Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

MINISTÈRE
DE L'ÉDUCATION
NATIONALE, DE
L'ENSEIGNEMENT
SUPÉRIEUR ET DE
LA RECHERCHE

EAE SIN 3

SESSION 2017

AGRÉGATION
CONCOURS EXTERNE

Section : SCIENCES INDUSTRIELLES DE L'INGÉNIEUR

Option : SCIENCES INDUSTRIELLES DE L'INGÉNIEUR ET INGÉNIERIE INFORMATIQUE

CONCEPTION PRÉLIMINAIRE D'UN SYSTÈME, D'UN PROCÉDÉ OU D'UNE ORGANISATION

Durée : 6 heures

Calculatrice électronique de poche - y compris calculatrice programmable, alphanumérique ou à écran graphique – à fonctionnement autonome, non imprimante, autorisée conformément à la circulaire n° 99-186 du 16 novembre 1999.

L'usage de tout ouvrage de référence, de tout dictionnaire et de tout autre matériel électronique est rigoureusement interdit.

Dans le cas où un(e) candidat(e) repère ce qui lui semble être une erreur d'énoncé, il (elle) le signale très lisiblement sur sa copie, propose la correction et poursuit l'épreuve en conséquence.

De même, si cela vous conduit à formuler une ou plusieurs hypothèses, il vous est demandé de la (ou les) mentionner explicitement.

NB : *La copie que vous rendrez ne devra, conformément au principe d'anonymat, comporter aucun signe distinctif, tel que nom, signature, origine, etc. Si le travail qui vous est demandé comporte notamment la rédaction d'un projet ou d'une note, vous devrez impérativement vous abstenir de signer ou de l'identifier.*

INFORMATION AUX CANDIDATS

Vous trouverez ci-après les codes nécessaires vous permettant de compléter les rubriques figurant en en-tête de votre copie

Ces codes doivent être reportés sur chacune des copies que vous remettrez.

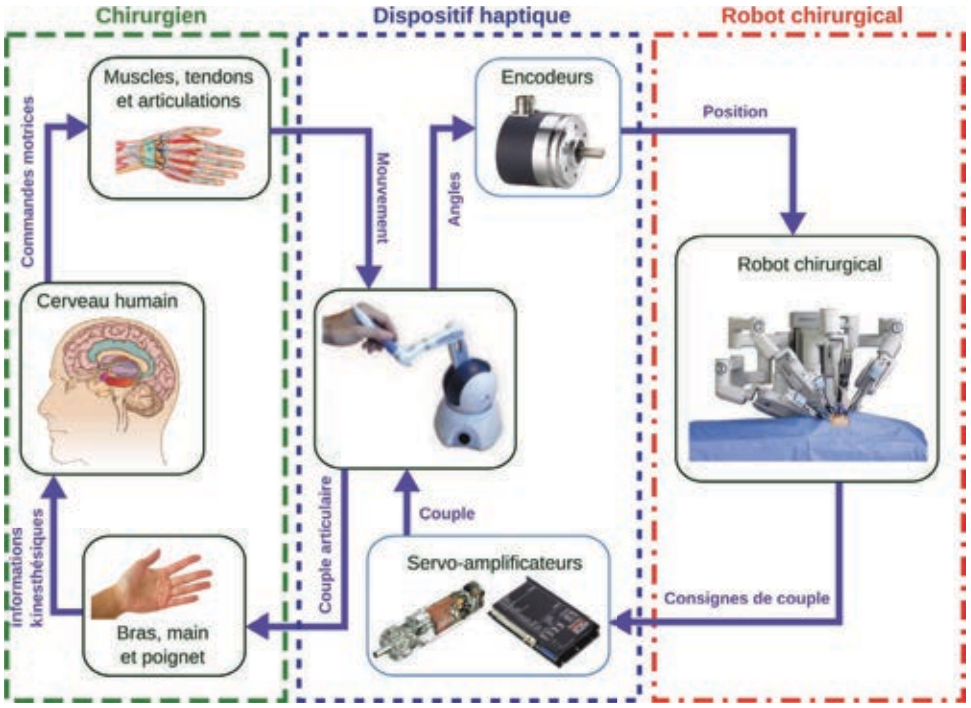
Concours	Section/option	Epreuve	Matière
EAE	1417A	103	1268

Conception d'un banc de test de moteur piézoélectrique

Les systèmes d'assistance opératoire robotisés utilisés en télé-chirurgie

L'idée de créer des robots manipulateurs dans le domaine chirurgical existe depuis longtemps, mais ce n'est que depuis une dizaine d'années que l'on voit arriver dans les blocs opératoires, de véritables systèmes d'assistance opératoire robotisés, donnant naissance à ce que l'on appelle communément la ***télé-chirurgie à interface robotique***.

Ici, le chirurgien pilote depuis sa console un robot constitué de bras articulés à l'extrémité desquels sont positionnés les outils permettant de réaliser l'acte chirurgical.
Les différentes commandes sont transmises au robot par le praticien au moyen de joysticks, et le robot étant par ailleurs équipé de caméras stéréoscopiques, le chirurgien dispose en retour, depuis son poste d'intervention, d'une vision extrêmement fine de l'opération qu'il est en train de réaliser.
Mais quelles que soient la qualité des images reçues et la précision mécanique des bras manipulateurs, il est une dimension sensorielle extrêmement importante qui guide grandement le chirurgien dans son acte opératoire, ***le toucher***, que le système de télé-chirurgie doit impérativement être en mesure de reconstituer artificiellement.
C'est précisément le rôle, au sein du système de télé-chirurgie, de ***l'interface haptique***.



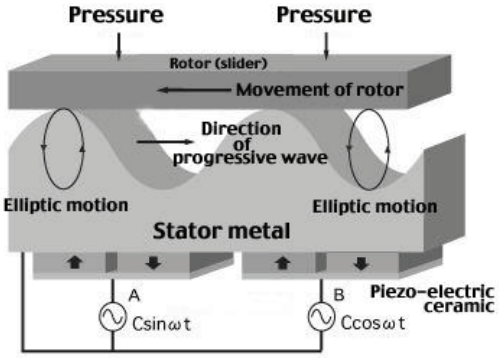
Tournez la page S.V.P.

B

C'est ainsi que les outils positionnés en extrémité des bras de robot sont munis de capteurs de force qui transmettent un signal électrique de commande à un moteur fixé dans l'embase du joystick. Ce dernier produit alors un effet vibratoire de retour d'effort dans la manette du joystick d'intensité proportionnelle à l'effort exercé sur les capteurs, permettant au chirurgien de disposer d'une information sensorielle relative au toucher.

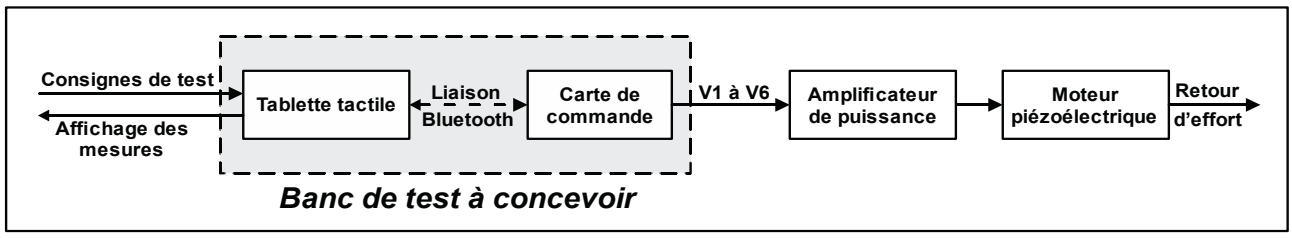
La technologie du moteur de l'interface haptique

Pour réaliser le dispositif de retour d'effort, on utilise un ***moteur piézoélectrique*** constitué de cellules piézoélectriques ayant la capacité à se déformer lorsqu'elles sont soumises à un champ électrique (effet piézoélectrique dit inverse).
Le principe de ce type de moteur consiste à coller des cellules piézoélectriques sous un anneau statorique recouvert d'une plaque métallique (rotor).
On excite alors chaque cellule piézoélectrique par une tension sinusoïdale $\mathbf{V = E\sin(2\pi ft + \varphi)}$, ce qui va provoquer une déformation mécanique de chacune d'entre elles et par conséquent entraîner un mouvement vibratoire de la plaque métallique.



Objet de l'épreuve de conception

Le service R&D d'une entreprise de conception de matériel médical entreprend de concevoir un nouveau moteur de technologie piézoélectrique, destiné à réaliser le dispositif de retour d'effort de l'interface haptique d'un robot de télé-chirurgie.
Ce moteur est équipé de 6 cellules piézoélectriques commandées par 6 tensions sinusoïdales.
L'ingénieur mécanicien en charge du développement de ce moteur a besoin, une fois la réalisation du prototype terminée, d'en caractériser le comportement sur les plans mécanique et énergétique.
Pour ce faire, il sollicite le service électronique de l'entreprise, afin que lui soit proposé un banc de test piloté à l'aide d'une tablette tactile, délivrant 6 tensions sinusoïdales de commande V_1 à V_6 qui, après amplification, exciteront les cellules piézoélectriques du moteur :



Le service électronique délègue alors 2 ingénieurs :
- un ingénieur hardware chargé de concevoir la carte électronique de commande du moteur ;
- un ingénieur software chargé d'en assurer la programmation.

C'est ce rôle d'ingénieur software que nous vous proposons d'endosser durant cette épreuve.

Spécifications techniques du banc de test à concevoir

Remarque préliminaire

Les 6 tensions V_1 à V_6 à élaborer sont des tensions sinusoïdales comportant une composante alternative de la forme $E\sin(2\pi ft + \varphi)$ et une composante continue $-V_0$; elles peuvent par conséquent s'écrire sous la forme $V = E\sin(2\pi ft + \varphi) - V_0$ où E constitue l'amplitude de la composante alternative de la tension V.

Afin de ne pas alourdir inconsidérément les notations, nous nous permettrons dans ce qui suit l'abus de langage consistant à désigner le terme E comme étant l'amplitude de la tension V (alors qu'il s'agit en toute rigueur de l'amplitude de la composante alternative de la tension V).

