MAI ET JUIN 2025 | ELEKTORMAGAZINE.FR

€ 9,95
19 FS | 14 $CA

elektor
MEDIA & LEARNING

ÉDITION
BONUS
NUMÉRIQUE

513B
DEPUIS 1961

FOCUS SUR

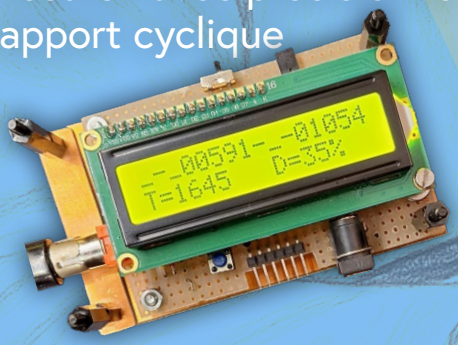
Test et mesure

oscilloscope/ multimètre/ générateur

Fnirsi 2C53T : avec deux voies et une bande passante de 50 MHz

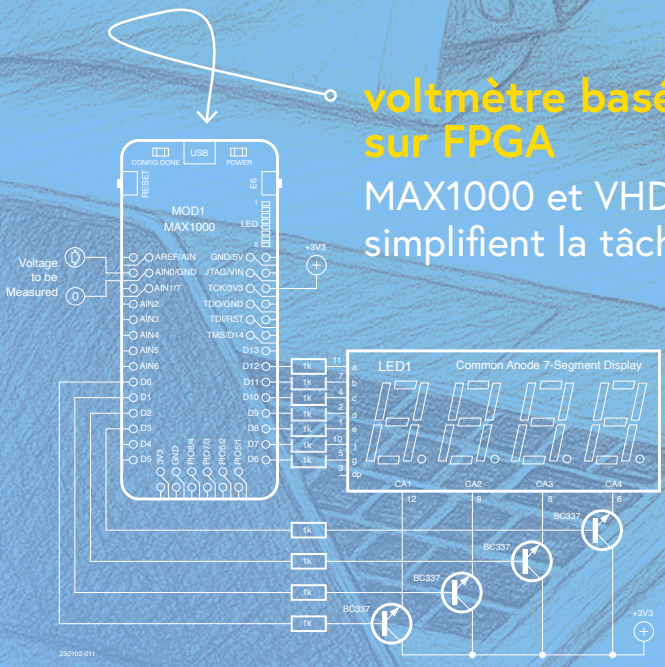
mesure de la modulation de largeur d'impulsion (PWM) avec un PIC

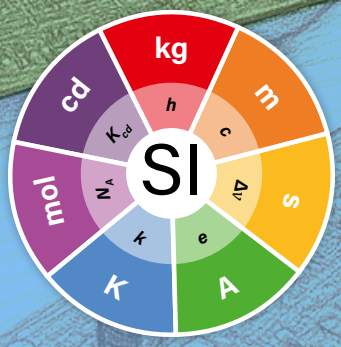
mesurez avec précision le rapport cyclique



voltmètre basé sur FPGA

MAX1000 et VHDPlus simplifient la tâche





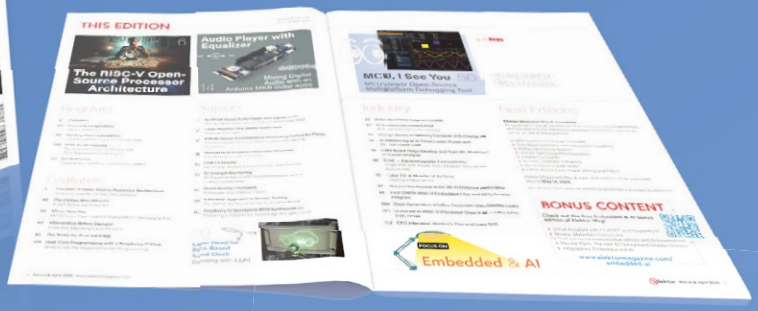
infographies :

test et mesure

Rejoignez la communauté Elektor

Devenez membre **maintenant !**

elektor GOLD membership



- accès à l'archive numérique depuis 1978 !
- 8x magazine imprimé Elektor
- 8x magazine numérique (PDF)
- 10 % de remise dans l'e-choppe et des offres exclusives pour les membres
- accès à plus de 5000 fichiers Gerber

Également disponible

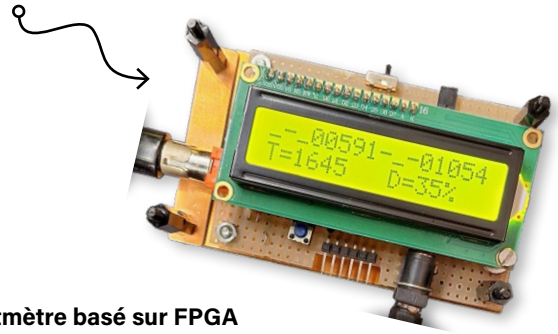
abonnement elektor GREEN membership sans papier !

- accès à l'archive numérique d'Elektor
- 10 % de remise dans l'e-choppe
- 8x magazine Elektor (PDF)
- accès à plus de 5000 fichiers Gerber

www.elektormagazine.fr/membres

elektor MEDIA & LEARNING

DANS CE NUMÉRO



ÉDITO

C. J. Abate

Directeur de contenu, Elektor

Test et mesure à portée de main

Les solutions de test et de mesure sont indispensables pour le débogage, la validation et l'optimisation des réalisations électroniques. Dans cette édition bonus, nous mettons en lumière des outils pratiques et accessibles que les électroniciens, les étudiants et les makers peuvent utiliser directement sur leur établi.

Ce numéro propose une approche ingénieuse de l'analyse des signaux dans l'article « mesure de la modulation de largeur d'impulsion (PWM) avec un PIC ». Lisez l'article pour découvrir comment utiliser un PIC16F628 pour mesurer la période des impulsions PWM, en capturant avec précision les fronts montants et descendants ainsi que les données sur une période complète.

Nous explorons également l'univers captivant de la programmation FPGA. Dans l'article « voltmètre basé sur FPGA », vous apprendrez à concevoir un voltmètre en utilisant l'environnement VHDPlus et de la carte FPGA MAX1000. Un projet idéal pour tous ceux qui souhaitent passer de la théorie à la conception numérique concrète.

Vous souhaitez optimiser votre équipement ? Découvrez le 2C53T : un instrument compact 3-en-1 réunissant oscilloscope, multimètre et générateur de fonctions. Nous vous proposons une revue complète de cet outil polyvalent.

Cette édition bonus est entièrement consacrée à l'optimisation de votre flux de travail en électronique. Que vous cherchiez à affiner des signaux, à déboguer des circuits ou à explorer de nouveaux outils de test, vous trouverez ici de quoi enrichir votre établi.

Bonne lecture, et n'hésitez pas à partager vos réalisations sur la plateforme Elektor Labs !

notre équipe

Rédacteur en chef : Jens Nickel | **Rédaction :** Hans Adams, Asma Adhimi, Roberto Armani, Eric Bogers, Jan Buiting, Rolf Gerstendorf (RG), Ton Giesberts, Saad Imtiaz, Alina Neacsu, Dr. Thomas Scherer, Jean-Francois Simon, Clemens Valens, Brian Tristam Williams | **Contributeurs réguliers :** David Ashton, Stuart Cording, Tam Hanna, Ilse Joostens, Prof. Dr. Martin Ossmann, Alfred Rosenkränzer | **Maquette :** Harmen Heida, Sylvia Sopamena, Patrick Wielders | **Des questions techniques :** redaction@elektor.fr

COLOPHON

48ème année n° 513B , ISSN 0181-7450
mai-juin 2025
N° de TVA Intracommunautaire : FR90319937454
Dépôt légal : mai 2025
CPPAP 1125 T 83713
Directeur de la publication : Donatus Akkermans

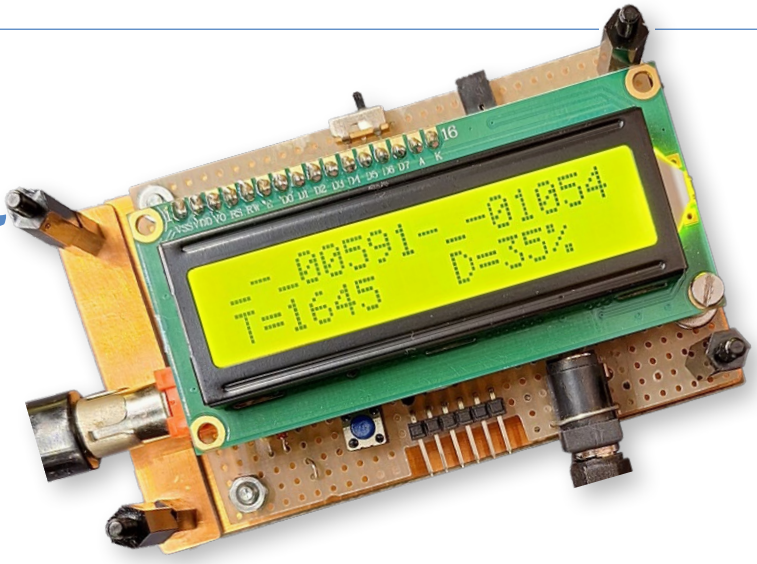
Elektor Magazine est publié 8 fois par an par
PUBLITRONIC SARL – c/o Regus Roissy CDG
1, rue de la Haye – BP 12910
FR - 95731 Roissy CDG Cedex
www.elektor.fr | www.elektormagazine.fr

Pour toutes vos questions : service@elektor.fr
Devenez membre : www.elektormagazine.fr/abo

Publicité : Ouafae Hassani
Tél. : +31 (0)6 41312932 | ouafae.hassani@elektor.com
www.elektormagazine.fr/publicité

Tarifs Annuels :
France 1 an 129,95 € (8 numéros)

mesure de la modulation de largeur d'impulsion (PWM) avec un PIC



Giovanni Carrera (Italie)

Bien que la programmation d'un Arduino soit plus simple que celle d'un PIC, ce dernier offre une précision et une exactitude supérieures dans le calcul de la synchronisation des signaux PWM grâce à son architecture. Les défis associés à son utilisation sont certes plus complexes, mais ils enrichissent considérablement l'expérience des passionnés de codage. L'appareil que nous proposons est capable de mesurer simultanément les durées d'impulsion haute et basse, ainsi que la période totale, avec une résolution d'une microseconde pour des durées d'impulsion allant de 10 µs à 65 535 µs pour des signaux PWM avec des fréquences allant de 7,63 Hz à 50 kHz.

La technique de modulation de largeur d'impulsion, communément désignée par l'acronyme PWM (*Pulse Width Modulation*), consiste à ajuster la durée des impulsions au cours du temps. Initialement développée pour les télécommunications, cette méthode est également employée dans les conversions numérique-analogiques pour moduler la tension ou le courant moyen. Cette valeur est déterminée par le rapport cyclique, qui est le rapport entre la durée de l'impulsion positive et la période totale.

La modulation PWM est couramment utilisée pour réguler la tension dans les alimentations à découpage, pour la commande de moteurs et les servomoteurs et même pour gérer la sortie de certains capteurs. De ce fait, la plupart des microcontrôleurs modernes disposent d'une ou de plusieurs sorties PWM. Certains transducteurs, tels que les capteurs de distance acoustiques, ont une sortie d'impulsion numérique dont la durée est proportionnelle à l'amplitude du signal. Le but de ce projet n'est pas de générer des signaux PWM, mais de mesurer leur durée d'impulsion avec précision.

Dans cet article, je présenterai une version du projet qui utilise le PIC16F**628** pour mesurer les durées des phases haute et basse des impulsions, ainsi que la période, comme illustré dans la photo d'introduction. Pour ceux désirant également calculer le rapport cyclique en pourcentage, il sera nécessaire d'utiliser un PIC16F648, car ce calcul requiert une opération de division. Le programme ne nécessite que quelques instructions supplémentaires, mais le code dépasse 2 K mots, et le F628 n'est doté que de 2 K de mémoire flash. Les codes sources et les fichiers HEX compilés sont disponibles pour les deux versions [1]. Les autres composants du projet restent identiques.

Le microcontrôleur PIC

Pourquoi ce choix ? Dans mon cas, j'avais déjà réalisé un projet similaire il y a quelques années, à une époque où je privilégiais l'utilisation des PIC pour mes réalisations, avant de passer au développement avec l'EDI Arduino et MicroPython.

Pour les microcontrôleurs PIC, la fréquence de l'horloge interne correspond à un quart de la fréquence de l'oscillateur à quartz. Ainsi, toutes les instructions sont exécutées en un seul cycle d'horloge, à l'exception des branches de programme. Avec un quartz de 16 MHz, l'horloge interne fonctionne à 4 MHz, le temps d'exécution des instructions n'est donc que de 0,25 µs.

Le PIC16F628 est un microcontrôleur de Microchip à architecture RISC (*Reduced Instruction Set Computer*) traitant des données sur 8 bits, mais ses 35 instructions sont codées chacune sur un mot de 14 bits. Pour ce modèle, la mémoire flash est de 2K mots. Parmi ses nombreuses fonctions d'entrée/sortie, ce PIC intègre une fonction de capture d'événements sur la broche CCP1 (RB3), où il enregistre le contenu du Timer1 dans deux registres, CCPR1L et CCPR1H. Une fonction similaire existe également pour l'ATmega328P de l'Arduino UNO, mais le choix d'un PIC permet de réaliser un système beaucoup plus simple et compact, avec une consommation d'énergie nettement réduite.

