

Synchronisation du contrôle de mouvements multi axes sur réseaux fonctionnant en temps réel

Par Jens Sorensen, Dara O'Sullivan et Christian Aaen, Analog Devices

Synthèse

Les protocoles Ethernet temps réel et déterministes tels qu'EtherCAT ont rendu possible le fonctionnement synchronisé de systèmes de contrôle de mouvements multi axes¹. Dans le cadre de cette synchronisation, deux aspects entrent en jeu : d'une part, l'envoi de commandes et de références entre les différents nœuds de contrôle doit être synchronisé par rapport à une horloge commune, et d'autre part, l'exécution des algorithmes de commande et des fonctions de rétroaction doit être synchronisée sur la même horloge. Si le premier mode de synchronisation est bien connu et constitue un élément à part entière des contrôleurs de réseau, le second a été longtemps négligé et est à présent à l'origine de goulots d'étranglement qui pénalisent les performances du contrôle de mouvements.

Dans cet article, nous présentons des concepts originaux de synchronisation des commandes moteur, depuis le contrôleur de réseau jusqu'aux bornes du moteur et aux capteurs. Les technologies décrites permettent d'améliorer considérablement la synchronisation, avec à la clé une augmentation significative des performances de contrôle.

Description du problème et état de l'art

Pour définir les limites des solutions qui représentent l'état de l'art, nous prendrons l'exemple d'un système de contrôle de mouvements en réseau sur 2 axes (Figure 1). Un organe de contrôle de mouvements « maître » envoie des commandes et des références sur un réseau en temps réel à deux servocontrôleurs, chacun constituant un nœud esclave sur le réseau. Le servocontrôleur proprement dit se compose d'un contrôleur de réseau, d'un contrôleur moteur, d'un onduleur de puissance et d'un moteur/encodeur.

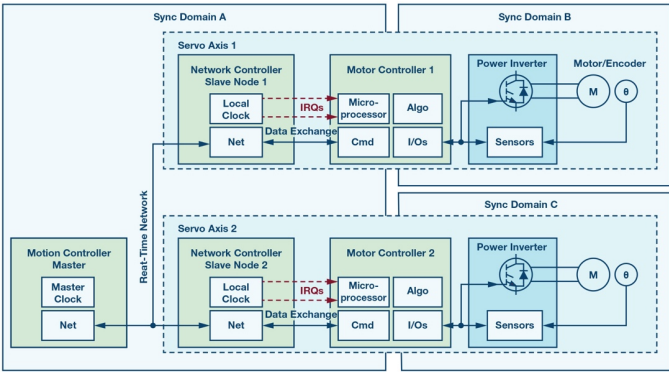


Figure 1. Système classique de contrôle de mouvements en réseau sur 2 axes.

Les protocoles réseau en temps réel utilisent différentes méthodes pour synchroniser les nœuds esclaves avec le maître. L'une des approches les plus couramment utilisées consiste à équiper chaque nœud d'une horloge locale synchronisée. Cette « compréhension » mutuelle du paramètre « temps » permet de synchroniser étroitement les références et les commandes pour l'ensemble des axes des servocontrôleurs. Autrement dit, tous les contrôleurs réseau connectés au réseau temps réel sont synchronisés.

De manière générale, on compte deux lignes d'interruption entre le contrôleur de réseau et le contrôleur de moteur. La première avertit le contrôleur de moteur qu'il doit rassembler des informations et les mettre sur le réseau. La seconde signale au contrôleur de moteur à quel moment il doit lire les données mises à sa disposition sur le réseau. Suivant cette approche, l'échange de données entre le contrôleur de mouvement et le contrôleur de moteur se déroule de façon synchronisée et un timing très précis est possible. Cependant, il ne suffit pas de transmettre des données synchronisées aux contrôleurs moteur : ces derniers doivent également être capables de réagir aux données de manière synchronisée, faute de quoi, les contrôleurs moteur ne pourront bénéficier de la précision du timing du réseau. En ce qui concerne la réaction aux références et aux commandes, les entrées/sorties des contrôleurs moteur posent un problème.