On précise ci-dessous les caractéristiques d'amplitude, de fréquence et de phase des 6 tensions V_1 à V_6 à élaborer.

$$\mathbf{V_i = E_{1\text{-}4}\sin\left(2\pi ft + \varphi_i\right) - V_{0d}} \text{ pour i = 1, 2, 3, 4 ;}$$

$$\mathbf{V_j = E_{5\text{-}6}\sin\left(2\pi ft + \varphi_j\right) - V_{0d}} \text{ pour j = 5, 6 ;}$$

avec :

- $\mathbf{E_{1\text{-}4}}$: amplitude identique pour chacune des 4 tensions V_1 à V_4 ;
- $\mathbf{E_{5\text{-}6}}$: amplitude identique pour chacune des 2 tensions V_5 et V_6 ;
- **f** : fréquence identique pour chacune des 6 tensions V_1 à V_6 ;
- $\boldsymbol{\varphi_i}$ et $\boldsymbol{\varphi_j}$: phases à l'origine des tensions V_i et V_j ;
- $\mathbf{V_{0d}}$: tension continue de valeur 4V, correspondant à une composante continue identique pour chacune des 6 tensions V_1 à V_6 et égale à -4V ;

Par ailleurs, les consignes $\mathbf{E_{1\text{-}4}}$, $\mathbf{E_{5\text{-}6}}$, **f** et $\boldsymbol{\varphi}$ devront être transmises à la carte par liaison Bluetooth depuis une tablette tactile fonctionnant sous système d'exploitation Android.

Enfin, l'application créée sur tablette tactile sera équipée d'un bouton permettant à l'ingénieur en charge du test du moteur, de mesurer et d'afficher à la demande, les grandeurs $\mathbf{E_{1\text{-}4}}$, $\mathbf{E_{5\text{-}6}}$, **f** et $\boldsymbol{\varphi}$ réellement transmises par la carte de commande à l'amplificateur de puissance.

Un extrait du diagramme des exigences est donné ci-dessous :

1	**Générer des tensions sinusoïdales réglables**	Spécifications
1.1	Réglage de la fréquence f	Entre 10Hz et 1kHz, par pas de 1Hz
1.2	Réglage des 6 phases φ_1 à φ_6	Entre 0° et 360°, par pas de 45°
1.3	Réglage de l'amplitude $E_{1\text{-}4}$ commune aux tensions V_1 à V_4	Entre 3V et 6V, par pas de 50mV
1.4	Réglage de l'amplitude $E_{5\text{-}6}$ commune aux tensions V_5 et V_6	Entre 3V et 6V, par pas de 50mV
2	**Permettre à l'opérateur de modifier les consignes**	
2.1	Interface de réglage	Sur une tablette Android
2.2	Connexion à la carte	Par Bluetooth
3	**Permettre à l'opérateur de vérifier les signaux de sortie de la carte**	
3.1	Réaliser une mesure	Asynchrone : « à la demande »
3.2	Afficher la fréquence, l'amplitude et la phase de chacune des six tensions V_1 à V_6	Affichage dans des zones de texte non modifiables

Le sujet comporte trois parties A, B et C :

- Partie A : élaboration de tensions sinusoïdales d'amplitude E, de fréquence f et de phase φ programmables (chaîne directe)
- Partie B : mesure et affichage des valeurs de E, f et φ des tensions sinusoïdales délivrées par la carte de commande (chaîne de mesure)
- Partie C : synthèse

Contraintes logicielles et matérielles : la programmation de la carte s'effectue à l'aide d'un microcontrôleur LPC4088 implanté sur le module de prototypage rapide « LPC4088 QuickStart Board », dont les principales caractéristiques sont données en DT4, et l'on trouvera en DT8 le modèle de programmation en langage C++ de ce microcontrôleur tel que défini par l'environnement de développement logiciel ***mbed***.
On trouvera par ailleurs en DT9 :

- la structure du programme *main*,
- la définition de l'objet *controleur* assurant la gestion de la carte,
- le codage de certaines méthodes.

A Étude et programmation du dispositif d'élaboration des 6 tensions sinusoïdales V_1 à V_6

L'objectif de la partie A consiste à implémenter les méthodes C++ conduisant à élaborer les 6 tensions sinusoïdales V_1 à V_6 conformément aux consignes entrées par l'opérateur sur tablette et aux spécifications du cahier des charges.

Le DT1 présente l'architecture matérielle de la carte de commande et les signaux échangés.
Le DT2 décrit les structures électroniques permettant d'élaborer la tension sinusoïdale V_1 (structures identiques pour les tensions V_2 à V_6).
Le DT3 reprend les notations des signaux présents sur les DT1 et DT2, et précise ainsi l'architecture globale de la carte.
Le DR6 spécifie l'architecture logicielle par un diagramme de classes, et en particulier les méthodes qui seront implémentées dans la suite du sujet.

A1 Élaboration des tensions sinusoïdales V_1 à V_6 et programmation de la fréquence f et de la phase φ

On s'attache, dans cette partie A1, à vérifier que la carte permet d'élaborer les 6 tensions sinusoïdales V_1 à V_6 de caractéristiques de fréquence et de phase conformes aux spécifications techniques attendues.

Le composant permettant d'élaborer une tension sinusoïdale est un circuit intégré DDS (Direct Digital Synthesizer) ou Synthétiseur Numérique Direct, de référence AD9850, dont un extrait de la documentation constructeur est donné en DT6.
Pour réaliser cette carte de commande, et afin de simplifier le tracé du circuit imprimé, l'ingénieur hardware a décidé d'implanter non pas 6 circuits intégrés AD9850 (un pour chaque voie), mais 6 modules de prototypage rapide DDSAD9850 (voir DT5).
Pour ce qui concerne la seule voie 1, le module porte la référence U2 sur le schéma structurel DT2.

La documentation constructeur DT6 indique le mot de commande à transmettre au DDS afin de programmer la fréquence f et la phase φ de sa tension de sortie avec les valeurs souhaitées**.**
Elle précise notamment que ce mot de commande peut être transmis par le µC sous forme série ou parallèle.

Q1 Citer les avantages et les inconvénients d'une commande série et d'une commande parallèle du DDS.

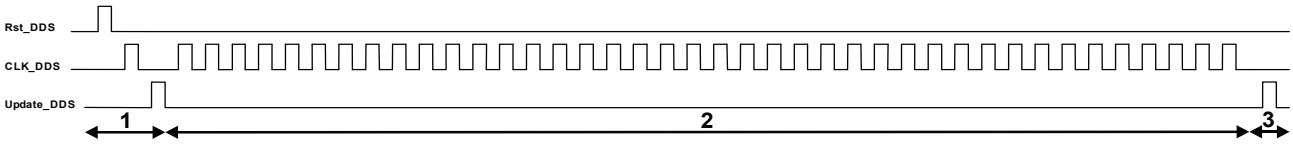
Q2 Préciser la raison pour laquelle, à la lecture du schéma structurel DT2, il apparait clairement que l'ingénieur hardware a choisi de piloter le DDS sous forme sérielle.

La classe *DDS* (DR6 et DT9) permet de définir un objet *dds* disposant de tous les attributs et méthodes utiles à l'initialisation et la commande des six composants DDS.

Q3 Quelle classe est utilisée par la classe DDS ? Quel est le type de la relation entre ces deux classes. Pourquoi cette relation a-t-elle été utilisée, en termes d'extension de fonctionnalités de la classe DDS ?

On précise ci-après quel doit être le cadencement des signaux de commande du DDS en mode série.
La programmation de la fréquence f et de la phase φ de la tension de sortie du DDS s'effectue en 3 phases successives :

1) Phase de configuration du DDS en mode série
2) Phase de transmission série d'un mot de commande de 40 bits
3) Phase de mise à jour de la tension sinusoïdale de sortie du DDS



1. Phase de configuration du DDS en mode série associée à la méthode « config_serie_dds »

La phase de configuration consiste à élaborer successivement 3 impulsions :

– une impulsion de réinitialisation sur la broche RESET du DDS
– une impulsion d'horloge sur la broche W_CLK du DDS
– une impulsion de mise à jour sur la broche FQ_UD du DDS

2. Phase de transmission série d'un mot de commande de 40 bits

Cette phase est décrite figure 12 du DT6. Elle consiste à transmettre sur la broche D7 du DDS, en synchronisme avec un signal d'horloge appliqué sur sa broche W_CLK, un mot binaire de 40 bits dont la valeur détermine la fréquence f et la phase φ de la tension sinusoïdale de sortie.

3. Phase de mise à jour de la tension sinusoïdale de sortie du DDS

Il s'agit, conformément aux chronogrammes de la figure 12, d'élaborer une impulsion sur la broche FQ_UD, qui a alors pour effet de mettre à jour la tension sinusoïdale de sortie avec les paramètres de fréquence f et de phase φ qui viennent d'être transmis.

Parmi les 40 bits du mot de commande transmis au DDS, 32 bits permettent de déterminer la fréquence f de la tension de sortie, que nous appellerons « Tuning Word » **TW**.

Tournez la page S.V.P.

On rappelle que le DDS intégré sur le module de prototypage rapide DDSAD9850 qui équipe la carte de commande est cadencé par un oscillateur à quartz délivrant un signal d'horloge de fréquence 125 MHz, notée CLKIN dans la documentation du DDS.

Q4 Calculer et exprimer en décimal le mot de 32 bits TW correspondant à une fréquence de 1 kHz.

Parmi les 40 bits du mot de commande transmis au DDS, 5 bits permettent de déterminer la phase φ de la tension de sortie, que nous appellerons « Phase Word » **PW**.
La phase φ en ° s'exprime sous la forme $\boldsymbol{\varphi = b_0.11{,}25 + b_1.22{,}5 + b_2.45 + b_3.90 + b_4.180}$ où b_0 (LSB) à b_4 (MSB) constituent les 5 bits du mot de commande de phase **PW**.

Q5 Calculer et exprimer en décimal le mot de 5 bits PW correspondant à une phase de 225°.

Q6 Compléter les chronogrammes du document réponse DR1 correspondant à la programmation d'une tension sinusoïdale V_{5_1} de fréquence **f = 1kHz** et de phase $\boldsymbol{\varphi_5}$ **= 225°** (les bits W32 à W34 seront positionnés à 0).

Q7 Proposer le codage de la méthode de prototype **void DDS::calcul_TW(void)** calculant l'attribut privé de la classe DDS (DR6) TW (Tuning Word) à partir de l'attribut de fréquence désirée *freq* (diagramme de classe) exprimée en Hz. Si nécessaire, la bibliothèque *math.h* peut être utilisée.

Q8 Proposer le codage de la méthode de prototype **void DDS::calcul_PW(void)** calculant l'attribut PW (tableau de six entiers « Phase Words ») à partir de l'attribut de phases désirées *tab_phi* (tableau de six phases) exprimées en degrés.