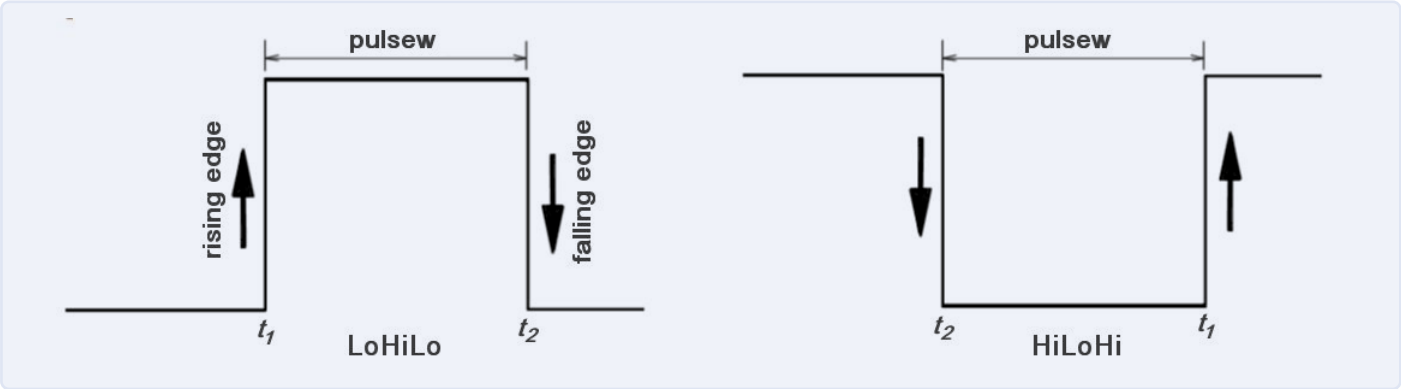


Figure 1. Deux modes de capture différents sont disponibles, pour lesimpulsions « LoHiLo » (à gauche) et « HiLoHi » (à droite).

Bien entendu, ceux qui sont habitués à travailler avec Arduino auront plus de difficultés à concevoir des applications avec des PIC. Il n'est pas aussi facile de trouver des cartes préfabriquées ou de bénéficier de l'abondance de bibliothèques disponibles pour Arduino, qui facilitent la gestion des composants utilisés. À mon avis, se limiter à Arduino n'offre pas la possibilité d'explorer en profondeur les principes sous-jacents, ni même de comprendre précisément ce que l'on fait.

De nombreux utilisateurs d'Arduino se contentent de reproduire les connexions de projets, souvent illustrées à l'aide de Fritzing et des images montrant les composants disposés sur des plaques à essai, sans comprendre le fonctionnement sous-jacent de ces montages. Parfois, ces projets ne fonctionnent pas du tout, en raison d'erreurs de connexion.

Le compilateur utilisé pour les PIC est moins convivial que l'EDI Arduino et les bibliothèques disponibles sont beaucoup moins nombreuses. Par exemple, il manque une fonction équivalente à `pulsein()` pour mesurer la durée d'une impulsion. Pour ces raisons, le concepteur doit acquérir une compréhension plus approfondie du fonctionnement des microcontrôleurs. C'est ce que j'ai dû faire pour ce projet, où l'étude approfondie de la fiche technique [2] était nécessaire pour maîtriser le fonctionnement des timers et de la capture CCP.

Le compilateur mikroPascal, que j'ai utilisé pour développer des programmes pour les PIC, est gratuit tant que la taille du code ne dépasse pas 2K mots (1 mot = 14 bits), ce qui correspond à la capacité de la mémoire flash du PIC utilisé, alors que la RAM disponible est seulement de 224 octets. Avec de telles contraintes, les possibilités sont limitées : par exemple, l'utilisation de divisions en virgule flottante peut rapidement saturer la mémoire disponible. Cependant, la compilation avec mikroPascal est nettement plus rapide que celle réalisée avec Arduino, et le code généré est souvent plus compact et optimisé. J'ai développé le programme en Pascal en utilisant un compilateur croisé performant pour les PC sous Windows : le mikroPascal PRO pour PIC, en version 7.6.0. Une version complète peut être téléchargée depuis le site de MIKROE [3].

Dans cette première version du programme, le code est relativement court, ce qui facilite sa compilation. Ceux qui sont plus familiers avec le langage C ou Basic peuvent télécharger les compilateurs correspondants et adapter le programme à ces langages. Si aucune modification n'est requise, il n'est même pas nécessaire de télécharger un compilateur ; il suffit de disposer d'un programmateur PIC utilisant le fichier hex déjà compilé, *PWMmeter.hex*.

J'ai également développé une autre version du programme qui calcule le rapport cyclique. Cependant, en raison de l'utilisation de la division en virgule flottante, cette version requiert davantage de mémoire flash et un PIC16F648. Dans ce cas, si vous n'avez pas acquis la licence du compilateur, vous devrez programmer la puce en utilisant le fichier HEX fourni avec ce projet.

Ces deux microcontrôleurs disposent du même brochage.

Mesure de la durée de l'impulsion

Certains microcontrôleurs dotés d'un interpréteur BASIC intégré, tels que Stamp, Picaxe ou BasicX, utilisent une routine spécifique (`PULSIN`) pour mesurer la durée des impulsions, mais ne peuvent pas mesurer les courtes impulsions à cause de leur lenteur relative. Les cartes Arduino utilisent la fonction `pulsein(pin, level, timeout)` pour mesurer la durée d'une impulsion, niveau haut ou bas, avec une résolution de l'ordre de la microseconde. Pour des durées plus longues, il existe la fonction `pulseInLong()`, qui utilise l'interruption et mesure des durées de 10 µs à 3 minutes. Les deux fonctions renvoient un `unsigned long`.

Le programme utilise deux interruptions : Timer0 est utilisé pour le temps d'échantillonnage tandis que la seconde source d'interruption est déclenchée par les événements de capture CCP (capture/compare/PWM) sur les fronts montants et descendants de l'impulsion. Les durées correspondantes sont enregistrées dans le Timer1.

Timing avec Timer0

Le Timer0 génère une interruption qui sert à échantillonner les mesures et à les afficher sur un écran LCD tous les 50 événements, ce qui équivaut à environ 0,5 seconde. Le Timer0 est un compteur TMR0 de 8 bits, précédé d'un diviseur programmable de 8 bits appelé prescaler. Le programme configure l'horloge interne à une fréquence qui est le quart de celle de l'oscillateur à quartz : 16 MHz /4 = 4 MHz. En ajustant le prescaler à sa valeur maximale (1:256), Timer0 reçoit une fréquence d'entrée de f1 = 4 MHz / 256 = 15. 625 kHz, avec une période de 64 µs ; charger 100 dans le compteur résulte en une interruption toutes les 156 (256-100) impulsions de fréquence f1, correspondant à une période de 156 × 64 = 9984 µs (environ 100 Hz).

L'interruption du Timer0 est générée lorsque le timer/compteur du registre TMR0 passe de 0xFF à 0x00, ce qui définit le bit T0IF. Cette interruption génère l'horloge utilisée pour échantillonner les mesures et les afficher sur l'écran LCD, tous les 50 événements, ce qui correspond à environ 0,5 s.

Capture d'événements avec Timer1

Lorsque le front montant est détecté sur la broche RB3, le programme bascule le mode de capture pour détecter ensuite le front descendant, et inversement. Les séquences d'impulsions peuvent alterner entre bas-haut-bas, que nous nommons LoHiLo, ou l'inverse, haut-bas-haut, dénommé HiLoHi, comme illustré dans la **figure 1**.

Pour les impulsions LoHiLo, le programme définit le registre `CCP1CON := $05` avec capture sur front montant. Lorsque cet événement se produit, la routine d'interruption enregistre le contenu des registres `CCPR1L` et `CCPR1H` qui sera ensuite transféré à t1. Ensuite, le `CCP1CON := $04`, réglant la capture sur front descendant, à l'événement duquel les registres `CCPR1L` et `CCPR1H` seront lus à nouveau, ce qui sera ensuite transféré à t2, comme le montre la figure 1. La durée de `pulsew` est calculée à partir de la différence t2-t1.

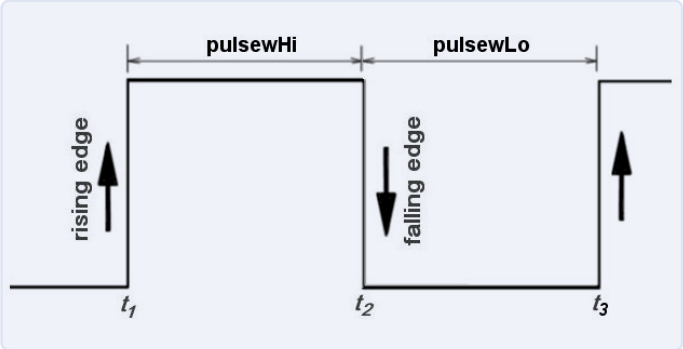


Figure 2. Acquisition des durées t_1, t_2, et t_3 pour effectuer les calculs nécessaires.

Pour les impulsions HiLoHi, le principe est le même, mais on commence par le front descendant (t2) et on termine par le front montant (t1), dans ce cas a durée de `pulsew` est calculée par la différence t1-t2. Timer1 intègre un compteur de 16 bits, précédé d'un prescaler de 2 bits. L'horloge interne de 4 MHz est également utilisée ici. Avec le prescaler = 1:1 on atteint une résolution de 0,25 µs, ce qui peut être considéré comme excessif pour un quartz ordinaire. Nous avons donc opté pour un rapport de 1:4 avec une résolution de 1 µs. Le système mesure donc des impulsions d'une durée comprise entre environ 10 µs et 65 535 µs. En dessous de 10 µs, cela ne fonctionne pas, car il y a plusieurs instructions dans la routine d'interruption, qui requièrent un temps d'exécution non négligeable. Pour mesurer des durées plus longues, avec une résolution de 2 µs, vous devriez utiliser un prescaler de 1:8. Dans ce cas, il faudra multiplier les durées par deux après les avoir converties en nombres entiers longs.
La version améliorée du programme initial permet désormais de mesurer simultanément deux durées, facilitant ainsi le calcul de la période PWM ainsi que du rapport cyclique. Dans ce cas, le programme sélectionne la capture pour le front montant, stocke le temps dans t1, met le flag `firstcre` à l'état haut pour distinguer ce front du dernier. Il configure ensuite la capture pour le front descendant, enregistre le temps dans t2, puis définit la capture pour le front montant suivant, qui sera enregistré dans t3, comme illustré dans la figure 2. Lors du troisième front, la routine d'interruption définit le drapeau `datok` pour signaler que les temps t1, t2, et t3 ont été correctement mesurés.
Il calcule donc :

pulsewHi = t2−t1, pulsewLo = t3−t2, period = pulsewHi + pulsewLo

Le calcul du rapport cyclique nécessite des opérations avec des variables flottantes (réelles en Pascal) qui dépassent les limites de la mémoire flash du 16F628, mais nous pouvons les effectuer avec une calculatrice :

Period = T = pulsewHi + pulsewLo
Duty cycle: D = pulsewHi / T × 100 [%]

Comparaison avec la fonction pulseIn() d'Arduino

Pour effectuer une comparaison, j'ai également développé un petit programme pour l'Arduino Uno, spécifiquement pour une impulsion LoHiLo :

```
// program pulseintest
#define pulsein 4

void setup() {
  Serial.begin(115200);
  pinMode(pulsein, INPUT);
}

void loop() {
 unsigned long duration = pulseIn(pulsein, HIGH);
  Serial.print("Pulse High [us] = ");
  Serial.println(duration);
  delay(500);
}
```

Pour vérifier les durées, j'ai utilisé mon propre circuit basé sur un quartz avec un diviseur programmable et une sortie d'onde carrée comme référence ; les résultats obtenus sont comparés à ceux de mon système basé sur un PIC dans le **tableau 1**.

Tableau 1. Mesures.

ref. [µs]	PIC [µs]	Arduino [µs]	Err.% PIC	Err.% Arduino
5		5…6		0…20
50	50	50…51	0	0…2
500	500	494…500	0	−0.8…0
5,000	5,000	4,966	0	−0.68
50,000	49,997	49,662	−0.006	−0.676

Comme on peut le constater, le système PIC est très précis.

Schéma du circuit

La **figure 3** illustre le schéma du circuit. L'utilisation d'un régulateur un LM7805 (un 78L05 fera aussi l'affaire), permet d'alimenter le système comme un Arduino UNO, c'est-à-dire avec une tension de 7 V (valeur optimale) à 12 V. Toutefois, il consomme moins de 10 mA sans rétroéclairage (W2 off), avec environ 3 mA consommés par le régulateur. Une alternative plus simple consiste à utiliser trois piles ordinaires de 1,5 V. Dans ce cas, l'élément le plus critique est l'écran LCD, qui nécessite une tension nominale de 5 V, bien que des appareils similaires fonctionnent aussi généralement sous 4,5 V.