Chacune des entrées/sorties d'un contrôleur moteur, qu'il s'agisse de signaux de modulation à largeur d'impulsion (MLI) ou de convertisseurs analogique/numérique, intègre sa propre quantification du temps et des retards. Prenons l'exemple d'une horloge MLI qui génère des signaux de commande de grille destinés à l'onduleur (Figure 2). L'horloge génère des signaux de commande de grille en comparant la référence M_x à un compteur-décompteur. Lorsque la référence M_x est modifiée par l'algorithme de commande, le nouveau rapport cyclique ne prend effet qu'à la période MLI suivante. Ceci équivaut à un effet de maintien qui signifie que le rapport cyclique n'est mis à jour qu'une seule fois par période MLI (T) ou deux fois, si un mode de double mise à jour est utilisé.

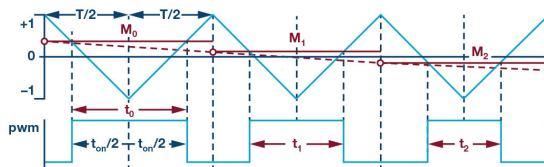


Figure 2. Evolution du rapport cyclique de l'horloge MLI.

Quel que soit le niveau de synchronisation des échanges de données sur le réseau temps réel, la quantification temporelle de l'horloge MLI finit par constituer le facteur déterminant de la synchronisation d'un axe. Lorsqu'une nouvelle référence est reçue, il n'est pas possible d'y répondre avant le début d'un nouveau cycle. Cette approche introduit une incertitude temporelle qui peut correspondre à une période MLI — soit, généralement, entre 50 et 100 μ s. Dans les faits, il existe une relation de phase à la fois variable et non définie entre la période de synchronisation du réseau et la période MLI. Si l'on compare cette valeur à une incertitude temporelle inférieure à 1 μ s sur le réseau temps réel, il apparaît clairement que les entrées/sorties du contrôleur de moteur jouent un rôle crucial dans la synchronisation du contrôle des mouvements dans un réseau. En fait, ce n'est pas le réseau en temps réel qui détermine la précision de la synchronisation : ce sont plutôt les entrées/sorties du système.

Toujours en Figure 1, le système compte trois domaines de synchronisation — A, B et C — qui ne sont pas reliés les uns aux autres : ils sont désynchronisés avec une incertitude variable pouvant atteindre la durée d'une période MLI.

Incertitudes liées à la synchronisation et répercussions sur les applications

L'impact de l'incertitude sur le timing est clairement visible dans les systèmes asservis multi axes de hautes performances utilisés dans des applications telles que la robotique et l'usinage. Le décalage temporel variable entre les axes de commande moteur au niveau des entrées/sorties a un impact direct et quantifiable sur la précision du positionnement tridimensionnel final du robot ou de la machine-outil.

Prenons à présent le cas d'un mouvement simple, tel que représenté en Figure 3. Dans cet exemple, la référence de vitesse du moteur (courbe bleue) augmente avant de redescendre. Si la pente de la rampe est compatible avec les capacités du système électromécanique, la vitesse réelle doit suivre la référence. Cependant, en cas de retard en n'importe quel point du système, la vitesse réelle (courbe rouge) sera

inférieure à la référence, entraînant une erreur de position, $\Delta\theta$.

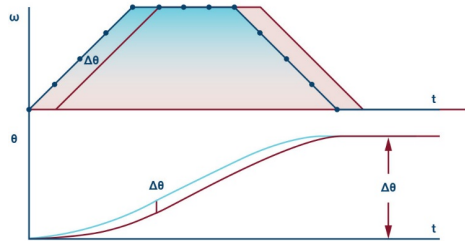


Figure 3. Effet du retard sur la précision de la position.

Dans les machines multi axes, une position cible (x, y, z) correspond à des profils d'axes angulaires $(\theta_1, \dots, \theta_n)$ qui varie en fonction de la construction mécanique de la machine. Ces profils définissent une séquence de commandes de position/vitesse espacées de façon égale dans le temps et pour chaque axe. Tout écart de timing entre les axes réduit la précision de la machine. Prenons à présent l'exemple à 2 axes illustré à la Figure 4. Le trajet cible de la machine est décrit par un ensemble de coordonnées (x, y) . Un retard provoque une erreur de timing au niveau de la commande de l'axe y , avec à la clé une trajectoire effective irrégulière.