On rappelle qu'une partie du code implanté dans le microcontrôleur est décrit dans le document technique DT9.

Q9 Identifier les instructions C++ permettant d'instancier la classe DDS. Combien d'objets de la classe DigitalOut seront instanciés lors de l'instanciation de la classe DDS ? Expliquer la présence des quatre arguments dans la méthode « init_DDS ».

Q10 Quelle est l'utilité, pour la transmission du mot binaire de 40 bits par communication série, de la méthode **void DDS::pulse_CLK_DDS(void)** dont le codage est donné en DT9 ?

On souhaite maintenant coder la méthode permettant aux 6 DDS d'élaborer les tensions V_{1_1} à V_{6_1}. Cette méthode permet d'envoyer simultanément aux 6 DDS, la valeur de fréquence (qui leur est commune à tous), ainsi que la valeur de phase (qui leur est spécifique). Il est précisé que le composant DDS n'impose pas de fréquence particulière pour la transmission des données.

Q11 Proposer le codage de la méthode de prototype **void DDS::envoi_f_phi(void)** qui appelle dans un premier temps les méthodes calcul_TW et calcul_PW permettant de calculer TW et PW, puis qui commande simultanément les six composants DDS en leur transmettant la trame série bit à bit (phase 2) et en mettant à jour leurs sorties simultanément (phase 3).

On conclut maintenant cette partie A1 en vérifiant que la solution envisagée est conforme, pour ce qui concerne les paramètres de fréquence et de phase, aux spécifications attendues.

Q12 Indiquer en Hz, à partir du DT6, le pas de réglage de la fréquence des tensions sinusoïdales V_{1_1} à V_{6_1}, et montrer que cette valeur est compatible avec les spécifications techniques du banc de test.

Q13 Indiquer, à partir du DT6, entre quelles valeurs minimale et maximale peut varier la fréquence des tensions V_{1_1} à V_{6_1}, et montrer que ces valeurs sont compatibles avec les spécifications techniques du banc de test.

A2 Contrôle de l'amplitude des tensions sinusoïdales V_1 à V_6

L'objectif de cette partie A2 consiste à vérifier la capacité de la carte de commande à élaborer des tensions sinusoïdales V_1 à V_6 présentant des valeurs d'amplitude E conformes aux spécifications techniques attendues.

Comme vu précédemment, la fréquence et la phase de la tension sinusoïdale de sortie des DDS est programmable, mais pas l'amplitude. Or, les actionneurs piézoélectriques nécessitent un contrôle de l'amplitude de leur tension de commande, et c'est pour cette raison qu'une structure électronique mixte (analogique et numérique) a été implantée en sortie de DDS (voir DT1 et DT2).

On précise ci-après le traitement de signal opéré par la carte de commande permettant, à partir de la tension V_{1_1}, d'élaborer la tension V_1 (ce traitement de signal depuis V_{k_1} jusqu'à V_k est le même pour chacune des 6 voies de la carte).

Caractéristiques de la tension V_{1_1}

La tension sinusoïdale V_{1_1} délivrée par le DDS possède les caractéristiques suivantes.

$$\mathbf{V_{1\_1} = E \sin(2\pi ft + \varphi) + V_{moy}}$$ avec :

- **f** et **φ** programmables selon le mode opératoire étudié au paragraphe A1 ;
- Amplitude **E** fixe et égale à 0,5V ;
- Composante continue **Vmoy** fixe et égale à 0,5V.

Caractéristiques de la tension V_{1_2}

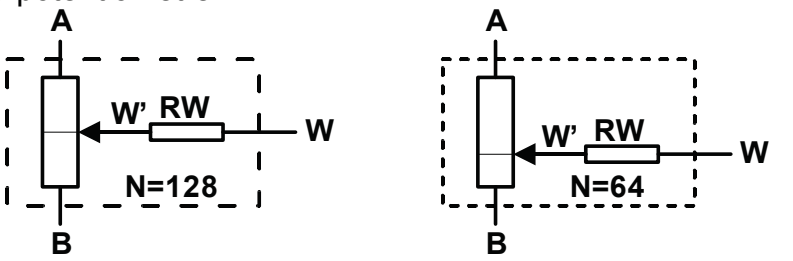
La résistance ajustable AJ1 est ajustée de telle façon que l'amplitude de V_{1_2} soit égale à 1,28V, soit une valeur crête à crête de 2,56V. L'amplification K1 est donc égale à 1,28/0,5 soit 2,56, et la composante continue de V_{1_2} est désormais égale à 0,5x2,56, soit 1,28V.

Caractéristiques de la tension V_{1_3}

La tension V_{1_3} est élaborée par le potentiomètre numérique AD5161 (noté U4 sur le DT2) dont un extrait de la documentation est donné en DT7.

Il s'agit d'un potentiomètre dont la position du curseur n'est pas ajustée manuellement, mais numériquement. La résolution de ce potentiomètre numérique est de 8 bits, ce qui signifie que son curseur W' peut prendre 256 positions distinctes et équidistantes entre les extrémités A et B.

On note N, le nombre compris entre 0 et 255 transmis par le microcontrôleur au potentiomètre, et on propose ci-après 2 exemples permettant d'illustrer le principe de fonctionnement du potentiomètre.



$R_{BW'}=R_{AW'}=R_{AB}/2$ $R_{BW'}=R_{AB}/4$ et $R_{AW'}=3R_{AB}/4$

R_W constitue la résistance équivalente présente entre le curseur W' et la broche W permettant d'accéder à ce dernier (W signifie « wiper » pour curseur en anglais).
On précise que dans notre application, l'intensité du courant circulant dans la résistance du curseur R_W est nulle, et que par conséquent la tension V_{RW} à ses bornes est nulle également.

Q14 Montrer, après avoir remarqué que la broche B du potentiomètre dans notre application est reliée à la masse, que l'amplitude de V_{1_3} est égale à **0,005.N**. Exprimer K2 en fonction de N.

Caractéristiques de la tension V_{1_4}
L'amplitude de la tension V_{1_4} est inchangée par rapport à celle de V_{1_3}, par contre sa composante continue est désormais nulle.

Caractéristiques de la tension V_{1_5}
La résistance ajustable AJ2 est ajustée de telle façon que l'amplification K3 soit égale à 10.

Caractéristiques de la tension V_{0d}
V_{0d} est une tension continue d'amplitude 4V.

Caractéristiques de la tension V_1
La tension V_1 est élaborée à partir d'un montage soustracteur tel que $\mathbf{V_1 = V_{1\_5} - V_{0d}}$.

Q15 Compléter les relevés d'oscillogrammes du document réponse DR2 correspondant au cas particulier où N=90.

Q16 En déduire entre quelles valeurs Nmin et Nmax le nombre N transmis par le µC au potentiomètre doit varier pour que l'amplitude de V_1 varie entre 3V et 6V conformément aux spécifications techniques attendues.

Q17 Calculer le pas de réglage d'amplitude de la tension V_1 et vérifier sa conformité vis-à-vis des spécifications techniques attendues.

On s'intéresse désormais à la façon dont le µC envoie le nombre N au potentiomètre.
On précise pour ce qui suit, que le nombre N transmis aux 4 premiers potentiomètres et permettant de fixer l'amplitude $E_{1\text{-}4}$ commune aux tensions V_1 à V_4, sera noté $N_{1\text{-}4}$, alors que le nombre N transmis aux 2 derniers potentiomètres sera noté $N_{5\text{-}6}$.

Le constructeur du potentiomètre numérique AD5161 indique, dans sa documentation technique (DT7), qu'il peut être piloté par une interface SPI ou I2C.

Q18 Préciser quel élément de câblage du schéma structurel DT2 permet d'identifier clairement que l'ingénieur hardware a choisi de piloter le DDS à l'aide d'une interface SPI.

Q19 Proposer, en prenant en compte les chronogrammes de la figure 37 du document technique DT7 d'une part, ainsi que le modèle de programmation de l'interface SPI (DT8) d'autre part, le codage de la méthode de prototype **void Potentiometre::init_SPI(void)** permettant de configurer l'interface SPI du µC pour qu'elle soit en mesure de piloter les 6 potentiomètres numériques avec une fréquence d'horloge de 25 MHz (on précise à titre indicatif, que cette fonction comporte 2 instructions, partiellement proposées dans le document DT9).

Q20 Proposer le codage de la fonction de prototype **void Potentiometre::calcul_N(void)** permettant de calculer la valeur de l'attribut *AW* à partir de l'attribut *amplitude* définissant l'amplitude de V_k désirée et exprimée en Volt.

Q21 Proposer le codage de la méthode de prototype **void Potentiometre::envoi_E(void)**, permettant, à partir de l'attribut d'amplitude désirée, de calculer le mot binaire *N* puis de commander le potentiomètre via l'interface SPI du microcontrôleur.

Q22 Compléter les chronogrammes du document réponse DR3 correspondant au cas où le nombre *N* transmis par l'interface SPI du µC au potentiomètre numérique est égal à 90.

Q23 Sachant que la fréquence maximale du signal d'horloge SCK_POT admissible par le potentiomètre numérique est de 25MHz lorsqu'il est piloté par une interface SPI, calculer dans ces conditions la durée minimale mise par le µC pour transmettre le nombre N.

On envisage maintenant le cas d'une commande des potentiomètres par le bus I2C du µC.

Q24 Indiquer les 2 adresses I2C permettant d'accéder au potentiomètre numérique, et expliquer par quel procédé, avec une seule interface I2C, il aurait été possible à un instant donné de transmettre le nombre $N_{1\text{-}4}$ aux 4 premiers potentiomètres de la carte, puis le nombre $N_{5\text{-}6}$ aux deux derniers potentiomètres.

Q25 Proposer alors (document DR4) le câblage des broches **DIS**, **CLK/SCL** et **SDI/SDA** de chacun des six potentiomètres programmables si une communication série I2C avait été envisagée avec le microcontrôleur.

Q26 Compléter les chronogrammes du document réponse DR5 correspondant au cas où le nombre $N_{1\text{-}4}$ transmis par l'interface I2C du µC aux 4 premiers potentiomètres numériques est égal à 90 (on positionnera les bits non significatifs à 0).

Q27 Sachant que la fréquence maximale du signal d'horloge SCK_POT admissible par le potentiomètre numérique est de 400 kHz lorsqu'il est piloté par une interface I2C, calculer dans ces conditions la durée minimale mise par le µC pour transmettre le nombre $N_{1\text{-}4}$.