Liste des composants

Résistances
R1 = 10 kΩ, ±5%, ¼ W
R2 = 100 Ω, ±5%, ¼ W
R3 = 100 Ω, ±5%, ½ W
Rp1 = 22 kΩ, trimmer

Condensateurs (céramique)
C1, C2 = 22 pF, 50 V
C3, C4, C5, C6 = 100 nF, 25 V

Semi-conducteurs
U1 = Microcontrôleur PIC16F628A ou PIC16F648A
U2 = Régulateur LM7805 5 V
D1, D2 = diode 1N4148

Divers
Afficheur = LCD 2×16
X1 = quartz 16 MHz
SW1 = bouton poussoir (RESET)

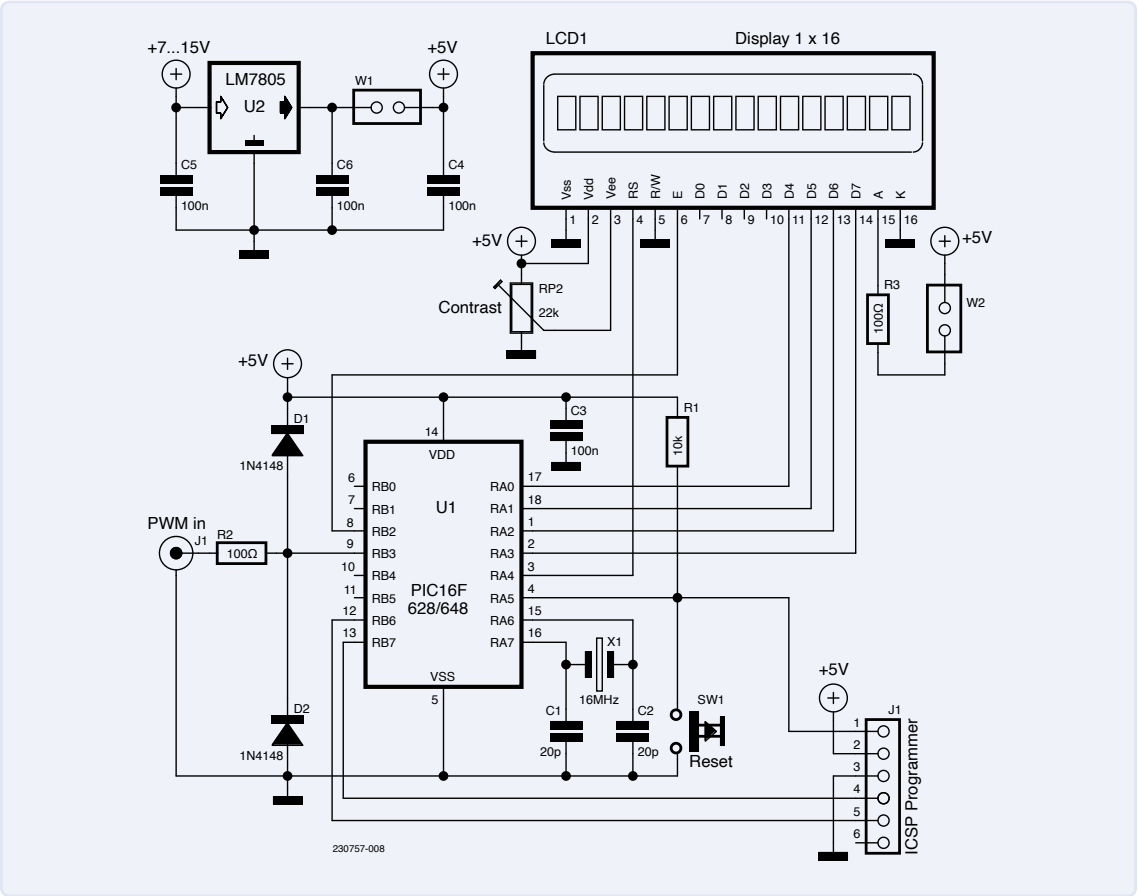


Figure 3.
Schéma du projet.

Pour l'affichage, j'ai opté pour un écran LCD standard de deux lignes de 16 caractères, doté d'une interface parallèle compatible HD44780. Le cavalier W1 sert à prévenir les conflits d'alimentation et doit être retiré lors de la programmation de la puce avec un PICKIT. Le cavalier (ou interrupteur) W2 est utilisé pour désactiver le rétroéclairage afin d'économiser de l'énergie.

Prototype

La **figure 4** montre la disposition des composants sur la carte perforée utilisée pour fabriquer le prototype. L'afficheur LCD a été retiré afin de révéler les composants montés en dessous.
En haut, un petit interrupteur (W2) permet d'éteindre le rétroéclairage de l'afficheur. Sur la gauche, un connecteur RCA est utilisé pour l'entrée PWM. En bas, un connecteur à 6 broches permet la programmation de la puce, compatible avec le programmateur en circuit PICkit. Si vous disposez d'un programmateur PIC équipé de sockets ZIF, ce connecteur n'est pas nécessaire, mais il vous faudra retirer le PIC de son support pour le programmer. À proximité, toujours en bas, se trouve le connecteur d'alimentation. Il est possible de réduire encore la taille de la carte, car celle-ci a été réutilisée d'un projet antérieur. Le connecteur femelle à 5 broches situé sur la droite n'est pas utilisé.

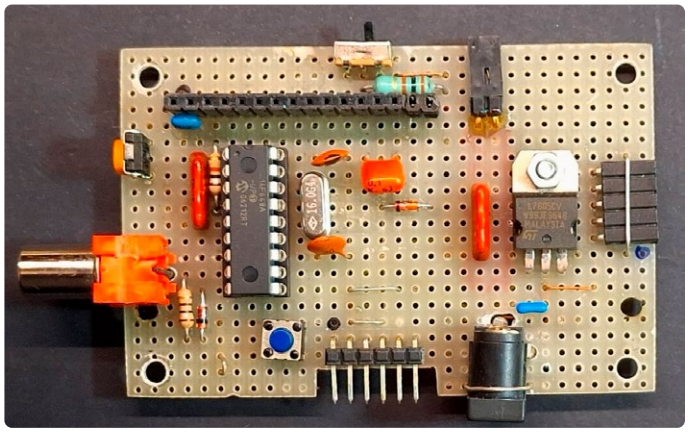


Figure 4. Le prototype terminé, réalisé sur une plaque d'essai récupérée d'un projet précédent. Le connecteur femelle à 5 broches (à droite) n'est pas utilisé.

Programme

Nous avons décrit une grande partie du programme précédemment. Les sorties sur le LCD sont plus basiques que les fonctions de la bibliothèque *LiquidCrystal* d'Arduino, et les messages doivent être des chaînes ou des tableaux de caractères, il est donc nécessaire de convertir les variables en chaînes et de supprimer tout espace initial. Le travail le plus important est effectué par la routine d'interruption. Dans la première partie, elle vérifie si l'interruption est causée par un débordement du Timer0, auquel cas `INTCON`, `T0IF` = 1, met le flag `int0flg`, qui est utilisé pour le timing à l'état haut. Elle recharge ensuite le nombre 100 dans le compteur de façon à déclencher une interruption après 156 impulsions et remet les drapeaux d'interruption `INTCON` et `T0IF` à 0.
La deuxième partie, plus complexe, est dédiée à la gestion des captures des fronts montants et descendants et à sauvegarder le contenu des deux registres de capture, `CCPR1L` et `CCPR1H`, dans les variables `t1`, `t2` et `t3`, correspondant au premier front montant, au front descendant et au deuxième front montant. Lorsque ces trois valeurs ont été capturées, le flag `datok` est activé, ce qui sera utilisé par le programme pour calculer les durées et les périodes.
Les durées des fronts montants et descendant du signal PWM, indiquée par `'_-_'` pour le front montant, indiquée par `'-_-'` pour le front descendant sont affichés sur la première ligne. La période PWM, c.-à-d. la somme des deux durées, est afichée sur la deuxième ligne.

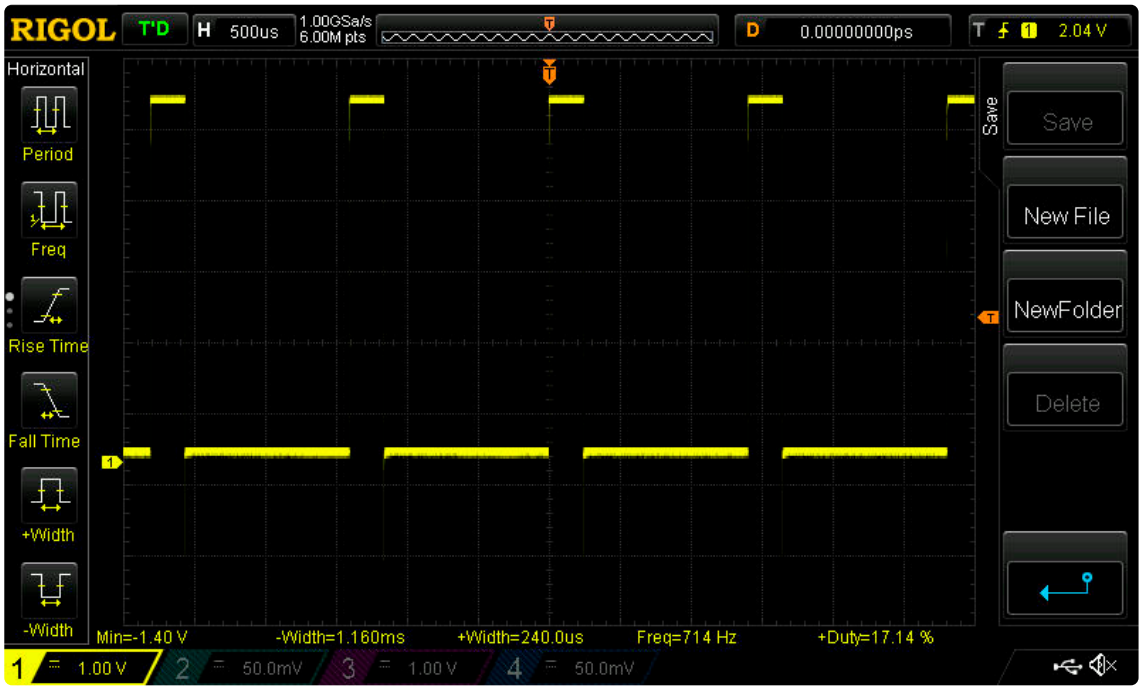


Figure 5. Capture d'écran de l'oscilloscope de la mesure présentée dans la figure 6.

En l'absence de signal, l'écran n'affiche rien. Le code complet est présenté dans le **listage 1**. Ce programme occupe 1 236 mots de mémoire flash (65 %) et 74 octets de mémoire vive (37 %), et la compilation a été réalisée en 187 ms.

Version du programme avec affichage du rapport cyclique

Pour cette version, il est nécessaire d'utiliser un PIC16F648A, semblable au F628 mais disposant de 4K mots de mémoire flash. Comme précédemment mentionné, pour compiler cette version, l'acquisition d'une licence du programme est requise afin de contourner la limite des 2K mots. Toutefois, un fichier hexadécimal déjà compilé pour le PIC16F648A est inclus dans le package de cet article, vous permettant de le charger directement sur le PIC.

Le programme affiche `pulsewHi` [µs], `pulsewLo` [µs], `period` [µs] et `dutyc` [%] sur l'écran.

Malheureusement, l'utilisation de la division en virgule flottante a entraîné des problèmes de mémoire lors de la compilation, le PIC16F648A s'avérant inadapté pour de telles opérations. J'ai résolu ce problème en convertissant le pourcentage du rapport cyclique en un nombre entier, ce qui a entraîné la perte des décimales.

Tests de comparaison

Les figures 5 et 6 présentent une comparaison entre les mesures réalisées avec un oscilloscope et celles obtenues par le système PIC16F648. Pour ces tests, nous avons utilisé un générateur d'impulsions TTL. Les mesures sont très similaires.

Listage du programme PIC16F648 PWMmeter

Le programme reste largement similaire à la version précédente, à l'exception de la routine `LCDprint`.

Nous avons ajouté des instructions pour le calcul et l'affichage du rapport cyclique. La routine modifiée est présentée dans le **listage 2**. Le code modifié occupe 2 266 mots de mémoire flash (55%) et 92 octets de mémoire vive (62%), et la compilation a été réalisée en 47 ms.

230757-04

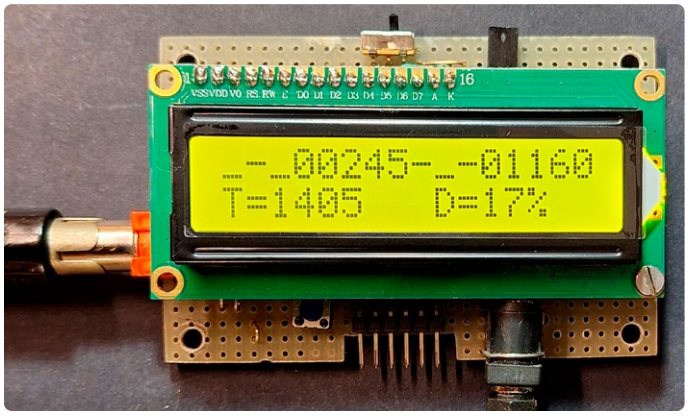


Figure 6. Lecture de la durée de l'impulsion «High» (245 µs), de l'impulsion «Low» (1 160 µs), de la période totale T (1 405 µs), et du rapport cyclique (17%).