Dans certains cas, l'impact d'un retard constant peut être minimisé par une compensation adaptée. Un retard variable et inconnu qu'il est impossible de compenser crée une situation plus délicate. De plus, un retard variable entraîne une variation du gain de la boucle de régulation, rendant son ajustement difficile pour atteindre des performances optimales.

Il est à noter qu'un retard survenant en n'importe quel point du système entraînera une imprécision dans la machine, de sorte que la réduction ou l'élimination des retards permet d'accroître la productivité ainsi que la qualité du produit final.

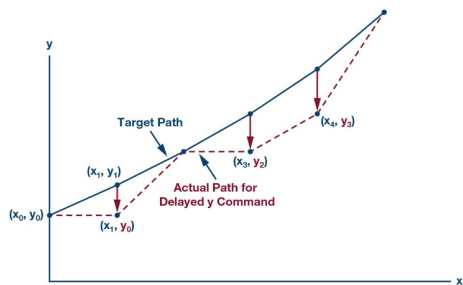


Figure 4. Impact du retard de synchronisation sur la précision de la position.

Synchronisation et nouvelles topologies de contrôle

L'approche traditionnelle du contrôle de mouvements est représentée dans la partie supérieure de la Figure 5. Un contrôleur de mouvement, généralement un automate programmable (PLC), envoie des références de position (θ^*) à un contrôleur moteur via un réseau temps réel. Le contrôleur moteur est composé de trois boucles de rétroaction montées en cascade : la boucle interne qui contrôle le couple/courant (T/i), la boucle centrale qui contrôle la vitesse (ω) et la troisième boucle chargée de

contrôler la position (θ). La boucle de couple affiche la plus grande bande passante, et la boucle de position a la bande passante la plus faible. La rétroaction est maintenue en local par le contrôleur moteur et est étroitement synchronisée avec l'algorithme de commande et le modulateur à largeur d'impulsions.

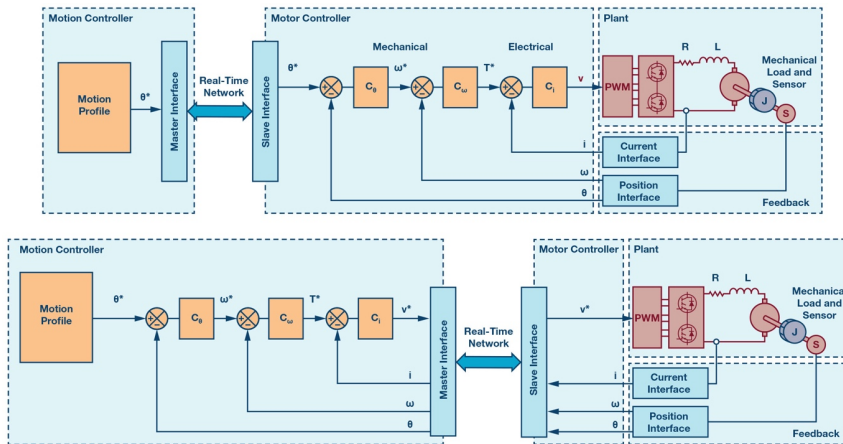


Figure 5. Topologies de contrôle de mouvements : traditionnelle (en haut) et émergente (en bas).

Avec ce type de système, les axes sont synchronisés lors de l'échange des références de position entre le contrôleur de mouvement et le contrôleur moteur. La corrélation avec la synchronisation des entrées/sorties du contrôleur moteur (rétroaction et MLI) ne devient problématique que pour des applications de très haute précision comme l'usinage à commande numérique (CNC). La boucle de position est souvent pénalisée par une bande passante relativement faible, ce qui la rend moins sensible à la synchronisation des entrées/sorties. En d'autres termes, la synchronisation des nœuds à un niveau de référence assure généralement des performances acceptables, même si le réseau et les entrées/sorties opèrent dans des domaines de synchronisation différents.