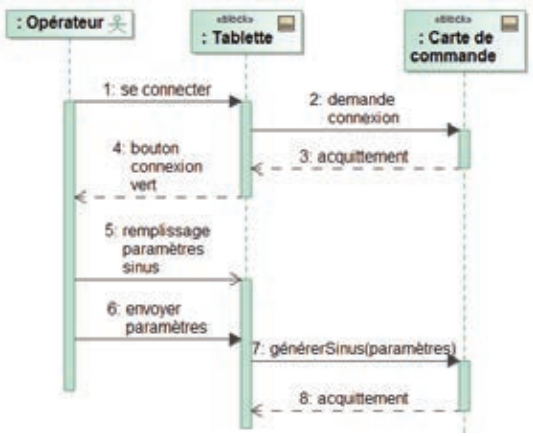
Q28 Justifier alors la raison pour laquelle l'ingénieur hardware a préféré piloter le potentiomètre à l'aide d'une interface SPI.

A3 Prise en compte des consignes de l'IHM

Une interface homme machine (IHM) est réalisée sur une tablette Android. L'interface graphique est donnée ci-contre. Deux boutons permettent de se connecter ou se déconnecter en Bluetooth à la carte de commande, des zones de texte permettent de saisir la fréquence, les amplitudes et les phases des tensions et un bouton « Envoi des données » transmet ces réglages de consigne à la carte. Celle-ci est équipée d'un module Bluetooth positionné sur le connecteur RF de la carte de prototypage (DT4), piloté depuis le microcontrôleur par une liaison RS232 (DT1).

Le diagramme de séquence donné ci-après précise les échanges entre l'opérateur, la tablette et la carte.

La réception des données par le microcontrôleur est réalisée par une méthode **void Interface_Bluetooth::IT_RX(void)** appelée par une interruption générée par l'interface Bluetooth configurée (voir DR6). Cette méthode IT_RX n'est pas abordée dans le sujet et au terme de la réception des données de consigne, l'attribut de la classe Interface_Bluetooth *data_RX* contient la chaine de caractères reçue et l'attribut *RECEPTION_BT* est basculé à VRAI. La méthode *Interface_Bluetooth::put_data_TX(char*)* permet d'envoyer des chaines de caractères.



La chaine de caractère transmise pour les réglages présents sur la figure est de la forme suivante :

!100|6|4|0|45|90|180|225|270?

La chaine commence par « ! », se termine par « ? » et contient les données dans l'ordre de l'interface séparées par « | ». Le traitement d'une chaine de caractères formatée peut s'effectuer à l'aide de la fonction *sscanf* décrite dans le DT8 page 4/4.

Une fois le message reçu et décodé, la carte renvoie par Bluetooth un message d'acquittement à l'interface Android, sous la forme d'une chaine de caractères « Acq ». Si le message n'a pas été correctement décodé, la carte renvoie le message d'erreur « Erreur ».

Q29 Proposer le codage de la méthode de prototype :
void Interface_Bluetooth::acquisition_consignes(DDS *dds, Potentiometre *pot) permettant de traiter la chaine de caractères *data_RX* et de stocker les consignes dans les attributs respectifs *f*, *phi* et *amplitude* des objets *dds* et *pot* passés en argument, et d'acquitter la réception des données à la tablette. À noter que l'argument **pot* fait référence à un tableau de deux éléments.

B Étude et programmation du dispositif de mesure et d'affichage des caractéristiques des 6 tensions sinusoïdales V_1 à V_6

L'objectif de la partie B consiste à compléter l'architecture logicielle afin d'assurer un enregistrement continu des tensions de sortie et de transmettre à l'utilisateur leurs caractéristiques.

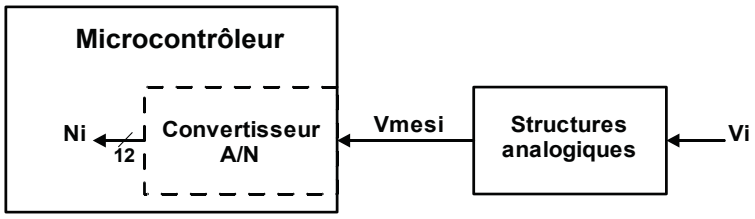
Afin de contrôler le bon fonctionnement de la carte sans avoir à brancher 2 oscilloscopes, une mesure des caractéristiques d'amplitude, de fréquence et de phase de chacune des 6 tensions V_1 à V_6 est réalisée en vue d'être affichées sur la tablette tactile.
L'affichage ne s'effectuera pas en permanence, mais à la demande de l'opérateur par simple appui, sur la tablette tactile, sur un bouton « Acquisition Mesures ».

B1 Mesure des caractéristiques E, f et φ des tensions sinusoïdales V_1 à V_6

Le µC LPC4088 est équipé d'un convertisseur analogique-numérique à 6 entrées multiplexées dont les caractéristiques sont :

- résolution de 12 bits
- excursion de la tension d'entrée comprise entre 0V et 3,3V

L'ingénieur hardware a donc conçu des structures électroniques (voir DT1 à DT3) permettant de modifier les caractéristiques d'amplitude des tensions V_1 à V_6 de façon à les rendre compatibles avec le convertisseur analogique-numérique du µC (V_{mes1} à V_{mes6}) :



Ces structures analogiques associées au convertisseur analogique-numérique intégré au µC permettent ainsi de réaliser un voltmètre numérique.

Sachant que lorsque E=Emin = 3V, V_i varie entre -7V et -1V, et que lorsque E=Emax = 6V, V_i varie entre -10V et +2V, les structures analogiques implantées sur la carte ont été conçues de telle façon que :

- lorsque $V_i = V_{imin}$ = -10V, $V_{mesi} = V_{mesimin}$ = 0,15V,
- lorsque $V_i = V_{imax}$ = +2V, $V_{mesi} = V_{mesimax}$ = 3,15V.

Q30 Exprimer V_{mesi} en fonction de V_i.

Afin d'afficher les caractéristiques d'amplitude en Volts sur l'interface homme-machine, il est nécessaire d'établir la relation entre le résultat de la conversion Ni et l'amplitude E en Volt.

Q31 Exprimer N_i en fonction de V_i, où N_i désigne le résultat de la conversion analogique numérique de la tension V_{mesi}.
Montrer alors que l'amplitude E de la tension V_i (notée $E_{1\text{-}4}$ ou $E_{5\text{-}6}$ selon la valeur de i) peut être déterminée par la relation **$E = 0{,}00322.N_{imax} - 6{,}6$**, où N_{imax} représente la valeur du résultat de conversion analogique numérique obtenue lorsque la tension $V_i(t)$ est maximale.

Tournez la page S.V.P.

L'objectif de la suite de cette partie est de réaliser une classe « Voies_Mesure » et de l'intégrer dans le diagramme de classes.
La méthode d'initialisation de l'instance est la suivante :

```
void Voies_Mesure ::init_Voies_Mesure(PinName *pinADC)
{
   //Déclaration des broches CAN
   for (int i=0; i<6; i++)
   {
      analog_pin[i]=new AnalogIn(pinADC[i]);
   }
   // Initialisation du numéro d'échantillon
   n_ech=0 ;
   // Initialisation du timer pour l'échantillonnage
   tic_echantillonnage.attach_us(this,&Voies_Mesure::mesure_Vk,T_echantillonnage);
}
```

L'acquisition des valeurs instantanées de l'amplitude des tensions $V_1(t)$ à $V_6(t)$ est faite en continu. Pour cela, les résultats de conversion analogique numérique Ni sont mémorisés dans le tableau Vk. Un entier « n_ech », compris entre 0 et 1999, mémorise l'indice de la dernière valeur stockée. Une interruption, déclenchée toutes les 50 µs (valeur de T_echantillonnage, voir méthode init_Voies_Mesure), exécute une méthode void mesure_Vk(void), permettant de réaliser une mesure sur chaque voie et de compléter les 6 lignes du tableau V à la colonne d'indice n_ech. Lorsque la fin du tableau est atteinte, l'enregistrement reprend au début de façon à écraser les échantillons les plus anciens. Le tableau « Vk », de type short, et l'entier « n_ech » sont définis comme des attributs de la classe « Voies_Mesure ». L'API concernant la conversion analogique numérique est décrite dans le document ressource DT8.

Q32 Préciser pour la classe Voie_Mesure les attributs et les prototypes des méthodes envisagées, en définissant leurs rôles.

Q33 Compléter le diagramme de classes du document réponse DR6 en complétant l'architecture logicielle permettant d'ajouter la fonctionnalité de mesure des tensions de sortie.

Q34 Proposer une implémentation de la méthode void mesure_Vk(void).

Le modèle de programmation du convertisseur analogique numérique est donné dans le document ressource DT8.

Q35 À la fréquence maximale d'échantillonnage de 400 kHz, combien d'échantillons par période obtient-on aux fréquences maximale et minimale des tensions ? Pour traiter les mesures sur au moins 10 périodes pour la fréquence la plus basse, avec au moins 20 points par période pour la fréquence la plus élevée, quel espace mémoire faut-il prévoir ? Est-ce compatible avec la capacité mémoire du microcontrôleur ?

Pour éviter un traitement trop fastidieux, le nombre d'échantillons est fixé à 2000 par signaux, à une fréquence de 10 kHz. À partir de ces mesures, les valeurs minimale et maximale sont déterminées, ainsi que les fréquences et les phases relatives par rapport au premier signal.

Les variables et tableaux suivants sont définis parmi les attributs :

```
unsigned short V[6][2000] ;  // tableau contenant les 2000 résultats de conversion analogique numérique de chacune des 6 tensions Vmesi
short n_ech =0 ; // variable contenant l'indice du dernier échantillon mesuré
float **minmax ; // tableau [6][2] contenant les valeurs de Nmin et Nmax de chacune des 6 tensions Vmesi
float *freq ;  // tableau [6] contenant les fréquences de chacune des 6 tensions Vmesi
float *phase ; // tableau [6] contenant les phases mesurées sur les cinq tensions Vmes2 à Vmes6  par rapport à la tension Vmes1  (la première valeur du tableau étant nulle).
```

Q36 Proposer une méthode de prototype **void mesure_minmax(void)** permettant de remplir le tableau *minmax* à partir des mesures contenues dans le tableau *V*.

Q37 Proposer une méthode de prototype **void mesure_freq(void)** permettant de remplir le tableau *freq* à partir des mesures contenues dans le tableau *V*.