Listage 2. PIC16F648 PWMmeter (section).

```
procedure LCDprint; // print measures on LCD
begin
  Lcd_Cmd(_LCD_CLEAR); // lcd clear
  WordToStrWithZeros(pulsewHi, texdH);// microseconds
  ltrim(texdH); // trims the leading spaces
  WordToStrWithZeros(pulsewLo, texdL);
  ltrim(texdL); // trims the leading spaces
  Lcd_Out(1, 1,'_-_' + texdH + '-_-' + texdL);
  LongWordToStr(period, texPer);
  ltrim(texPer); // trims the leading spaces
  //FloatToStr_FixLen(dutyc, texd, 5);
  Lcd_Out(2, 1,'T=' + texPer);
  dutyc:= real(pulsewHi)/real(period)*100.0;
  IntToStr(integer(dutyc), texd);
  ltrim(texd);
  Lcd_Out(2, 10,'D=' + texd + '%');
  LCDout:= false;
end;
```

À propos de l'auteur

Giovanni Carrera détient un diplôme en ingénierie électronique. Professeur à l'université dans la faculté d'ingénierie navale de Gênes, en Italie, il a enseigné divers cours tels que l'automatisation navale et la simulation des systèmes de propulsion des navires. M. Carrera a commencé sa carrière à la fin des années 1970 avec le microprocesseur 6502, avant de se tourner vers d'autres processeurs. Aujourd'hui, il se passionne pour la conception et le développement de circuits électroniques, tant analogiques que numériques, et partage ses réalisations sur ses blogs (ArduPicLab et GnssRtkLab) ainsi que dans divers magazines spécialisés.

Questions ou commentaires ?

Contactez Elektor (redaction@elektor.fr).

Produits

- **Tam Hanna, *Microcontroller Basics with PIC*, Elektor, 2020**
  www.elektor.fr/19188
- **Microchip MPLAB PICkit 5 In-Circuit Debugger/ Programmer**
  www.elektor.fr/20665

LIENS

[1] Téléchargements : https://elektormagazine.fr/230757-04
[2] PIC16F627A/628A/648A - Fiche technique : https://ww1.microchip.com/downloads/en/DeviceDoc/40044G.pdf
[3] Site web de MIKROE : https://mikroe.com/mikropascal-pic

Listage 1. PIC16F628 PWMmeter.

```
program PWMmeter;
// PIC16F628 measures PWM (resolution= 1 usec)
// display the value on LCD every 0.5 sec
// timer#0 for interrupt every 9.984 msec (approx 100Hz)
// the LCD display has  2 rows x 16
// by G. Carrera 23/12/2023
// I/O configuration:
// PortB.3 = pulse input   (in)
// PortA.0..3,= LCD data (4 bit mode) (out)
// PortB.2 = LCD E (out)
// PortA.4 = LCD RS (out)
// xtal = 16.00 MHz
var
  t1,t2,t3: word;
  int0flg,datok,LCDout,firstcre: boolean;
  i,t1l,t1h,t2l,t2h,t3l,t3h: byte;
  pulsewLo,pulsewHi: word;
  texdL: string[5];
  texdH: string[5];
  period: longint;
  texPer: string[10];
  dutyc: real;
  texd: string[6];
```

(Suite sur la page suivante)

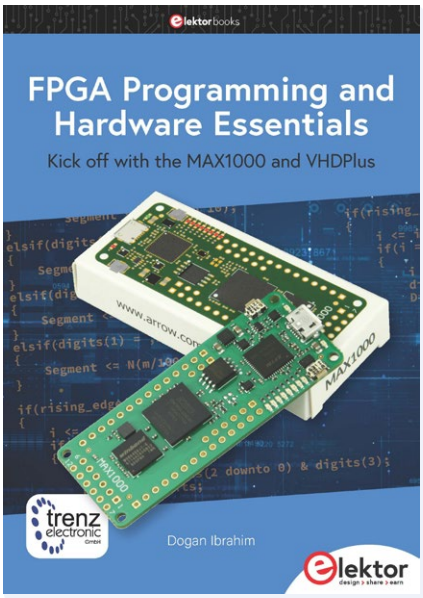
```
// Lcd module connections
var LCD_RS : sbit at RA4_bit;
var LCD_EN : sbit at RB2_bit;
var LCD_D4 : sbit at RA0_bit;
var LCD_D5 : sbit at RA1_bit;
var LCD_D6 : sbit at RA2_bit;
var LCD_D7 : sbit at RA3_bit;
var LCD_RS_Direction : sbit at TRISA4_bit;
var LCD_EN_Direction : sbit at TRISB2_bit;
var LCD_D4_Direction : sbit at TRISA0_bit;
var LCD_D5_Direction : sbit at TRISA1_bit;
var LCD_D6_Direction : sbit at TRISA2_bit;
var LCD_D7_Direction : sbit at TRISA3_bit;
// End Lcd module connections
procedure interrupt;
// read capture times between rising and falling edge or vice versa
begin
  if TestBit(INTCON, T0IF) then  // timer#0 interrupt
    begin
      ClearBit(INTCON, T0IF); // reset Timer#0 interrupt flag
      TMR0:= 100;
      SetBit(INTCON, T0IE); // Enables Timer#0 interrupt
      int0flg:= true;
    end;
  if TestBit(PIR1, CCP1IF) then // capture interrupt
    begin
      if  TestBit(CCP1CON, CCP1M0)  then // rising edge ($05)
        begin
          if firstcre then  // already captured the first rising edge t1
            begin
              t3l := CCPR1L;  //  load final time to t3
              t3h := CCPR1H;
              firstcre:= false;
              datok:= true;
            end
          else
            begin
              t1l := CCPR1L;  //  load initial time to t1
              t1h := CCPR1H;
              firstcre:= true; // the first rising edge is captured now
            end;
          CCP1CON := $04; // change to falling edge on CCP1
          ClearBit(PIR1, CCP1IF); //Clear CCP1 Int. flag (also due to changing)
        end
      else  // falling edge
        begin
          t2l := CCPR1L;  //  load intermediate time to t2
          t2h := CCPR1H;
          CCP1CON := $05; // change to rising edge on CCP1
          ClearBit(PIR1, CCP1IF); //Clear CCP1 Int. flag (also due to changing)
        end;
    end;
end;
procedure IOinit; // initialize I/O
begin
  TRISB:= %00111011;  // PORTB bit2 : output
  TRISA:= %10100000;  // PORTA bits 0..4 are outputs
  CMCON:= $07; // comparators off (RA0..RA4 usable as digital IO)
  Lcd_Init(); // Initialize LCD
  Lcd_Cmd(_LCD_CURSOR_OFF); // Turn off cursor
```

(Suite sur la page suivante)

```
  CCP1CON := $05;  //  rising edge on CCP1
  T1CON := $21;   // 1:4 prescaler, int. clk, enable t1
  OPTION_REG:= $07; // set prescaler of Timer#0 to 256
  ClearBit(OPTION_REG,7); //  enable port B pull-ups
  SetBit(PIE1,CCP1IE); // Enables the CCP1 interrupt
  SetBit(INTCON, PEIE); // Enables peripheral interrupt
  TMR0:= 100; // T0 overflow every 156 counts
  SetBit(INTCON, T0IE); // Enables Timer#0 interrupt
  ClearBit(INTCON, T0IF); // reset Timer#0 interrupt flag
  SetBit(INTCON, GIE); // Enables global interrupt
  int0flg:= false;
  Lcd_Out(1, 1, 'PWMmeter-231223'); // Print text to LCD (row,column,text)
  Delay_ms(2000);
  Lcd_Cmd(_LCD_CLEAR); // lcd clear
  i:=0;
  firstcre:= false;
end;
procedure LCDprint; // print measures on LCD
begin
  Lcd_Cmd(_LCD_CLEAR); // lcd clear
  WordToStrWithZeros(pulsewHi, texdH);// microseconds
  ltrim(texdH); // trims the leading spaces
  WordToStrWithZeros(pulsewLo, texdL);
  ltrim(texdL); // trims the leading spaces
  Lcd_Out(1, 1,'_-_' + texdH + '-_-' + texdL);
  LongWordToStr(period, texPer);
  ltrim(texPer); // trims the leading spaces
  //FloatToStr_FixLen(dutyc, texd, 5);
  Lcd_Out(2, 1,'T=' + texPer);
  LCDout:= false;
end;
begin  // main program
  IOinit; // Initialize
  LCDout:= false;
  while true Do //  endless loop
    begin
      if int0flg then
        begin
          int0flg:= false;
          i:=i+1;
          if i= 50 then // about 0.5 seconds
            begin
              LCDout:= true;
              i:= 0;
            end;
        end;
      if datok then
        begin
          datok:= false;
          t1:= (t1h shl 8)+t1l;
          t2:= (t2h shl 8)+t2l;
          t3:= (t3h shl 8)+t3l;
          pulsewHi:= t2-t1;
          pulsewLo:= t3-t2;
          period:= longint(pulsewHi) + longint(pulsewLo);
          if LCDout then LCDprint;
        end;
    end;
end
```

voltmètre basé sur un FPGA

MAX1000 et VHDPlus facilitent la tâche



Dogan Ibrahim (Royaume-Uni)

Prêt à maîtriser la programmation FPGA ? Le cours *FPGA Programming and Hardware Essentials* est la porte d'entrée idéale pour explorer l'univers des matrices prédiffusées programmables (FPGA). Pour vous en donner un aperçu concret, découvrons un exemple rapide : un voltmètre développé avec le puissant environnement de programmation et implémenté sur le FPGA MAX1000.

Note de la rédaction. *Cet article est un extrait du livre d'Elektor* FPGA Programming and Hardware Essentials. *L'extrait a été formaté et légèrement modifié pour correspondre aux normes éditoriales et à la mise en page du magazine Elektor.* L'auteur et l'éditeur seront heureux de répondre aux questions – pourles contacter, voir l'encadré **« questions ou commentaires ? ».**

Le FPGA est un circuit intégré unique et configurable qui peut être programmé pour exécuter une multitude de tâches habituellement associées à des puces individuelles. Si vous pensiez que la programmation à domicile d'un FPGA était difficile et réservée aux professionnels, laissez-vous surprendre par la suite de cet article !

Présentation du FPGA MAX1000

Le FPGA presenté dans ce livre est principalement la carte de développement MAX1000 d'Arrow Electronics (**figure 1**). Cette carte est spécialement conçue pour faciliter vos premiers pas avec un FPGA. Bien que d'autres chapitres du livre détaillent davantage cette carte, ses caractéristiques de base doivent être mentionnées ici :

- Périphérique Intel MAX10
- Programmateur USB Arrow 2
- SDRAM 64 Mbit
- Mémoire Flash 64 Mbit
- 1× oscillateur MEMS 12 MHz

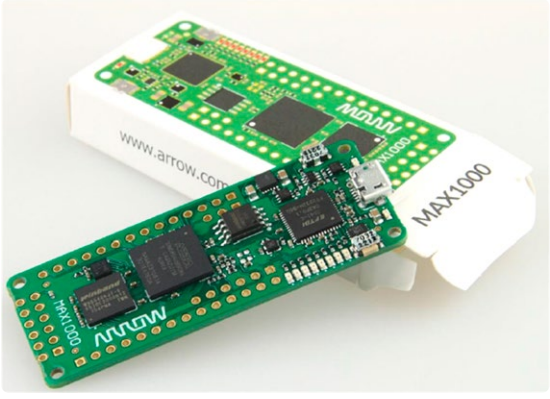


Figure 1. Le FPGA MAX1000.

- 1× oscillateur MEMS optionnel de fréquence préférée
- 8× LED utilisateur rouge
- 2× LED indicateur de carte
- 2× bouton utilisateur
- 1× accéléromètre 3 axes
- 1× connecteur Pmod 12 broches
- 1× connecteur JTAG utilisateur
- 1× connecteur E/S utilisateur
- Connecteur mini USB type B

EDI VHDPlus comme logiciel

De nombreux logiciels peuvent être utilisés pour développer des applications FPGA sur la carte de développement FPGA MAX1000. Dans le livre, vous découvrirez à la fois les fonctionnalités de base et le processus d'installation de l'EDI VHDPlus pour programmer votre MAX1000.
L'EDI VHDPlus est un sur-ensemble de VHDL, qui facilite la programmation en étendant ses fonctionnalités et en simplifiant sa syntaxe. Bien que VHDPlus ne soit pas un langage entièrement différent, il étend les fonctionnalités de VHDL. Ainsi, tout ce qui est réalisable avec VHDL peut également l'être avec VHDPlus. VHDPlus propose une approche moderne qui rend la programmation FPGA plus rapide et plus accessible, en particulier pour les débutants. L'EDI VHDPlus supporte la norme ouverte CRUVI. La simulation d'un programme FPGA conçu peut prendre beaucoup de temps. L'assistant de simulation de l'EDI VHDPlus permet de simuler vos programmes et de corriger rapidement les erreurs. L'EDI VHDPlus intègre les principales fonctionnalités de Quartus et est disponible sur les plateformes Windows et Linux. L'EDI VHDPlus prend également en charge le langage C++, y compris un débogueur.