Si la topologie de commande illustrée dans la partie supérieure de la Figure 5 est courante, d'autres approches sont également utilisées avec un partitionnement des commandes dans lesquels les boucles de position et/ou de vitesse sont mises en œuvre côté contrôleur de mouvement et les références de vitesse/couple sont transmises via le réseau. Les tendances qui se dessinent depuis peu dans l'industrie témoignent d'une évolution vers une nouvelle méthode de partitionnement où toutes les boucles de commande sont séparées des contrôleurs moteur afin d'être transférées vers un puissant contrôleur de mouvement, situé côté maître du réseau (voir la partie inférieure de la Figure 5). Les données échangées sur le réseau temps réel sont les suivantes : une référence de tension (v^*) pour le contrôleur moteur et les informations de rétroaction (i , ω , θ) pour le contrôleur de mouvement. Cette topologie de contrôle, dont le fonctionnement repose sur de puissants automates programmables multicœurs et des réseaux temps réel, offre certains avantages. Premièrement, son architecture est hautement évolutive. Les axes peuvent également être facilement ajoutés/supprimés sans se soucier de la puissance de traitement du contrôleur moteur. Deuxièmement, il est possible d'atteindre un niveau de précision accru, dans la mesure où la planification des trajectoires et le contrôle des mouvements sont centralisés en un seul endroit.

La nouvelle topologie de contrôle présente également des inconvénients. Avec la suppression des algorithmes de commande du contrôleur moteur, la synchronisation étroite entre l'exécution du code et les entrées/sorties est perdue. Plus la bande passante d'une boucle de contrôle est importante, plus la perte de synchronisation des entrées/sorties devient problématique. La boucle couple/courant est particulièrement sensible à la synchronisation.

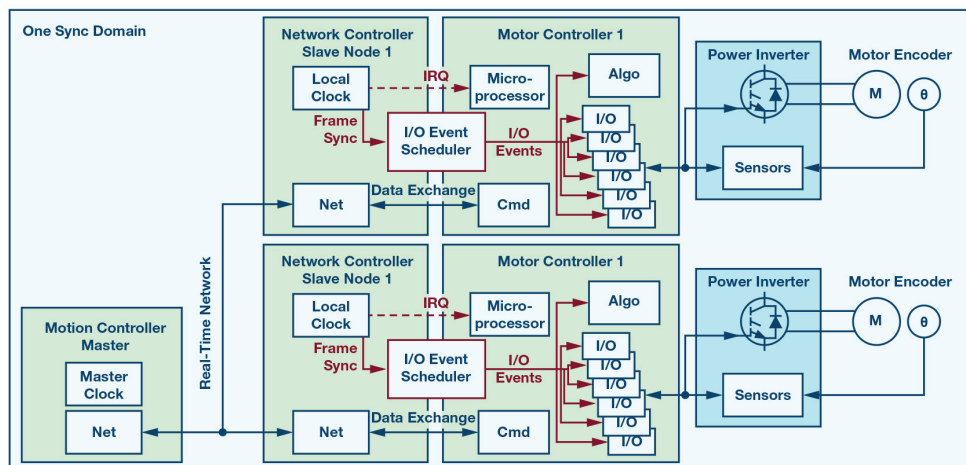


Figure 6. Un programmeur d'entrées/sorties associe les domaines de synchronisation.

Proposition de solution

Le fait de déplacer les boucles de contrôle les plus rapides vers le contrôleur de mouvements crée une demande de synchronisation entre le maître du réseau et les bornes du moteur.

Le principe de base consiste à synchroniser les entrées/sorties de tous les axes sur le réseau pour que tous les organes puissent fonctionner dans un unique domaine de synchronisation. La Figure 6 montre un programmeur d'événements en entrée/sortie qui est placé entre le contrôleur de réseau et le contrôleur moteur. Sa fonction principale est de générer des impulsions de synchronisation/réinitialisation à l'attention de divers périphériques afin de les synchroniser avec le trafic réseau. Le programmeur d'événements en entrée/sortie prend le signal de synchronisation de trames qui est dérivé de l'horloge locale du contrôleur de réseau et génère des déclencheurs matériels pour toutes les entrées/sorties qui doivent rester synchronisées sur le réseau.