B2 Transmission à destination de la tablette tactile des caractéristiques E, f et φ des tensions sinusoïdales V_1 à V_6

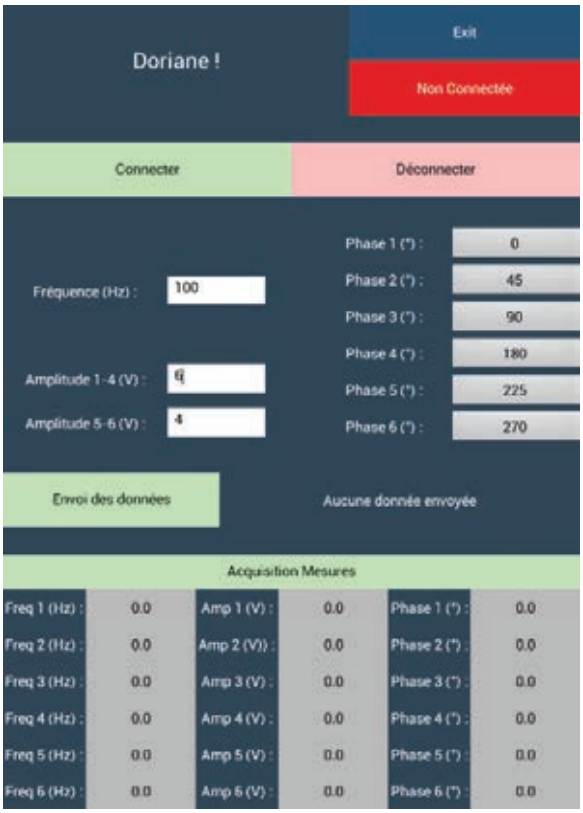
Les mesures sont renvoyées à l'utilisateur « à la demande », c'est-à-dire lorsqu'il appuie sur un bouton « Acquisition Mesure » de l'interface. La tablette transmet la requête à la carte en lui envoyant une chaine sans données, c'est-à-dire la chaine « !? ». À la réception de cette chaine, la carte doit traiter les mesures et renvoyer les fréquences, les amplitudes et les phases des six tensions, sous la forme de six messages de la forme :
!100|5.35|90?
Où dans cet exemple, 100 est la fréquence en Hertz, 5.35 l'amplitude en Volts, et 90 la phase en degrés.

Q38 Proposer une modification de la fonction **void Controleur::run(void)** permettant de prendre en compte la demande de mesures de l'utilisateur (sans modifier les autres fonctionnalités de la carte) et d'envoyer les mesures vers l'IHM par Bluetooth (par la méthode *envoi_mesure()*).

Tournez la page S.V.P.

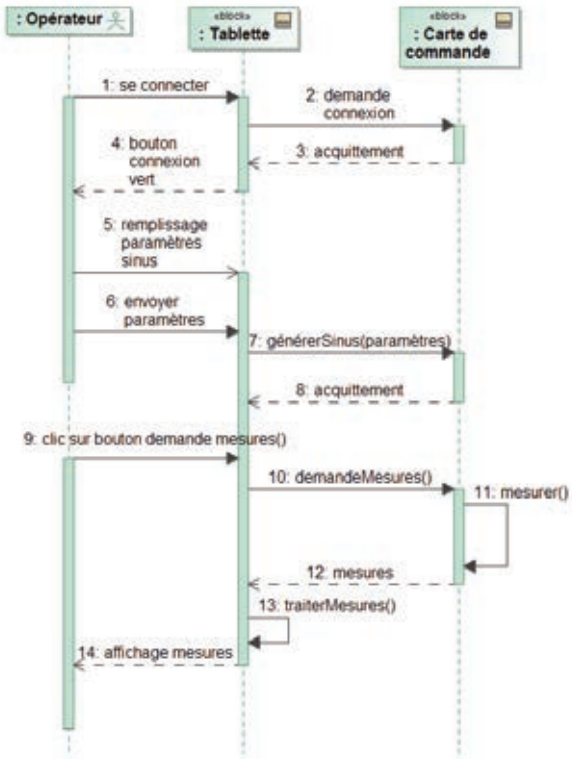
B3 Affichage des caractéristiques E, f et φ des tensions sinusoïdales V_1 à V_6

Dans un souci de rapidité d'implémentation et de flexibilité, l'interface Android est réalisée en langage Python. Les périphériques Android sont rendus accessibles par l'interface SL4A (Script Layer for Android) et l'affichage est pris en charge par le module fullscreenwrapper2. Cette partie ne requiert aucune connaissance spécifique de ces modules.



L'organisation graphique de l'interface graphique ci-contre est définie par un fichier XML dont la structure est donnée sur le document réponse DR7. Le code python de l'application est donné dans le document ressource DT10.

L'objectif de cette partie est de modifier l'application actuelle pour ajouter la fonctionnalité permettant à l'utilisateur de demander une mesure des caractéristiques des tensions, de traiter la réponse envoyée par la carte et d'afficher les mesures. L'interface graphique prendra la forme finale de la figure ci-contre et le diagramme de séquence modélisant le comportement est donné ci-après.



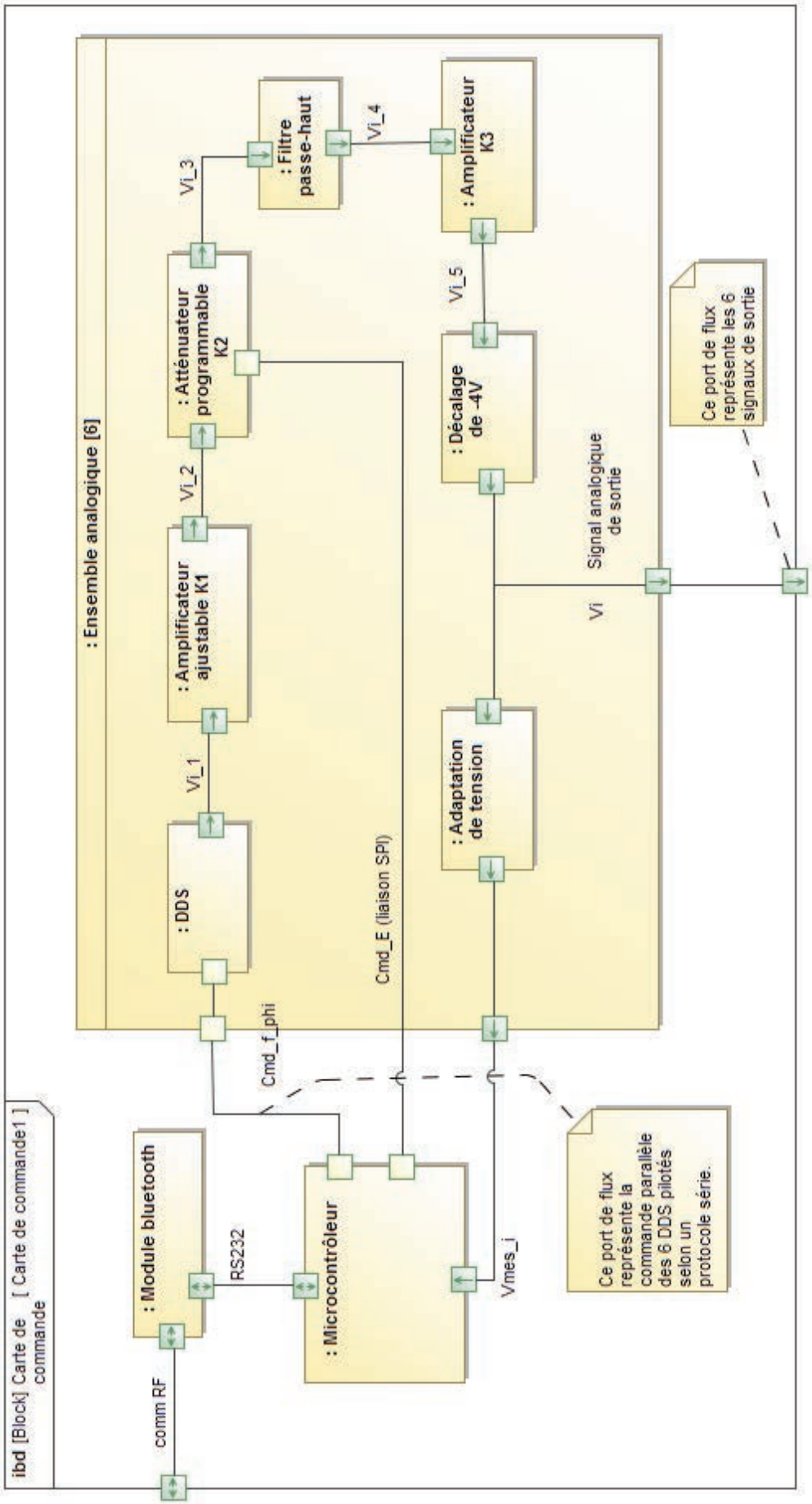
Q39 Compléter la structure du fichier XML sur le document réponse DR7, permettant d'ajouter dans l'interface graphique le bouton « Acquisition Mesures » et le tableau de mesures.

Q40 Quelle ligne faut-il ajouter à la méthode onShow() pour lancer une fonction mesure() lors d'un clic sur le bouton « Acquisition Mesures », dont l'objet est appelé « but_mesure » ?

C Synthèse

Q41 Déterminer, en justifiant vos réponses, si la solution proposée permet de satisfaire aux exigences définies en début de sujet.