Installation de l'EDI VHDPlus

Les étapes pour télécharger l'EDI VHDPlus sur votre ordinateur Windows sont les suivantes :

- Allez sur le site Web de VHDPlus [1].
- Cliquez sur l'étape 1 pour télécharger le fichier de *prise en charge des périphériques MAX 10* (**Figure 2**). Le fichier sera téléchargé dans votre dossier *Téléchargements*. Enregistrez-le dans un dossier.
- Faites défiler vers le bas jusqu'à l'étape 2 et téléchargez **Quartus Prime Lite pour Windows**. Le fichier sera téléchargé dans votre dossier *Téléchargements*.
- Cliquez sur le fichier pour installer le programme Quartus Lite.

Mises à jour de la part de l'auteur
Depuis la publication du livre et en réponse aux commentaires des lecteurs, nous avons reçu deux mises à jour de l'auteur concernant principalement le type de FPGA utilisé

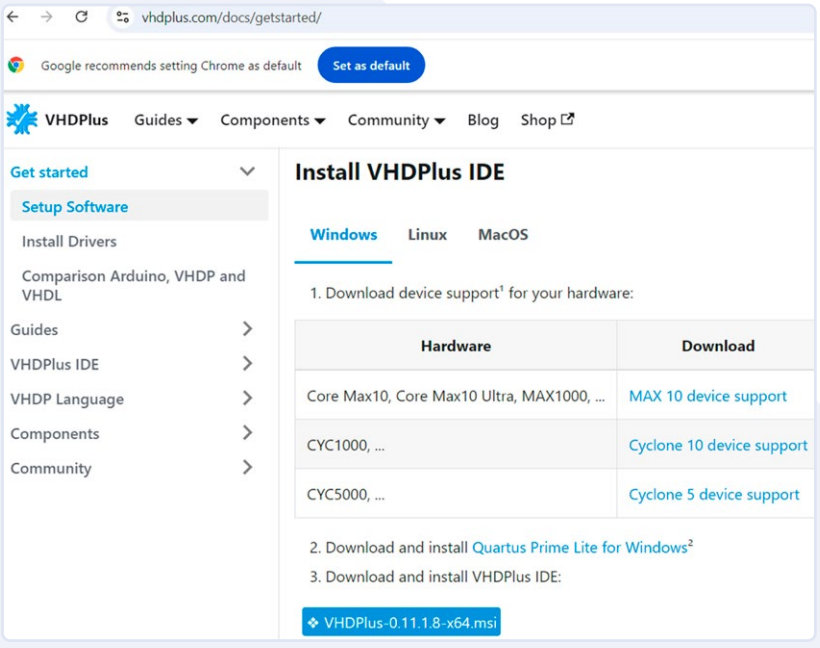
1. Veuillez vous référer à la page 34, deuxième puce :
Une fenêtre s'ouvrira (Figure 3.6) dans laquelle vous pourrez sélectionner la carte de développement que vous utilisez.
- Cette fenêtre est intitulée : *Connect and Compile*.
- En haut à gauche, vous verrez le texte : « FPGA : » suivi d'une liste déroulante.
- Sélectionnez *MAX1000* dans la liste pour la variante 8 kLE (l'option par défaut)
- Sélectionnez *MAX1000 16k* dans la liste pour la variante 16 kLE.

Important : si vous sélectionnez la mauvaise variante, une erreur JTAG se produira lorsque vous tenterez de télécharger le code compilé sur le FPGA.

2. La version 8 kLE est identifiée par le marquage *10M08* sur la grande puce de la carte, tandis que la version 16 kLE est marquée *10M16*

- Lorsque vous démarrez le programme Quartus Lite, vous pouvez obtenir l'erreur suivante : *Vous avez installé avec succès le logiciel Quartus Prime mais vous n'avez installé aucun périphérique. Voulez-vous lancer le programme d'installation des périphériques pour en ajouter* ? Installez le fichier de support des périphériques .qdz comme décrit ci-dessous.
- Allez dans le menu *Démarrer de* Windows et recherchez le programme appelé *Device Installer*(Quartus Prime 18.1).

Figure 2. Téléchargement du fichier de support du dispositif MAX 10.



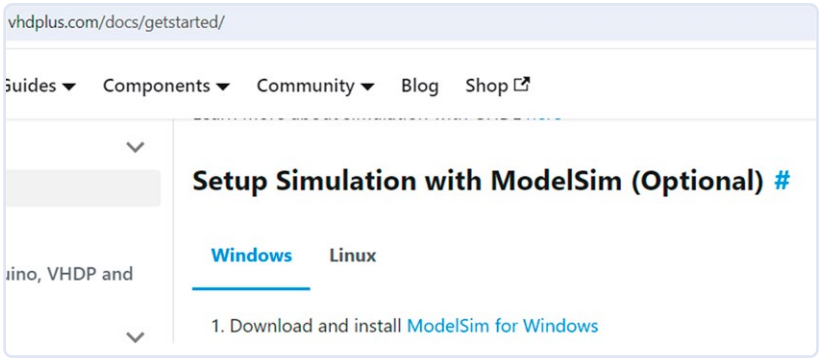


Figure 3. Le téléchargement du simulateur ModelSim.

- Le *Device Installer* vous demandera le dossier dans lequel vous avez enregistré le fichier *.qdz*. Sélectionnez ce dossier.
- Le programme recherchera tous les fichiers .qdz. Choisissez le périphérique à installer. Le fichier .qdz n'est qu'un package d'installation jusqu'à ce que vous l'installiez.
- Cliquez sur l'étape 3 pour télécharger VHDPlus. Le fichier sera téléchargé dans votre dossier *Téléchargements*.
- Cliquez sur le fichier pour installer VHDPlus.

Vous devez installer les pilotes pour que votre programmateur puisse programmer votre FPGA. Procédez comme suit :

- Téléchargez le pilote *Arrow USB Programmer* pour votre système d'exploitation depuis [2].
- Décompressez le fichier téléchargé et exécutez le programme d'installation pour terminer l'installation.

Pour la compilation et la programmation directes avec l'EDI VHDPlus, une connexion à Quartus est nécessaire. Si Quartus est installé dans le répertoire par défaut, aucune autre étape n'est nécessaire.

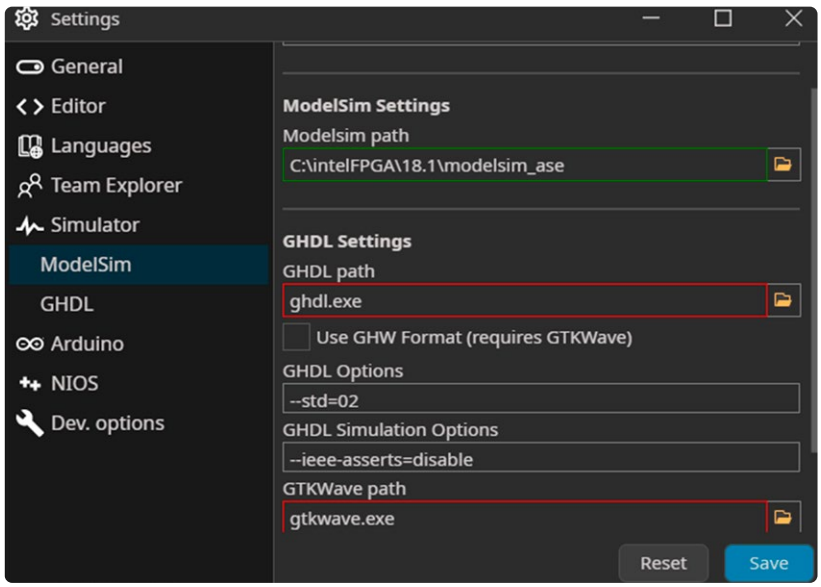


Figure 4. La spécification du chemin d'accès au simulateur dans l'EDI VHDPlus.

Sinon, ajustez le chemin d'accès à Quartus dans l'EDi VHDPlus en allant dans *Extras Settings General*. Une fois Quartus détecté, la bordure deviendra verte.

Configuration de la simulation ModelSim

Pour installer le simulateur ModelSim, suivez les étapes suivantes :

- Visitez le site Web de VHDPlus [1].
- Faites défiler vers le bas et cliquez pour télécharger *ModelSim pour Windows* (**figure 3**). Le fichier sera enregistré dans votre dossier *Téléchargements*.
- Installez *ModelSim* en double-cliquant sur le fichier téléchargé.
- Dans VHDPlus IDE, spécifiez le chemin d'accès au dossier *modelsim_ase* sous *Extras Settings Simulator* (**figure 4**).
- Lancez la simulation en cliquant avec le bouton droit de la souris sur un fichier VHDL.

Convertisseur analogique-numérique (CAN)

Les CAN (ADC) jouent un rôle crucial dans la plupart des applications basées sur des microcontrôleurs. La plupart des capteurs physiques délivrent des signaux de sortie analogiques. Par exemple, un capteur de température analogique peut fournir une tension de sortie directement proportionnelle à la température mesurée. Nous allons explorer les entrées analogiques du FPGA du MAX1000 et développer un projet utilisant ces entrées.

Le MAX1000 FPGA possède neuf entrées analogiques nommées AIN et A0 à A7. Chaque entrée analogique offre une résolution de 12 bits, ce qui correspond à 4096 pas. Avec la tension de référence par défaut de +3,3 V, la résolution d'une entrée ADC est de 3,3 V/4096 = 0,8 mV. Par conséquent, les tensions d'entrée analogiques peuvent être détectées avec une résolution de 0,8 mV. Par exemple, si la tension d'entrée est de 0,8 mV, la sortie du CAN sera binaire *0000 00000001*.

Tableau 1. Canaux ADC du MAX1000.

Broche analogique	Canal
AIN0	1
AIN1	6
AIN2	5
AIN3	7
AIN4	3
AIN5	0
AIN6	2
AIN7	4
AIN	8

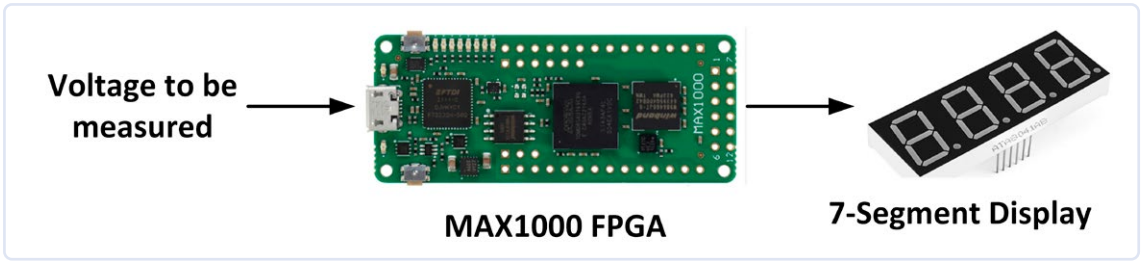


Figure 5. Schéma fonctionnel du projet «voltmètre».

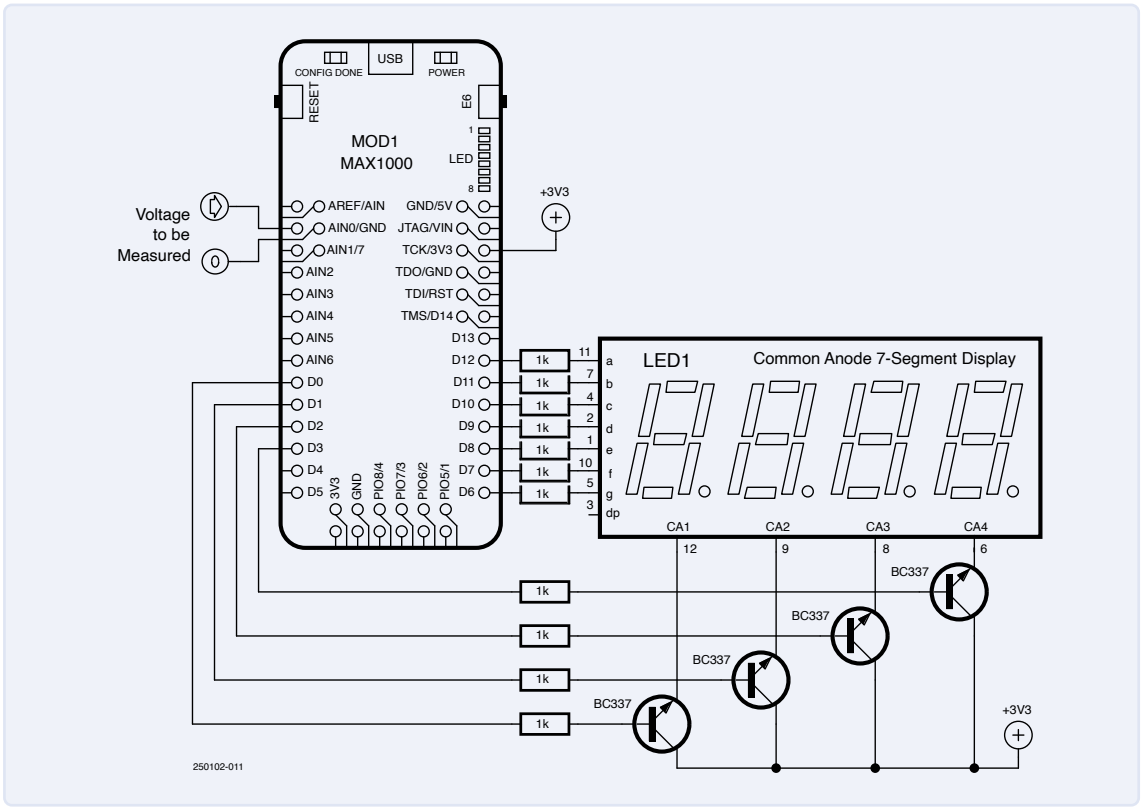


Figure 6. Schéma de circuit du projet «voltmètre» basé sur un FPGA.