Chaque entrée/sortie possède ses propres exigences de timing/réinitialisation, ce qui signifie que le programmeur d'événements en entrée/sortie doit fournir des déclencheurs adaptés à chaque entrée/sortie. Bien que les conditions de déclenchement varient, un principe général s'applique à tous les déclencheurs. Tout d'abord, tous les déclencheurs doivent être référencés par rapport à la synchronisation des trames. Deuxièmement, un retard/décalage est associé à chaque déclencheur. Ce retard est lié au délai inhérent à l'entrée/sortie — par exemple, le temps de conversion d'un convertisseur analogique/numérique ou le temps de propagation de groupe d'un filtre sinc. Troisièmement, le temps de réponse de l'entrée/sortie — par exemple, le transfert des données en provenance d'un convertisseur A/N. Le programmeur d'événements en entrée/sortie connaît les exigences de synchronisation de chaque entrée/sortie et ajuste en permanence les impulsions de déclenchement/réinitialisation par rapport à l'horloge locale. Le principe de la génération de chaque impulsion de sortie du programmeur d'événements en entrée/sortie est représenté à la Figure 7.

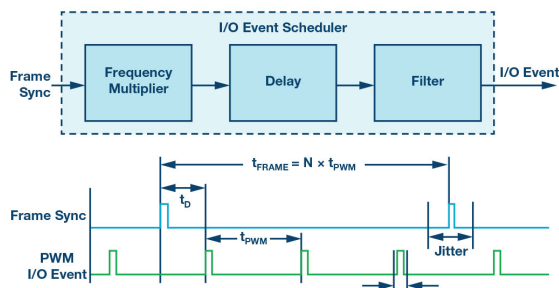


Figure 7. Le programmeur d'événements en entrée/sortie génère les impulsions de déclenchement.

Dans la plupart des systèmes de contrôle de mouvements en réseau, la fréquence des trames, et donc la fréquence de synchronisation des trames, est inférieure ou égale à la vitesse d'actualisation MLI du contrôleur moteur, ce qui signifie que le programmeur d'événements en entrée/sortie doit fournir une, voire plusieurs impulsions de déclenchement par période de trames. Par exemple, si la fréquence de trames est de 10 kHz et la fréquence MLI également de 10 kHz, le programmeur d'événements en entrée/sortie doit fournir 1 impulsion de déclenchement par trame réseau. De même, si la fréquence de trames est de 1 kHz et la fréquence MLI de 10 kHz, il devra envoyer 10 impulsions de déclenchement par trame réseau. Ceci correspond au multiplicateur de fréquences de la Figure 7. Un retard « tD » est appliqué à chaque impulsion de synchronisation pour compenser le retard inhérent à chaque entrée/sortie. Le dernier élément du programmeur d'événements en entrée/sortie est une fonction de filtrage intelligente. Chaque réseau est sujet à une certaine gigue sur le trafic. Ce filtre réduit la gigue sur les impulsions de déclenchement tout en veillant à limiter la vitesse à laquelle varie la fréquence de synchronisation des trames.

La moitié inférieure de la Figure 7 représente un diagramme de synchronisation MLI. Il est à noter que la fréquence de synchronisation de trames est un multiple de la fréquence MLI et que la gigue du signal de déclenchement en entrée/sortie est réduite.

Mise en œuvre

La Figure 8 présente un exemple du schéma de synchronisation proposé, et qui a été mis en œuvre et testé dans un système de contrôle de mouvements en réseau. Le maître du réseau est un automate programmable Beckhoff CX2020 connecté à un PC pour le développement et le déploiement du programme de l'automate. Le protocole réseau temps réel (flèches rouges) est EtherCAT.

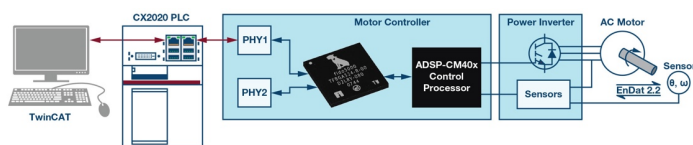


Figure 8. Mise en place d'un schéma de synchronisation.

Les principaux organes du contrôleur moteur sont le [commutateur Ethernet fido5200](#) et le contrôleur ADSP-CM408, tous deux développés par Analog Devices. Ensemble, ils forment un jeu de circuits hautement intégré pour commande moteur connectée en réseau.