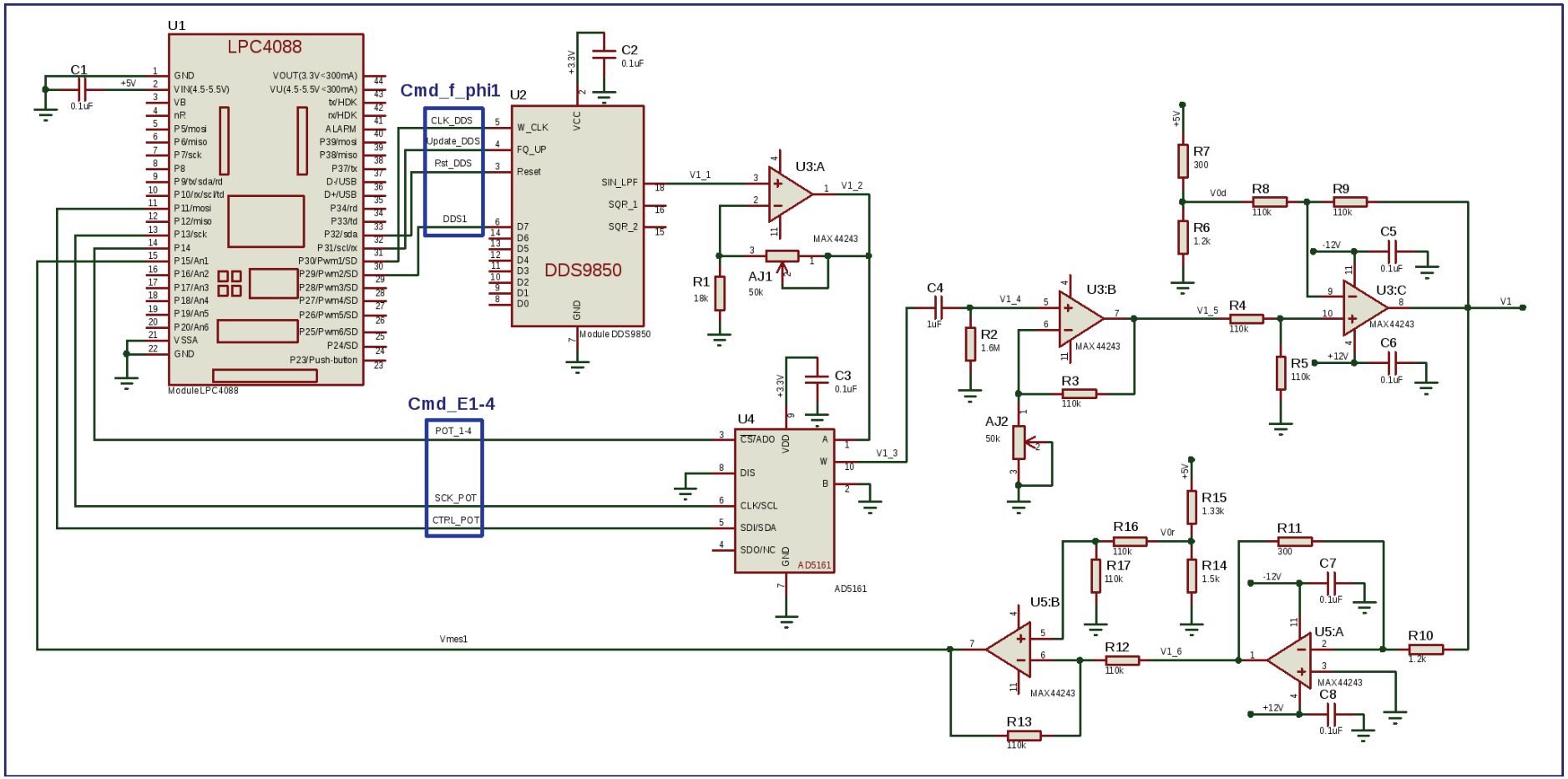
DT1 : Diagramme de bloc interne (ibd) de la carte de commande



Tournez la page S.V.P.

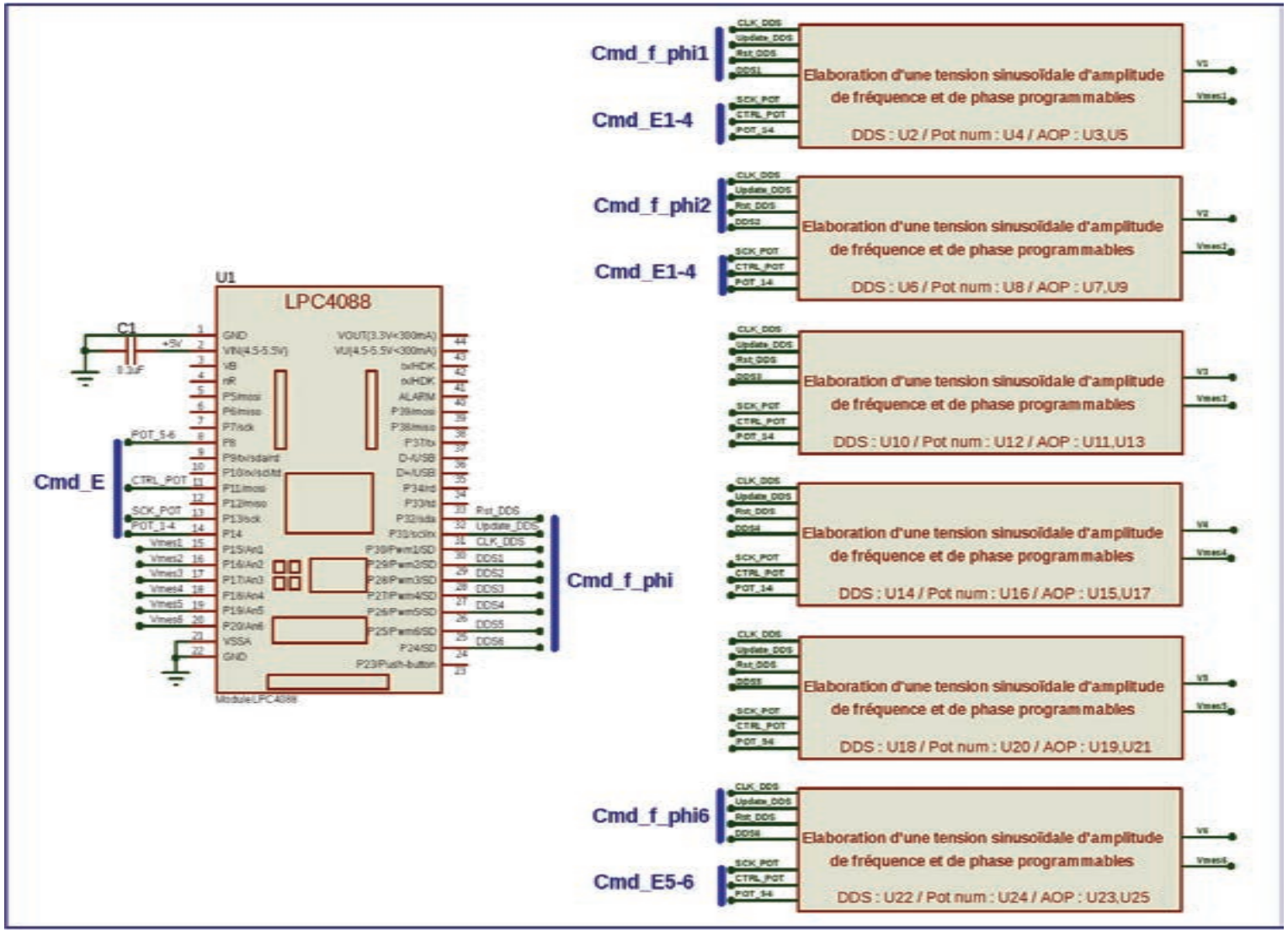
C

DT2 : Schéma structurel partiel (commande de l'actionneur piézoélectrique n°1)



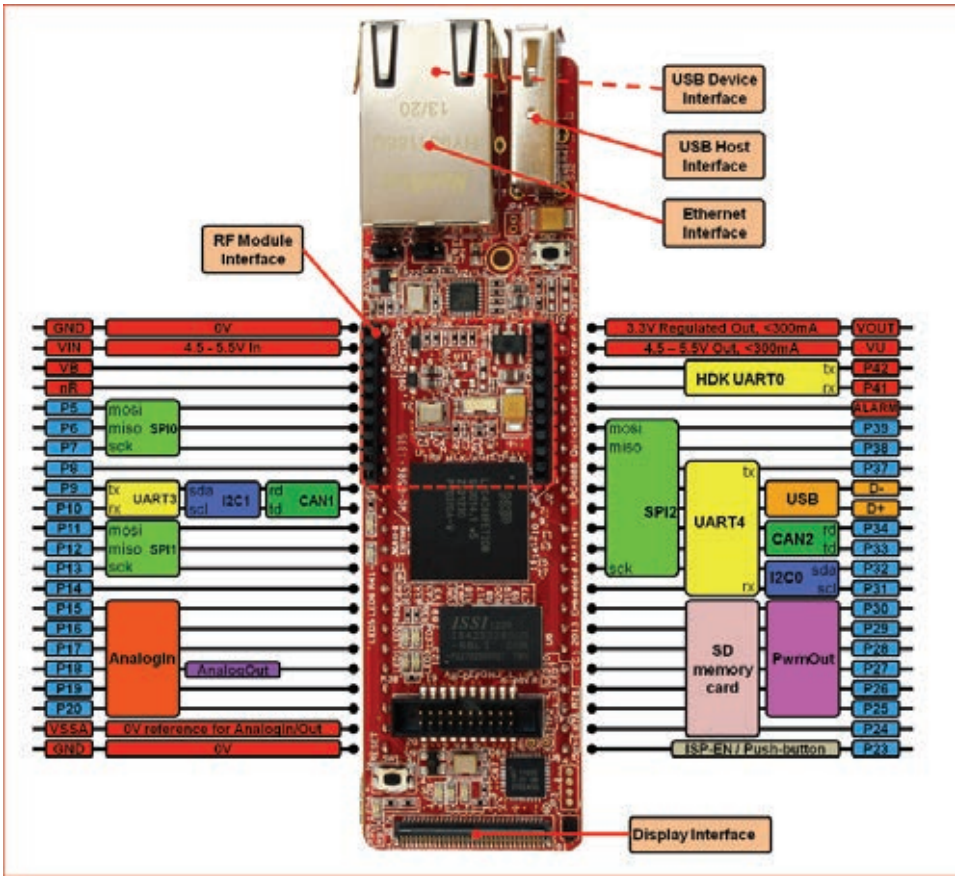
Page **17** sur **40**

DT3 : Schéma structurel global de la carte de commande



Tournez la page S.V.P.

DT4 : le module de prototypage rapide LPC4088 QuickStart Board



Le module de prototypage rapide « LPC4088 QuickStart Board » commercialisé par le constructeur Embedded Artists et qui équipe la carte de commande, intègre le microcontrôleur LPC4088 (constructeur NXP) architecturé autour d'un cœur ARM 32 bits de type Cortex M4 cadencé à 120 MHz, disposant de 512ko de mémoire flash programme, 96ko de mémoire de données SRAM et 4032 octets de mémoire de données EEPROM.

