- Utilisation de plusieurs Cogs : `cognew`
- Types de variables : `byte`, `word`, `long`
- Nécessité du “Stack”
- Opérateur d'adresse : `@`

Démo 5

- Création d'un objet
- Fonctions privées (bloc `PRI`)
- Arrêt d'un cog : **`cogstop`**
- Utilisation des objets (bloc `OBJ`)

Démo 6

- Librairies standards
- Utilisation du port série
- Opérateur `String`
- Constante d'un autre objet : **#**

Démo 7

- Utilisation de multiples pilotes
- Utilisation d'un ADC externe
- Recherche d'objets dans l'OBEX

Démo 8

- Utilisation des compteurs `frqa, phsa, ctra`
- Modes :

Table 2-7: Counter Modes (CTRMODE Field Values)

CTRMODE	Description	Accumulate FRQx to PHSx	APIN Output*	BPIN Output*
%00000	Counter disabled (off)	0 (never)	0 (none)	0 (none)
%00001	PLL internal (video mode)	1 (always)	0	0
%00010	PLL single-ended	1	PLLx	0
%00011	PLL differential	1	PLLx	!PLLx
%00100	NCO single-ended	1	PHSx[31]	0
%00101	NCO differential	1	PHSx[31]	!PHSx[31]
%00110	DUTY single-ended	1	PHSx-Carry	0
%00111	DUTY differential	1	PHSx-Carry	!PHSx-Carry
%01000	POS detector	A^1	0	0
%01001	POS detector with feedback	A^1	0	$!A^1$
%01010	POSEDGE detector	A^1 & $!A^2$	0	0
%01011	POSEDGE detector w/ feedback	A^1 & $!A^2$	0	$!A^1$
%01100	NEG detector	$!A^1$	0	0
%01101	NEG detector with feedback	$!A^1$	0	$!A^1$
%01110	NEGEDGE detector	$!A^1$ & A^2	0	0
%01111	NEGEDGE detector w/ feedback	$!A^1$ & A^2	0	$!A^1$
%10000	LOGIC never	0	0	0
%10001	LOGIC !A & !B	$!A^1$ & $!B^1$	0	0
%10010	LOGIC A & !B	A^1 & $!B^1$	0	0
%10011	LOGIC !B	$!B^1$	0	0
%10100	LOGIC !A & B	$!A^1$ & B^1	0	0
%10101	LOGIC !A	$!A^1$	0	0
%10110	LOGIC A <> B	A^1 <> B^1	0	0
%10111	LOGIC !A | !B	$!A^1$ | $!B^1$	0	0
%11000	LOGIC A & B	A^1 & B^1	0	0
%11001	LOGIC A == B	A^1 == B^1	0	0
%11010	LOGIC A	A^1	0	0
%11011	LOGIC A | !B	A^1 | $!B^1$	0	0
%11100	LOGIC B	B^1	0	0
%11101	LOGIC !A | B	$!A^1$ | B^1	0	0
%11110	LOGIC A | B	A^1 | B^1	0	0
%11111	LOGIC always	1	0	0

Démo 9

- Applications avancées
- Développement du jeu “Guess my number”
- Utilisation du péripérique vidéo
- Pilote clavier
- Valeurs aléatoires : ?