Le commutateur Ethernet temps réel multiprotocoles (REM) fido5200 compte deux ports Ethernet et constitue une interface flexible entre un processeur hôte et la couche physique Ethernet industriel. Il est doté d'une unité de commande d'horloge TCU (*Timer Control Unit*) configurable qui permet d'implémenter

des schémas de synchronisation avancés pour les différents protocoles Ethernet industriel. Des fonctions supplémentaires telles que la capture d'entrées et la sortie de signaux en ondes carrées peuvent également être mises en œuvre au moyen de broches de timing dédiées. L'entrée/sortie de l'horloge est maintenue en phase avec l'heure locale synchronisée et, par conséquent, avec le trafic réseau, ce qui permet non seulement de synchroniser les entrées/sorties d'un unique nœud esclave, mais également les nœuds esclaves répartis dans l'ensemble du réseau.

Le commutateur REM possède deux ports Ethernet et se connecte par conséquent à deux Phy (PHY1 et PHY2). Cette topologie supporte à la fois les réseaux en anneau et en ligne. Cependant, dans le montage expérimental, un seul nœud esclave est utilisé à titre d'illustration et un seul port Ethernet est actif.

Le commutateur REM communique avec un processeur hôte par l'intermédiaire d'un bus mémoire parallèle, ce qui garantit un débit élevé et une faible latence.

Le processeur hôte utilisé pour constituer le contrôleur moteur est l'ADSP-CM408, un processeur pour applications spécifiques basé sur un cœur ARM® Cortex®-M4F conçu pour implémenter des fonctions applicatives et de contrôle. Le processeur contient des périphériques qui prennent en charge les applications de commandes industrielles telles que les horloges de commande des onduleurs MLI, l'échantillonnage A/N et les interfaces avec l'encodeur de position. Une unité de routage flexible des déclenchements TRU (*Trigger Routing Unit*) est utilisée pour maintenir la synchronisation entre tous les périphériques et le réseau. Cette unité redirige les déclenchements générés par l'unité TCU du commutateur fido5200 vers tous les périphériques du processeur ADSP-CM408 dont les timings sont critiques : le modulateur à largeur d'impulsion, le filtre Sinc pour la mesure du courant de phase, le convertisseur analogique/numérique et l'interface de l'encodeur absolu. Les principes de la synchronisation des entrées/sorties sont illustrés à la Figure 9.

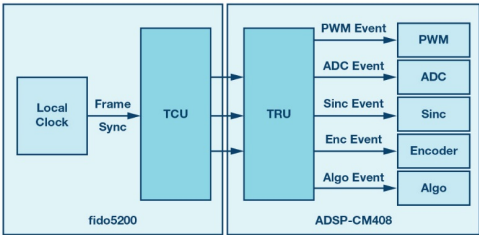


Figure 9. Génération d'événements de synchronisation en entrées/sorties.

La Figure 9 montre que le programmeur d'événements en entrée/sortie est réalisé en utilisant l'unité TCU du commutateur REM et l'unité TRU du processeur de contrôle moteur. En d'autres termes, la fonction est implémentée en utilisant deux circuits intégrés.

La rétroaction du contrôleur moteur ~~est~~ le courant de phase et la position du rotor d'un servomoteur triphasé. Le courant de phase est mesuré à l'aide de convertisseurs analogique/numérique sigma-delta ($\Sigma\Delta$) et la position du rotor avec un encodeur absolu EnDat. Le convertisseur A/N $\Sigma\Delta$ et le codeur sont directement interfacés avec le processeur ADSP-CM408 sans utiliser de FPGA, ni de CPLD externe.

Mis en forme : Non Surlignage

La fréquence de découpage MLI est de 10 kHz et l'algorithme de commande est exécuté une fois par période MLI. Comme nous l'avons vu dans cet article, l'unité TCU envoie des impulsions de synchronisation au processeur ADSP-CM408 une fois par période MLI.