Il permet de programmer un certain nombre de périphériques intégrés au µC, parmi lesquels :

- des ports d'E/S ;
- 3 interfaces série synchrone SPI ;
- 2 interfaces série synchrone I2C ;
- 3 interfaces série asynchrone UART ;
- 2 interfaces CAN ;
- 1 interface USB ;
- 1 convertisseur analogique numérique à 6 entrées multiplexées ;
- 1 convertisseur numérique analogique.

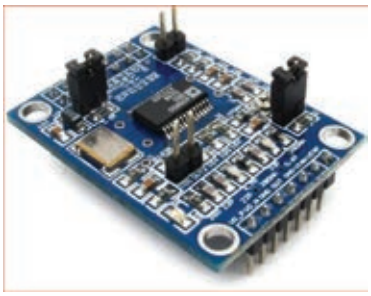
Il dispose en outre, en matière de connectique, d'un connecteur Ethernet RJ45, de deux connecteurs USB micro-B, d'un connecteur USB-A ainsi que d'une interface RF de type Xbee.

Ce module s'implante sur la carte de commande via deux connecteurs SIL mâles de 22 broches chacun (RF Module Interface).

Enfin, la programmation du microcontrôleur s'effectue en langage C/C++ via la plateforme de développement logiciel mbed (compilateur C en ligne).

DT5 : le module de prototypage rapide DDSAD9850

Le module de prototypage rapide « DDSAD9850 » est équipé du générateur de signaux sinusoïdaux à synthèse directe de référence DDS AD9850 (constructeur Analog Devices) cadencé par un oscillateur à quartz de fréquence 125MHz.
Il se positionne sur la carte de commande au moyen de 3 connecteurs SIL mâles de 7 broches chacun.



DT6 : extrait de la documentation du composant AD9850 (1 sur 4)

ANALOG DEVICES

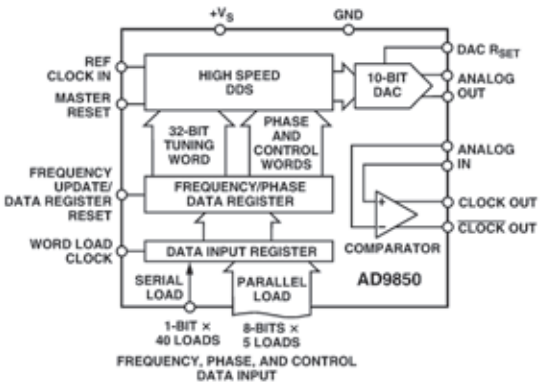
CMOS, 125 MHz Complete DDS Synthesizer

AD9850

FEATURES
125 MHz Clock Rate
On-Chip High Performance DAC and High Speed Comparator
DAC SFDR > 50 dB @ 40 MHz A_{OUT}
32-Bit Frequency Tuning Word
Simplified Control Interface: Parallel Byte or Serial Loading Format
Phase Modulation Capability
3.3 V or 5 V Single-Supply Operation
Low Power: 380 mW @ 125 MHz (5 V)
155 mW @ 110 MHz (3.3 V)
Power-Down Function
Ultrasmall 28-Lead SSOP Packaging

APPLICATIONS
Frequency/Phase—Agile Sine Wave Synthesis
Clock Recovery and Locking Circuitry for Digital Communications
Digitally Controlled ADC Encode Generator
Agile Local Oscillator Applications

FUNCTIONAL BLOCK DIAGRAM



GENERAL DESCRIPTION

The AD9850 is a highly integrated device that uses advanced DDS technology coupled with an internal high speed, high performance D/A converter and comparator to form a complete, digitally programmable frequency synthesizer and clock generator function. When referenced to an accurate clock source, the AD9850 generates a spectrally pure, frequency/phase programmable, analog output sine wave. This sine wave can be used directly as a frequency source, or it can be converted to a square wave for agile-clock generator applications. The AD9850's innovative high speed DDS core provides a 32-bit frequency tuning word, which results in an output tuning resolution of 0.0291 Hz for a 125 MHz reference clock input. The AD9850's circuit architecture allows the generation of output frequencies of up to one-half the reference clock frequency (or 62.5 MHz), and the output frequency can be digitally changed (asynchronously) at a rate of up to 23 million new frequencies per second. The device also provides five bits of digitally controlled phase modulation, which enables phase shifting of its output in increments of 180°, 90°, 45°, 22.5°, 11.25°, and any combination thereof. The AD9850 also contains a high speed comparator that can be configured to accept the (externally) filtered output of the DAC to generate a low jitter square wave output. This facilitates the device's use as an agile clock generator function.

The frequency tuning, control, and phase modulation words are loaded into the AD9850 via a parallel byte or serial loading format. The parallel load format consists of five iterative loads of an 8-bit control word (byte). The first byte controls phase modulation, power-down enable, and loading format; Bytes 2 to 5 comprise the 32-bit frequency tuning word. Serial loading is accomplished via a 40-bit serial data stream on a single pin. The AD9850 Complete DDS uses advanced CMOS technology to provide this breakthrough level of functionality and performance on just 155 mW of power dissipation (3.3 V supply).

The AD9850 is available in a space-saving 28-lead SSOP, surface-mount package. It is specified to operate over the extended industrial temperature range of –40°C to +85°C.

Tournez la page S.V.P.

DT6 : extrait de la documentation du composant AD9850 (2 sur 4)

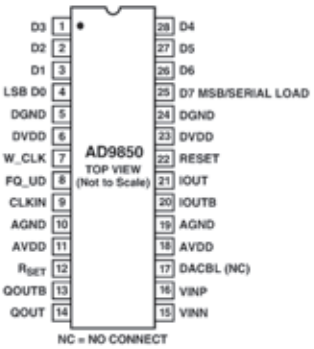


Table I. PIN FUNCTION DESCRIPTIONS

Pin No.	Mnemonic	Function
4 to 1, 28 to 25	D0 to D7	8-Bit Data Input. This is the 8-bit data port for iteratively loading the 32-bit frequency and the 8-bit phase/control word. D7 = MSB; D0 = LSB. D7 (Pin 25) also serves as the input pin for the 40-bit serial data-word.
5, 24	DGND	Digital Ground. These are the ground return leads for the digital circuitry.
6, 23	DVDD	Supply Voltage Leads for Digital Circuitry.
7	W_CLK	Word Load Clock. This clock is used to load the parallel or serial frequency/phase/control words.
8	FQ_UD	Frequency Update. On the rising edge of this clock, the DDS updates to the frequency (or phase) loaded in the data input register; it then resets the pointer to Word 0.
9	CLKIN	Reference Clock Input. This may be a continuous CMOS-level pulse train or sine input biased at 1/2 V supply. The rising edge of this clock initiates operation.
10, 19	AGND	Analog Ground. These leads are the ground return for the analog circuitry (DAC and comparator).
11, 18	AVDD	Supply Voltage for the Analog Circuitry (DAC and Comparator).
12	R_{SET}	DAC's External R_{SET} Connection. This resistor value sets the DAC full-scale output current. For normal applications (F_S I_{OUT} = 10 mA), the value for R_{SET} is 3.9 kΩ connected to ground. The R_{SET}/I_{OUT} relationship is I_{OUT} = 32 (1.248 V/R_{SET}).
13	QOUTB	Output Complement. This is the comparator's complement output.
14	QOUT	Output True. This is the comparator's true output.
15	VINN	Inverting Voltage Input. This is the comparator's negative input.
16	VINP	Noninverting Voltage Input. This is the comparator's positive input.
17	DACBL (NC)	DAC Baseline. This is the DAC baseline voltage reference; this lead is internally bypassed and should normally be considered a no connect for optimum performance.
20	IOUTB	Complementary Analog Output of the DAC.
21	IOUT	Analog Current Output of the DAC.
22	RESET	Reset. This is the master reset function; when set high, it clears all registers (except the input register), and the DAC output goes to cosine 0 after additional clock cycles—see Figure 7.

THEORY OF OPERATION AND APPLICATION

The AD9850 uses direct digital synthesis (DDS) technology, in the form of a numerically controlled oscillator, to generate a frequency/phase-agile sine wave. The digital sine wave is converted to analog form via an internal 10-bit high speed D/A converter, and an on-board high speed comparator is provided to translate the analog sine wave into a low jitter TTL/CMOS compatible output square wave. DDS technology is an innovative circuit architecture that allows fast and precise manipulation of its output frequency under full digital control. DDS also enables very high resolution in the incremental selection of output frequency; the AD9850 allows an output frequency resolution of 0.0291 Hz with a 125 MHz reference clock applied. The AD9850's output waveform is phase continuous when changed.

The basic functional block diagram and signal flow of the AD9850 configured as a clock generator is shown in Figure 4.

The DDS circuitry is basically a digital frequency divider function whose incremental resolution is determined by the frequency of the reference clock divided by the 2^N number of bits in the tuning word. The phase accumulator is a variable-modulus counter that increments the number stored in it each time it receives a clock pulse. When the counter overflows, it wraps around, making the phase accumulator's output contiguous.

The frequency tuning word sets the modulus of the counter, which effectively determines the size of the increment (Δ Phase) that is added to the value in the phase accumulator on the next clock pulse. The larger the added increment, the faster the accumulator overflows, which results in a higher output frequency. The AD9850 uses an innovative and proprietary algorithm that mathematically converts the 14-bit truncated value of the phase accumulator to the appropriate COS value. This unique algorithm uses a much reduced ROM look-up table and DSP techniques to perform this function, which contributes to the small size and low power dissipation of the AD9850. The relationship of the output frequency, reference clock, and tuning word of the AD9850 is determined by the formula

$$f_{OUT} = (\Delta\ Phase \times CLKIN)/2^{32}$$

where:
$\Delta Phase$ is the value of the 32-bit tuning word.
$CLKIN$ is the input reference clock frequency in MHz.
f_{OUT} is the frequency of the output signal in MHz.