Si la tension d'entrée analogique est de 0,16 mV, la sortie du CAN sera *0000 00000010*. De même, si la tension d'entrée analogique est de 2,4 mV, la sortie du CAN sera *0000 00000011*, et ainsi de suite.
Les entrées analogiques se présentent sous la forme de canaux. Le **tableau 1** indique les numéros de canal de chaque entrée analogique.

Projet voltmètre

Matériel

Pour notre tout premier projet, nous allons réaliser un voltmètre simplifié capable de mesurer la tension appliquée à l'une des entrées de l'ADC et l'affiche en millivolts sur un afficheur à 7 segments. La tension d'entrée maximale est de +3,3 V (3300 mV). La **figure 5** illustre le schéma fonctionnel du projet. Le schéma électrique est présenté à la **figure 6**, où la tension à mesurer est appliquée à l'entrée AIN0 (canal 1) du CAN.

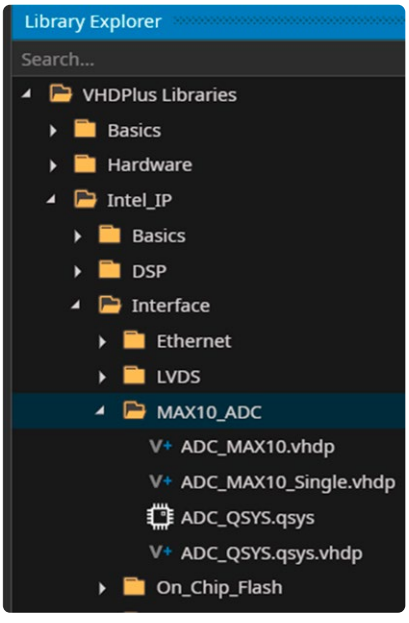


Figure 7. Ajout de la bibliothèque ADC à votre projet.

Figure 8. Confirmation que la bibliothèque ADC a été ajoutée au projet.

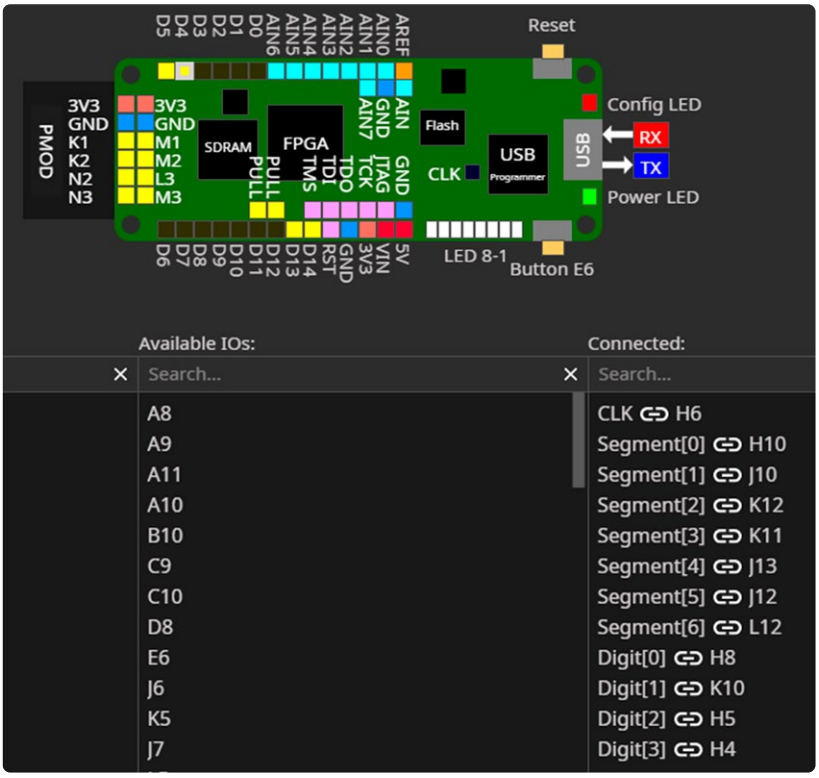


Figure 9. Schéma de connexion du FPGA.

Logiciel

Nous devons utiliser la bibliothèque *ADC* (convertisseur analogique-numérique) pour utiliser les entrées ADC. Les étapes sont les suivantes :

- Ouvrez l'EDI et donnez un nom à votre programme (par exemple, *Voltmètre*).
- Cliquez sur *Library Explorer* dans le coin inférieur gauche de l'écran
- Sélectionnez les éléments suivants (**figure 7**) : *Intel_IP Interface* (Interface) *MAX10_ADC*
- Cliquez avec le bouton droit de la souris sur *MAX10_ADC*, puis sélectionnez *Add to active project*.
- La bibliothèque *ADC* a maintenant été ajoutée à notre projet. Un message devrait apparaître en bas de l'écran indiquant que la bibliothèque *ADC_MAX10 ADC* a été ajoutée au projet actif (**figure 8**).
- Vous pouvez maintenant commencer à écrire votre code en utilisant le composant *ADC_MAX10_Single* de la bibliothèque *ADC*.

La **figure 9** montre le schéma de connexion, qui est essentiellement la connexion de l'afficheur 7 segments, décrite plus en détail dans un autre chapitre du livre.
Le code (Programme : *Voltmeter)* est présenté dans le **listage 1**. La partie du programme relative à la LED à 7 segments est la même que celle que nous avons vue

Listage 1. Le code VHDPlus qui permet au voltmètre de fonctionner sur un FPGA.

```
Main
(
   Segment: OUT STD_LOGIC_VECTOR(6 downto 0);
   Digit: OUT STD_LOGIC_VECTOR(3 downto 0);
)

{
     TYPE MyArray is an array (0 to 9) of STD_LOGIC_VECTOR(1 to 7);
     SIGNAL N: MyArray;
     SIGNAL m: integer range 0 to 9999 := 0;
     SIGNAL temp: integer range 0 to 3300 := 0;
     SIGNAL digits: STD_LOGIC_VECTOR(3 downto 0) := "0001";
     SIGNAL i: integer range 0 to 36000 := 0;
```

```
    SIGNAL ADCdata: natural range 0 to 4095 := 0;
    SIGNAL ADCchannels: natural range 0 to 8 := 0;

    NewComponent ADC_MAX10_Single
    (
        Channel => ADCchannels,
        Data => ADCdata
    );

    N(0) <= "1000000";
    N(1) <= "1111001";
    N(2) <= "0100100";
    N(3) <= "0110000";
    N(4) <= "0011001";
    N(5) <= "0010010";
    N(6) <= "0000010";
    N(7) <= "1111000";
    N(8) <= "0000000";
    N(9) <= "0010000";

 Process()
 {
      if(digits(0) = '1')
      {
         Segment <= N(m mod 10);
      }
      elsif(digits(3) = '1')
      {
         Segment <= N(m/10 mod 10);
      }
      elsif(digits(2) = '1')
      {
         Segment <= N(m/100 mod 10);
      }
      elsif(digits(1) = '1')
      {
         Segment <= N(m/1000 mod 10);
      }
      if(rising_edge(CLK))
      {
       i <= i + 1;
       if(i = 36000)
       {
            i <= 0;
            digits <= digits(2 downto 0) & digits(3);
            Digit <= digits;
       }
       ADCchannels <= 1;            -- Read channel 1 (AIN0)
       temp <= ADCdata;             -- As raw data
       m <= temp * 3300 / 4095;     -- in millivolts
      }
 }
}
```

au chapitre 5. De plus, nous avons défini les signaux `temp`, `ADCdata` et `ADCchannels`. Le signal `temp` va de 0 à 3300, ce qui correspond à la tension de sortie maximale de l'ADC. ADCdata est la donnée brute de l'ADC qui peut prendre une valeur comprise entre 0 et 4095, où 4095 correspond au binaire *1111 11111111*. `ADCchannels` est le numéro de canal et peut prendre les valeurs de 0 à 8.

Un nouveau composant nommé `ADC_MAX10_Single` est défini dans le programme et utilisé pour accéder au module `ADC` unique dans la bibliothèque *ADC*. Ce composant possède deux variables dans la bibliothèque : le numéro de canal (`Channel`) et les données du canal (`Data`).

Sur le front montant de l'horloge, le canal 1 est sélectionné, ce qui correspond à l'entrée AIN0 de l'ADC. Les données brutes de l'ADC sont copiées dans la variable `temp`. Cette valeur est ensuite multipliée par 3300 et divisée par 4095 afin que la lecture brute soit convertie en millivolts physiques. La copie de cette valeur dans `m` affiche la lecture de l'ADC, c'est-à-dire la lecture du voltmètre sur l'afficheur à 7 segments.

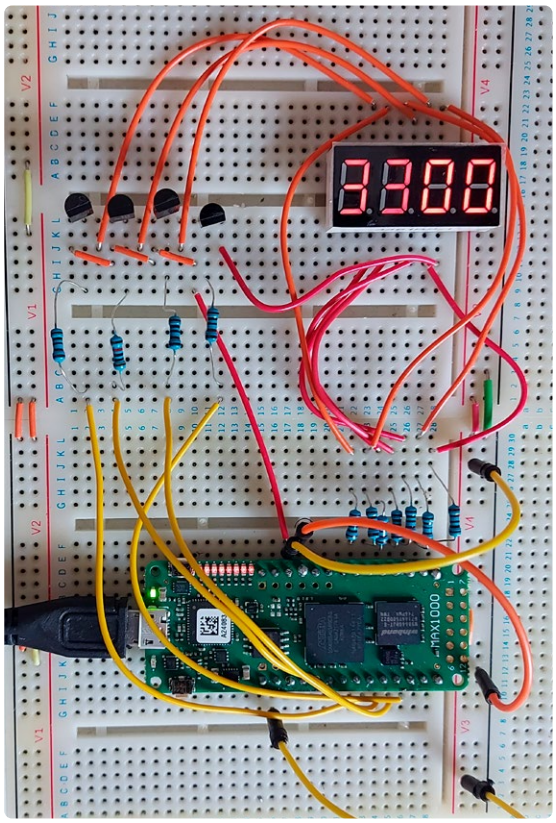


Figure 10. Le projet «voltmètre» réalisé sur une plaque d'essai.

Testons-le !

Enfin, la **figure 10** illustre le projet tel qu'il a été réalisé par l'auteur pour l'édition de cet ouvrage. Malgré ses capacités limitées, ce modeste voltmètre constitue une excellente initiation à l'univers captivant de la programmation FPGA, en utilisant des équipements et des logiciels peu onéreux et facilement disponibles. Quel sera votre prochain projet ?

250102-04

Questions ou commentaires ?

Envoyez un courriel à l'auteur (d.ibrahim@btinternet.com), ou contactez Elektor (redaction@elektor.fr).

À propos de l'auteur

Dogan Ibrahim est diplômé en ingénierie électronique, titulaire d'un Master en ingénierie de contrôle automatique et d'un doctorat en traitement numérique du signal. Avant de revenir à l'université, il a œuvré dans plusieurs organisations industrielles. Le professeur Ibrahim a écrit plus de 70 ouvrages techniques et publié plus de 200 articles dans les domaines des microcontrôleurs et microprocesseurs. Il est ingénieur électricien certifié et membre de l'Institution of Engineering Technology, ainsi que professionnel Arduino certifié.