Résultats des essais

Une photo du montage d'essai est représentée en Figure 10. Pour illustrer la synchronisation du système, l'automate PLC a été configuré pour exécuter un programme dans un délai de 200 μ s. La durée d'exécution des tâches détermine également la fréquence de trames sur le réseau EtherCAT. Le

contrôleur moteur fonctionne avec une période d'actualisation de commande et de MLI de 100 μ s (10 kHz), de sorte que les impulsions de synchronisation doivent se produire à cette fréquence. Le résultat est illustré en Figure 11.

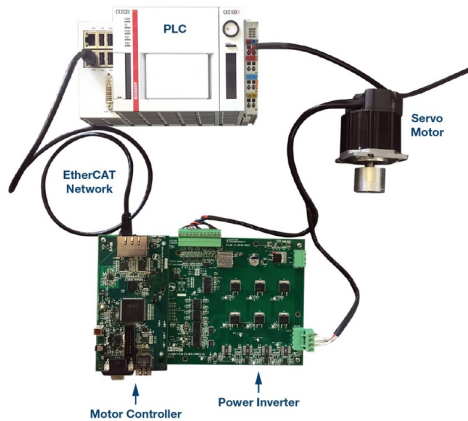


Figure 10. Implémentation d'un montage de synchronisation

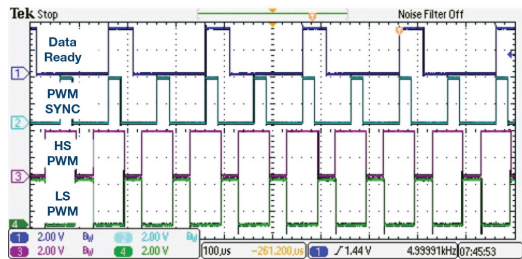


Figure 11. Génération d'événements de synchronisation en entrées/sorties.

Le signal **Data Ready** indique à quel moment le commutateur temps réel REM a mis les données réseau à la disposition de l'application de contrôle moteur. Le signal est validé toutes les 200 μ s, ce qui correspond à la fréquence de trames du protocole EtherCAT. Le signal de synchronisation du MLI est également généré par le commutateur REM et utilisé pour maintenir la synchronisation entre les entrées/sorties du contrôleur moteur par rapport au trafic réseau. La période MLI étant de 100 μ s, le commutateur REM programme deux impulsions de synchronisation MLI par trame EtherCAT. Les deux signaux inférieurs représentés en Figure 11 (HS PWM et LS PWM) sont des signaux MLI high-side et low-side ») pour l'une des phases moteur. Il convient de noter que les signaux MLI sont synchronisés sur le trafic réseau.

Résumé

Le protocole Ethernet temps réel est abondamment utilisé dans les systèmes de contrôle de mouvements, et certains protocoles autorisent une synchronisation temporelle avec une précision inférieure à la microseconde (*sub-1 μ s*). Cependant, la synchronisation concerne uniquement le trafic des données entre le maître du réseau et les organes asservis (esclaves). Les solutions réseau existantes n'incluent pas la synchronisation des entrées/sorties de contrôle de mouvements, ce qui limite les performances de contrôle.

Le schéma de synchronisation proposé dans cet article permet de synchroniser la totalité des éléments depuis le maître du réseau jusqu'aux bornes du moteur. Grâce à une synchronisation nettement optimisée, le système proposé améliore les performances de contrôle de façon significative. Ce schéma assure en outre une synchronisation transparente sur plusieurs axes. D'autres axes peuvent être ajoutés facilement et la synchronisation adaptée individuellement au contrôleur moteur.

La synchronisation est basée sur un programmeur d'événements en entrée/sortie qui est placé entre le contrôleur réseau et le contrôleur moteur. Cet organe peut être programmé à la volée pour minimiser l'effet des variations de gigue et de fréquence.

Le schéma proposé dans l'article a été validé dans un cadre expérimental et les résultats ont été présentés. Si le protocole EtherCAT a été utilisé comme protocole de communications, les concepts abordés s'appliquent à tout protocole Ethernet fonctionnant en temps réel.

ⁱ Jie Ma. « *Multi-DOF Motion Control System Design and Realization Based on EtherCAT*. » 6^e conférence internationale sur l'instrumentation et la mesure, l'informatique, les communications et le contrôle, juillet 2016.