Produits

- **Dogan Ibrahim, MAX1000 FPGA Programming Bundle (Elektor 2024)**
  www.elektor.fr/21082
- **Dogan Ibrahim, *FPGA Programming and Hardware Essentials* (Elektor 2024, Book)**
  www.elektor.fr/21054
- **Dogan Ibrahim, *FPGA Programming and Hardware Essentials* (Elektor 2024, E-Book)**
  www.elektor.fr/21055

LIENS

[1] VHDPlus website: https://vhdplus.com/docs/getstarted/#install-vhdplus-ide
[2] Arrow USB Programmer driver, Trenz Electronic: https://tinyurl.com/Arrow-USB-Programmer

REJOIGNEZ NOTRE COMMUNAUTÉ

TÉLÉCHARGEZ GRATUITEMENT

Abonnez-vous maintenant à **elektormagazine.fr/ezine-24**

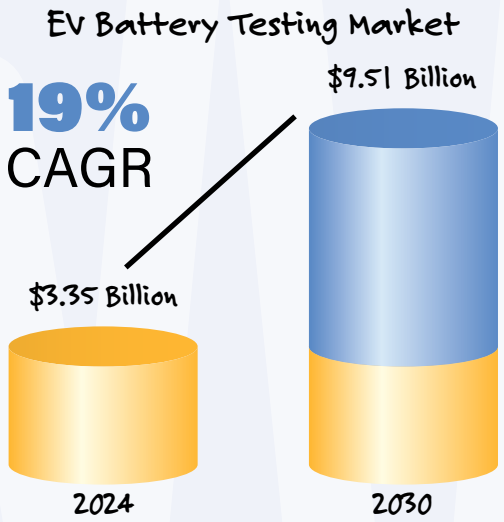
elektor
MEDIA & LEARNING

Les tests de batteries des véhicules électriques

Sur le marché des équipements de test pour véhicules électriques (VE), nous observons une augmentation de la demande pour des outils spécialisés comme les testeurs de batterie et les systèmes de gestion thermique, qui sont essentiels pour évaluer les performances critiques des véhicules électriques [1]. Le secteur des véhicules électriques à batterie (VEB) est dominant et stimule la croissance des équipements de test dédiés à ces véhicules en raison de leurs besoins spécifiques en matière de test. Avec environ 40 millions de véhicules électriques en circulation en 2023 [2], les VEB nécessitent des tests approfondis de leurs batteries, de leur transmission et de leurs interfaces de charge pour assurer leurs performances, leur efficacité et leur sécurité. Les avancées dans les technologies de batterie, notamment les chimies lithium-ion à semi-conducteurs et à haute capacité, exigent des protocoles spécialisés pour valider leur efficacité.

Le secteur des tests de performance devrait enregistrer la croissance la plus rapide d'ici 2033. Des entreprises telles que TÜV SÜD se positionnent comme leaders dans ce secteur grâce à leur expertise dans le domaine des batteries lithium-ion. Elles proposent des services tels que des tests de vieillissement cyclique et calendaire, des profils de charge rapide et la spectroscopie d'impédance électrochimique pour garantir des performances optimales des batteries [3].

Principaux acteurs
- TÜV SÜD
- Bureau Veritas
- SGS Societe Generale De Surveillance SA
- TÜV Rheinland AG Group
- Dekra
- Applus+



Source : Research and Markets [3]

Les oscilloscopes restent en tête

Le marché des équipements de test à usage général (GPTE) bénéficie d'une croissance soutenue, portée par une demande accrue pour des outils de mesure de haute précision dans diverses industries. Selon Technavio [7], ce marché est projeté à augmenter de 2,06 milliards de dollars entre 2023 et 2028.

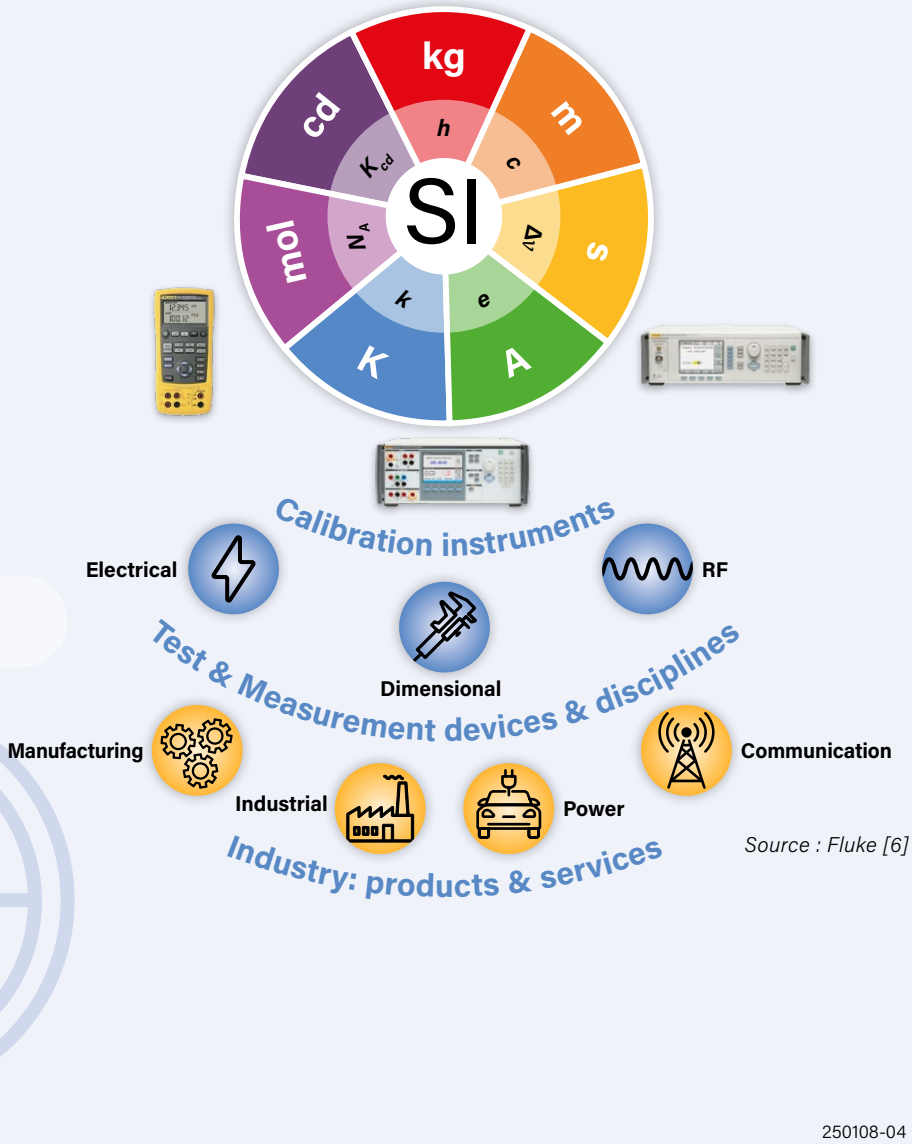
Au sein de ce secteur, les oscilloscopes occupent une position prédominante, représentant la plus grande part de marché. Ils devraient voir leur chiffre d'affaires passer de 3,74 milliards de dollars en 2025 à 5,49 milliards de dollars en 2030, affichant ainsi un taux de croissance annuel composé de 7,99 %.

Les leaders du marché des oscilloscopes
- Tektronix
- Keysight Technologies
- Rohde & Schwarz
- Teledyne LeCroy
- Yokogawa Test & Measurement Corporation

Source : Mordor Intelligence [8]

L'étalonnage

À mesure que la fabrication de haute précision renforce ses normes de qualité, le secteur des services d'étalonnage est en plein essor et devrait connaître un taux de croissance annuel composé (TCAC) de 4,5 % entre 2025 et 2033 [4]. La métrologie joue un rôle clé en assurant une exactitude rigoureuse tant en pré-production qu'en post-production, en maintenant la conformité avec des normes strictes. Cependant, l'évolution des normes d'étalonnage ajoute à la complexité, confrontant les prestataires à la nécessité de rester en permanence à la pointe de la technologie [5]. Parallèlement, l'émergence des appareils et des solutions logicielles d'étalonnage automatique pourrait bouleverser les pratiques traditionnels, obligeant les prestataires à s'adapter.



Source : Fluke [6]

250108-04

LIENS

[1] Towards Automotive, "Electric Vehicle Test Equipment Market Size, Trends and Developments," September 2024: https://tinyurl.com/EV-market
[2] IEA, "Trends in Electric Cars," 2024: https://www.iea.org/reports/global-ev-outlook-2024/trends-in-electric-cars
[3] Research and Markets, "EV Battery Testing Market," December 2024: https://www.researchandmarkets.com/reports/6036229
[4] IMARC Group, "Calibration Services Market," 2024: https://www.imarcgroup.com/calibration-services-market
[5] MarketsandMarkets, "Calibration Services Market," 2023: https://tinyurl.com/mam-calibration-market
[6] Fluke, "Why is Calibration Important?": https://www.fluke.com/en-us/learn/blog/calibration/why-is-calibration-important
[7] Technavio, "General Purpose Test Equipment (GPTE) Market," Aug 2024: https://tinyurl.com/technavio-GPTE-market
[8] Mordor Intelligence, "Oscilloscope Market," 2024: https://www.mordorintelligence.com/industry-reports/oscilloscope-market

oscilloscope / multimètre / générateur

Fnirsi 2C53T : avec deux voies et une bande passante de 50-MHz

Figure 1. Le Fnirsi 2C53T est compact et remplace trois appareils de mesure séparés.

Harry Baggen (Pays-bas)

Le fabricant chinois Fnirsi ne cesse de lancer de nouveaux produits. Cette fois, nous examinerons le 2C53T, un instrument de mesure compact 3-en-1 qui combine un oscilloscope, un multimètre et un générateur de fonctions. Il ressemble exactement au 2C23T, que j'ai examiné il y a environ un an. Alors, quoi de neuf ?

Le Fnirsi 2C53T [1] est un instrument de mesure compact 3-en-1 qui combine un oscilloscope, un multimètre et un générateur de fonctions (**figure 1**). Le fabricant chinois Fnirsi ne cesse pas de lancer de nouveaux produits. Il s'agit parfois de versions améliorées d'appareils déjà commercialisés, comme c'est le cas du 2C53T que nous avons testé ici. De l'extérieur, l'appareil ressemble exactement au 2C23T, que j'ai examiné il y a environ un an [2].

Rafraîchir votre mémoire

Comme je l'ai déjà noté, le boîtier reste exactement le même, avec seulement de légères modifications au niveau du marquage de certains boutons poussoirs. Pour ceux qui ne connaissent pas son prédécesseur, voici les principales caractéristiques physiques du 2C23T et du Fnirsi 2C53T. L'appareil a un boîtier assez petit mesurant 17 × 9 × 3,5 cm, doté d'un écran LCD couleur de 2,8 pouces qui fournit une image raisonnablement lumineuse avec un bon contraste. Le boîtier est doté d'un matériau bleu semblable à du caoutchouc sur les coins et semble assez robuste. L'alimentation est assurée par une batterie au lithium intégrée de 3 Ah, qui dure environ six heures d'utilisation.
En façade, on retrouve quatre prises banane pour le multimètre, et en haut de l'appareil, trois prises BNC : deux pour l'oscilloscope (à deux voies) et une pour le générateur de fonctions. L'appareil fonctionne via 15 boutons-poussoirs. Un connecteur USB-C sur le côté permet de charger la batterie au lithium interne et de se connecter à un PC pour les mises à jour du micrologiciel et les téléchargements de captures d'écran.

Plusieurs accessoires inclus

Le 2C53T est désormais livré dans un étui de rangement pratique qui offre également de la place pour le jeu de cordons de test du multimètre inclus, deux sondes d'oscilloscope 100 MHz (le 2C23T n'en avait qu'une), un câble BNC avec pinces crocodiles pour le générateur, un câble USB-C et un petit manuel (**figure 2**). Comme pour la plupart des appareils Fnirsi, tout est soigneusement fini et bien emballé. Même l'étui de rangement inclus est robuste et conçu de manière pratique.

Figure 2. Avec le 2C53T, vous recevez deux sondes d'oscilloscope et un étui de rangement pratique.se.

Différences entre le Fnirsi 2C53T et le 2C23T

Avec le 2C53T [1], les spécifications et les capacités de la fonction oscilloscope ont été considérablement améliorées et étendues. La bande passante d'entrée est passée de 10 MHz à 50 MHz et le taux d'échantillonnage a été augmenté de 50 à 250 Méchantillons/s (**figure 3**). C'est exceptionnel pour un appareil de mesure dans cette gamme de prix ! De plus, l'oscilloscope comprend désormais un mode X-Y et propose huit fonctions mathématiques parmi lesquelles choisir. Il existe également une analyse FFT, même si elle n'est pas particulièrement utile pour un appareil de ce type.
Une fonction dite *Persistance* a été ajoutée, permettant aux signaux de « briller » pendant une certaine période, de la même manière que les anciens oscilloscopes à tubes cathodiques affichaient les formes d'onde. Le nombre de valeurs de mesure pouvant être affichées à l'écran est passé à 14 par canal. Enfin, tous les paramètres de l'oscilloscope ont été regroupés dans un menu dédié, rendant la navigation bien plus claire.

Mises à jour du multimètre

Le multimètre n'a pas beaucoup changé. La résolution est passée de 4 chiffres (9999) à 4½ chiffres (19999), et l'affichage a été repensé avec une disposition plus épurée et plus claire. Il n'y a désormais qu'une seule échelle analogique (**figure 4**). Les spécifications et les capacités restent les mêmes : une précision de base de 0,5%, la possibilité de mesurer la tension et le courant continu et alternatif, ainsi que la résistance, la capacité et la température. Un testeur de continuité et un testeur de diodes sont également inclus.
Comme avec la plupart des multimètres Fnirsi, l'appareil détecte automatiquement si une tension continue, alternative ou une résistance est appliquée aux entrées, mais vous pouvez également passer manuellement à une fonction spécifique.

Modifications du générateur de fonctions

Certaines modifications ont été apportées au générateur de fonctions intégré. Le nombre de formes d'onde disponibles est passé de six à treize. Le signal de sortie a désormais une amplitude maximale de 3 V crête à crête (auparavant 3,3 V). Cependant, l'inconvénient est que la plage de fréquences a été réduite à 50 kHz (contre 1 MHz sur le 2C23T). Il faut bien faire quelques compromis.

Premiers pas avec le Fnirsi 2C53T

Si vous avez déjà travaillé avec un 2C23T, vous vous sentirez parfaitement à l'aise avec le 2C53T. Mis à part quelques détails mineurs, le fonctionnement reste le même. Lorsque vous allumez l'appareil, un menu apparaît avec différentes icônes vous permettant de sélectionner un instrument de mesure ou d'accéder au menu des paramètres. Il est également possible de configurer le 2C53T pour démarrer directement avec un instrument de mesure spécifique.
L'oscilloscope est l'instrument le plus important de cet appareil, principalement en raison de sa large bande passante et de son taux d'échantillonnage élevé. Malheureusement, l'écran n'a pas été agrandi, il semble donc toujours assez encombré, surtout lorsque plusieurs mesures sont affichées. L'oscilloscope réagit rapidement aux changements de signal et son fonctionnement a été quelque peu amélioré grâce à l'ajout d'un menu de paramètres dédié. Le réglage de la base de temps, de la sensibilité, du niveau de déclenchement et de la position du signal sur l'écran est désormais plus clair : en appuyant sur la touche Select, vous pouvez choisir lequel de ces paramètres peut être ajusté à l'aide des touches du pavé directionnel. Les paramètres sélectionnés sont affichés en haut de l'écran.
Il est difficile de déterminer si la fonction Auto-Setup a été mise à jour, mais j'ai eu l'impression qu'elle fonctionne légèrement plus rapidement que sur son prédécesseur. Les connecteurs BNC en

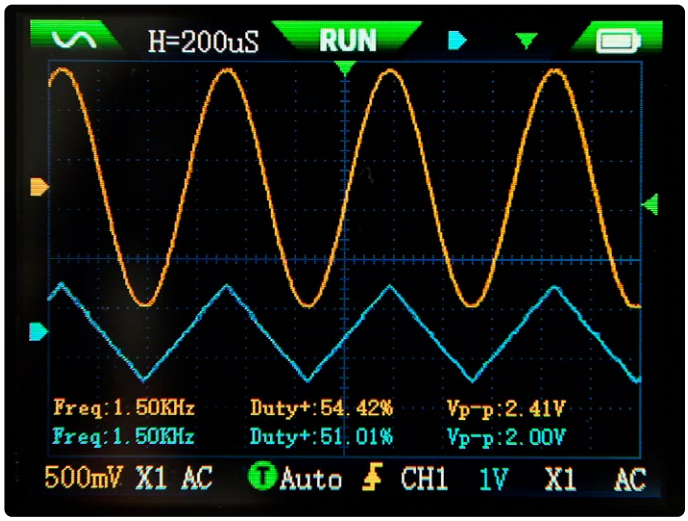


Figure 3. L'oscilloscope est désormais utilisable jusqu'à 50 MHz et semble également répondre plus rapidement que le 2C23T.

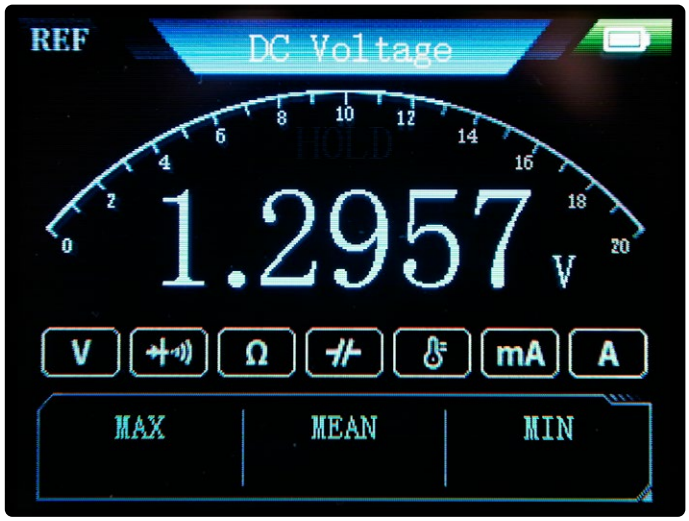


Figure 4. L'écran du multimètre a un design plus épuré, il n'y a désormais plus qu'une échelle analogique et cela suffit.

haut sont toujours très proches les uns des autres, ce qui signifie que les sondes avec couvercles en plastique ne s'adapteront pas. Heureusement, deux sondes compatibles sont incluses.
La bande passante de l'oscilloscope s'étend effectivement jusqu'à au moins 50 MHz. Cependant, l'amplitude du signal a légèrement augmenté de 10 MHz à 50 MHz, ce qui a également provoqué une légère variation des valeurs mesurées. Cela est probablement dû à un étage d'entrée légèrement non linéaire. Cependant, une fois que vous en êtes conscient, vous pouvez compenser en conséquence.

Performances du multimètre Fnirsi 2C53T

Comme le 2C23T, le multimètre du Fnirsi 2C53T est doté d'une détection automatique, mais il ne fonctionne pas en dessous de 0,7 V. Si nécessaire, vous pouvez changer manuellement de mode si vous pensez que la valeur affichée est incorrecte ou si une valeur de résistance est affichée à la place d'une lecture de tension.
Je trouve l'affichage du multimètre une amélioration par rapport au 2C23T, car il comprend désormais une seule échelle analogique au-dessus de la valeur mesurée. Les valeurs mesurées minimum, maximum et moyenne sont affichées en bas. Une fois de plus, j'ai testé la précision du multimètre sur différentes valeurs de tension, de courant, de résistance et de capacité, et les résultats étaient identiques à ceux du 2C23T. En fait, la précision est au moins deux fois supérieure aux spécifications indiquées, ce qui est très impressionnant !
Cela dit, je remets en question la décision d'augmenter le nombre de chiffres, car ils ne sont pas particulièrement nécessaires étant donné la précision de base de 0,5%. Cependant, cela pourrait être utile pour des mesures comparatives. Le multimètre comprend deux fusibles internes pour les deux plages de courant, qui peuvent être remplacés en dévissant le boîtier.
Enfin, il est un peu décevant que, comme sur son prédécesseur, les quatre prises d'entrée du multimètre soient légèrement moins espacées que le standard 19 mm, ce qui le rend incompatible avec certains accessoires standards. Fnirsi aurait probablement dû élargir le boîtier un peu pour s'adapter au bon espacement.
Bien que le générateur de fonctions offre désormais plus d'options de forme d'onde, la qualité du signal ne s'est pas améliorée. L'onde sinusoïdale apparaît clairement légèrement aplatie en bas (**figure 5**). Un zoom avant révèle les étapes relativement grossières qui composent le signal, ce qui le rend approprié uniquement pour les tests généraux où la qualité du signal n'est pas une priorité. La limitation de la plage de fréquences à 50 kHz est particulièrement notable, ce qui représente une réduction significative par rapport au 2C23T.
Le fonctionnement du générateur reste inchangé. La fréquence peut toujours être réglée avec précision par hertz. La tension de sortie maximale est désormais de 3 V crête à crête dans des conditions sans charge. L'impédance de sortie a été ajustée et est maintenant d'environ 50 Ω. Cependant, gardez à l'esprit que le signal de sortie

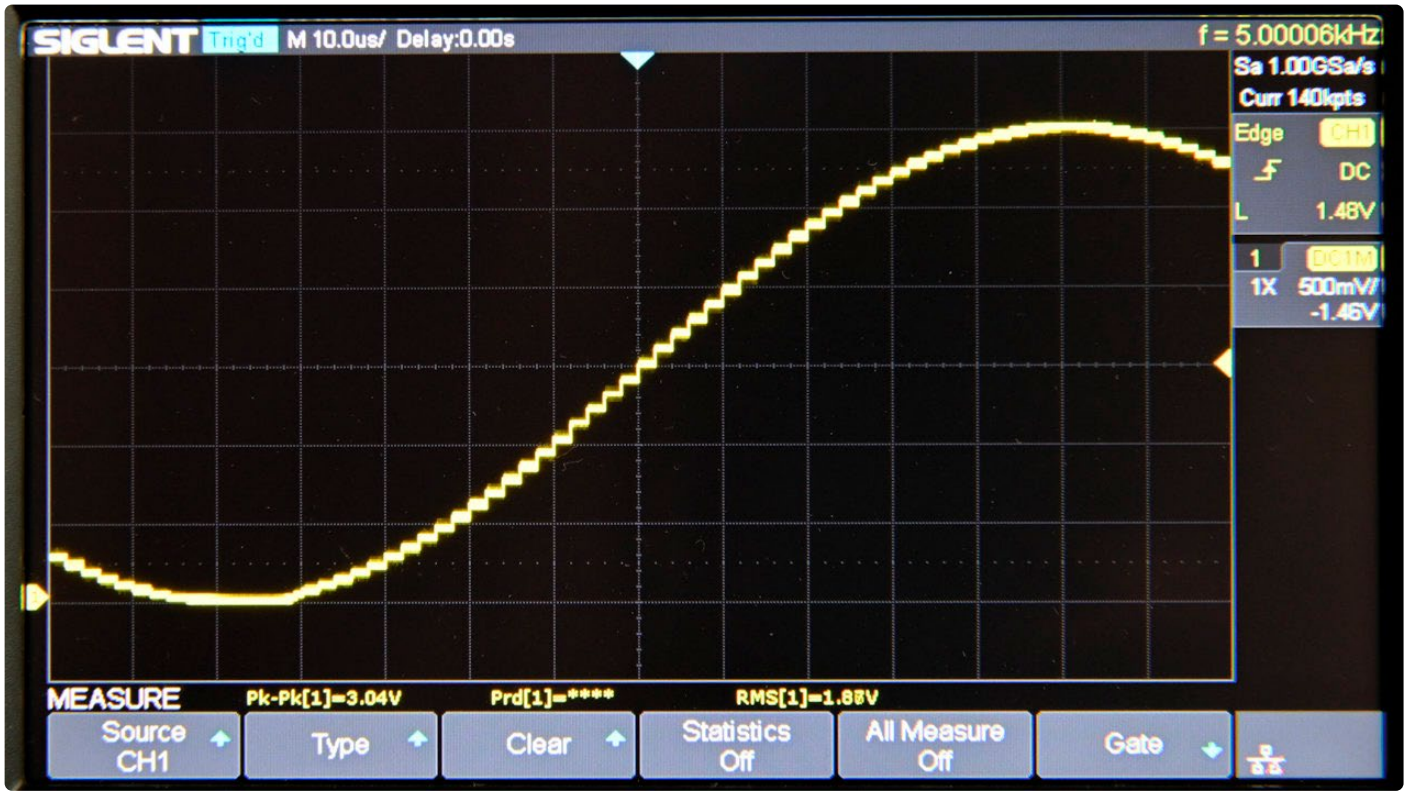


Figure 5. Les signaux du générateur de fonctions sont assez grossiers, on voit clairement les pas de quantification.

est entièrement positif par rapport à la masse, car il n'y a pas de condensateur de sortie ni d'alimentation symétrique.

Conclusion

Le nouveau Fnirsi 2C53T [1] peut coûter quelques euros de plus que son prédécesseur, le 2C23T, mais il apporte des améliorations significatives dans la section oscilloscope. En particulier, la bande passante de 50 MHz et le taux d'échantillonnage de 250 Méchantillons/s constituent des améliorations majeures.
Même si le multimètre reste largement inchangé et que la plage de fréquences du générateur de fonctions est considérablement réduite, je choisirais toujours le 2C53T uniquement pour ses capacités d'oscilloscope. De plus, à ce prix, vous obtenez également deux sondes d'oscilloscope et un étui de rangement pratique, un joli bonus !

250137-04

Produits

- **FNIRSI 2C53T (3-en-1) Oscilloscope 2 voies (50 MHz) + Multimètre + Générateur de signaux**
  www.elektor.fr/21112
- **FNIRSI 2C53P (3-en-1) Oscilloscope 2 voies (50 MHz) + Multimètre + Générateur de signaux**
  www.elektor.fr/20917
- **FNIRSI 2C23T (3-en-1) Oscilloscope 2 voies (10 MHz) + Multimètre + Générateur de signaux**
  www.elektor.fr/20717

LIENS

[1] Fnirsi 2C53T : https://www.elektor.fr/products/fnirsi-2c53t-upgrade-3-in-1-2-ch-oscilloscope-50-mhz-multimeter-signal-generator
[2] H. Baggen, «oscilloscope, multimètre et générateur de signaux FNIRSI 2C23T» elektormagazine.fr, mai 2024 : https://www.elektormagazine.fr/review/essais-oscilloscope-multimetre-et-generateur-de-signaux-fnirsi-2c23t

Vous concevez. Nous délivrons.

Les dernières nouveautés pour vos conceptions les plus récentes™

mouser.fr/new

MOUSER ELECTRONICS