The digital sine wave output of the DDS block drives the internal high speed 10-bit D/A converter that reconstructs the sine wave in analog form. This DAC has been optimized for dynamic performance and low glitch energy as manifested in the low jitter performance of the AD9850. Because the output of the

DT6 : extrait de la documentation du composant AD9850 (3 sur 4)

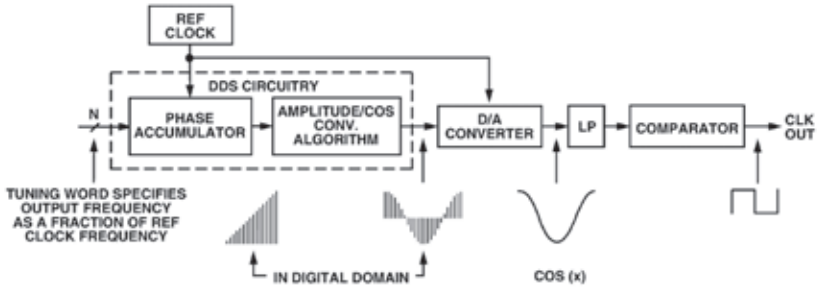


Figure 4. Basic DDS Block Diagram and Signal Flow of AD9850

AD9850 is a sampled signal, its output spectrum follows the Nyquist sampling theorem. Specifically, its output spectrum contains the fundamental plus aliased signals (images) that occur at multiples of the reference clock frequency ± the selected output frequency. A graphical representation of the sampled spectrum, with aliased images, is shown in Figure 5.

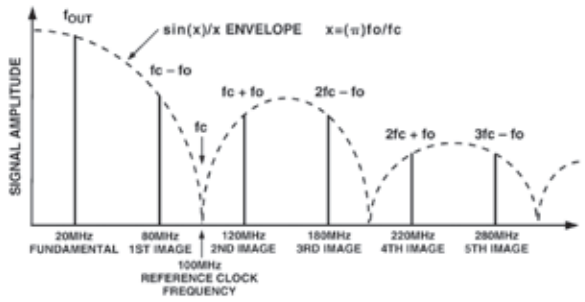


Figure 5. Output Spectrum of a Sampled Signal

In this example, the reference clock is 100 MHz and the output frequency is set to 20 MHz. As can be seen, the aliased images are very prominent and of a relatively high energy level as determined by the sin(x)/x roll-off of the quantized D/A converter output. In fact, depending on the fo/reference clock relationship, the first aliased image can be on the order of –3 dB below the fundamental. A low-pass filter is generally placed between the output of the D/A converter and the input of the comparator to further suppress the effects of aliased images. Obviously, consideration must be given to the relationship of the selected output frequency and the reference clock frequency to avoid unwanted (and unexpected) output anomalies.

To apply the AD9850 as a clock generator, limit the selected output frequency to <33% of reference clock frequency, and thereby avoid generating aliased signals that fall within, or close to, the output band of interest (generally dc-selected output frequency). This practice eases the complexity (and cost) of the external filter requirement for the clock generator application.

The reference clock frequency of the AD9850 has a minimum limitation of 1 MHz. The device has internal circuitry that senses when the minimum clock rate threshold has been exceeded and automatically places itself in the power-down mode. When in this state, if the clock frequency again exceeds the threshold, the device resumes normal operation. This shutdown mode prevents excessive current leakage in the dynamic registers of the device.

The D/A converter output and comparator inputs are available as differential signals that can be flexibly configured in any manner desired to achieve the objectives of the end system. The typical application of the AD9850 is with single-ended output/input analog signals, a single low-pass filter, and the generation of the comparator reference midpoint from the differential DAC output as shown in Figure 1.

Programming the AD9850

The AD9850 contains a 40-bit register that is used to program the 32-bit frequency control word, the 5-bit phase modulation word, and the power-down function. This register can be loaded in a parallel or serial mode.

In the parallel load mode, the register is loaded via an 8-bit bus; the full 40-bit word requires five iterations of the 8-bit word. The W_CLK and FQ_UD signals are used to address and load the registers. The rising edge of FQ_UD loads the (up to) 40-bit control data-word into the device and resets the address pointer to the first register. Subsequent W_CLK rising edges load the 8-bit data on words [7:0] and move the pointer to the next register. After *five* loads, W_CLK edges are ignored until either a reset or an FQ_UD rising edge resets the address pointer to the first register.

In serial load mode, subsequent rising edges of W_CLK shift the 1-bit data on Pin 25 (D7) through the 40 bits of programming information. After 40 bits are shifted through, an FQ_UD pulse is required to update the output frequency (or phase).

The function assignments of the data and control words are shown in Table III; the detailed timing sequence for updating the output frequency and/or phase, resetting the device, and powering up/down, are shown in the timing diagrams of Figures 6 through 12.

Note: There are specific control codes, used for factory test purposes, that render the AD9850 temporarily inoperable. The user must take deliberate precaution to avoid inputting the codes listed in Table II.

Table II. Factory Reserved Internal Test Control Codes

Loading Format	Factory Reserved Codes
Parallel	1) W0 = XXXXXX10 2) W0 = XXXXXX01
Serial	1) W32 = 1; W33 = 0 2) W32 = 0; W33 = 1 3) W32 = 1; W33 = 1

Tournez la page S.V.P.

DT6 : extrait de la documentation du composant AD9850 (4 sur 4)

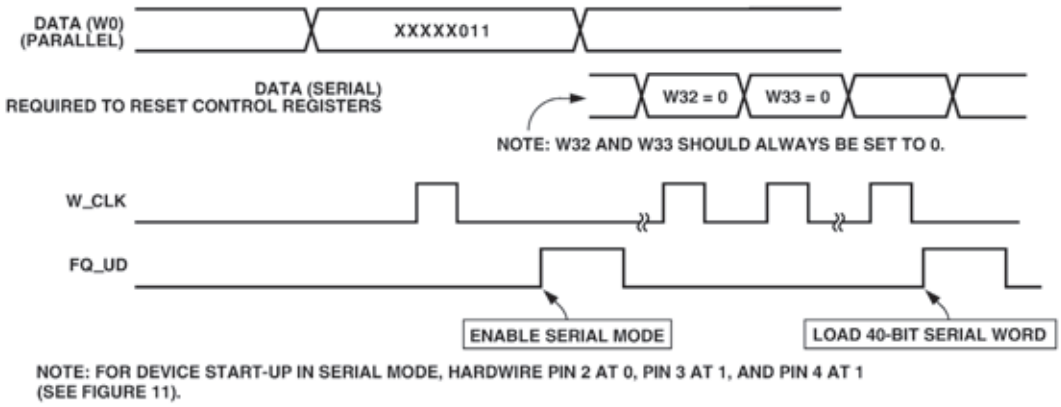


Figure 10. Serial Load Enable Sequence

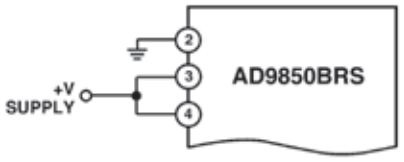


Figure 11. Pins 2 to 4 Connection for Default Serial Mode Operation

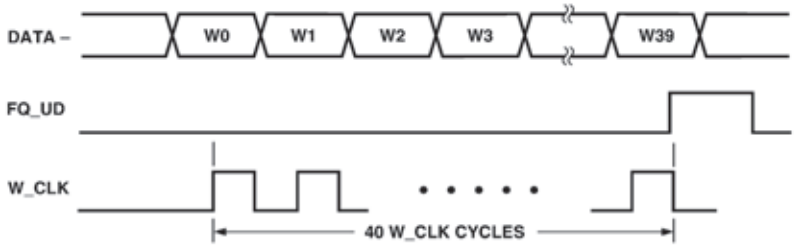


Figure 12. Serial Load Frequency/Phase Update Sequence

Table IV. 40-Bit Serial Load Word Function Assignment

W0	Freq-b0 (LSB)	W14	Freq-b14	W28	Freq-b28
W1	Freq-b1	W15	Freq-b15	W29	Freq-b29
W2	Freq-b2	W16	Freq-b16	W30	Freq-b30
W3	Freq-b3	W17	Freq-b17	W31	Freq-b31 (MSB)
W4	Freq-b4	W18	Freq-b18	W32	Control
W5	Freq-b5	W19	Freq-b19	W33	Control
W6	Freq-b6	W20	Freq-b20	W34	Power-Down
W7	Freq-b7	W21	Freq-b21	W35	Phase-b0 (LSB)
W8	Freq-b8	W22	Freq-b22	W36	Phase-b1
W9	Freq-b9	W23	Freq-b23	W37	Phase-b2
W10	Freq-b10	W24	Freq-b24	W38	Phase-b3
W11	Freq-b11	W25	Freq-b25	W39	Phase-b4 (MSB)
W12	Freq-b12	W26	Freq-b26		
W13	Freq-b13	W27	Freq-b27		

DT7 : extrait de la documentation du composant AD5161 (1 sur 5)

ANALOG DEVICES

256-Position SPI/I²C Selectable Digital Potentiometer

AD5161

FEATURES

256-position
End-to-end resistance 5 kΩ, 10 kΩ, 50 kΩ, 100 kΩ
Compact MSOP-10 (3 mm × 4.9 mm) package
Pin selectable SPI/I²C compatible interface
Extra package address decode pin AD0
Full read/write of wiper register
Power-on preset to midscale
Single supply 2.7 V to 5.5 V
Low temperature coefficient 45 ppm/°C
Low power, I_{DD} = 8 µA
Wide operating temperature −40°C to +125°C
SDO output allows multiple device daisy-chaining
Evaluation board available

APPLICATIONS

Mechanical potentiometer replacement in new designs
Transducer adjustment of pressure, temperature, position, chemical, and optical sensors
RF amplifier biasing
Gain control and offset adjustment

FUNCTIONAL BLOCK DIAGRAM

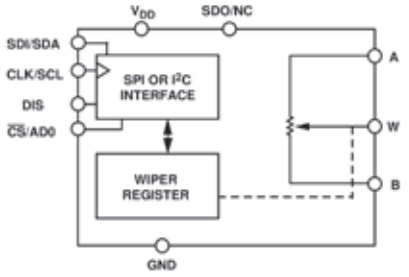


Figure 1.

PIN CONFIGURATION

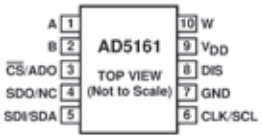


Figure 2.

GENERAL DESCRIPTION

The AD5161 provides a compact 3 mm × 4.9 mm packaged solution for 256-position adjustment applications. These devices perform the same electronic adjustment function as mechanical potentiometers or variable resistors, with enhanced resolution, solid-state reliability, and superior low temperature coefficient performance.

The wiper settings are controllable through a pin selectable SPI or I²C compatible digital interface, which can also be used to read back the wiper register content. When the SPI mode is used, the device can be daisy-chained (SDO to SDI), allowing several parts to share the same control lines. In the I²C mode, address pin AD0 can be used to place up to two devices on the same bus. In this same mode, command bits are available to reset the wiper position to midscale or to shut down the device into a state of zero power consumption.

Operating from a 2.7 V to 5.5 V power supply and consuming less than 5 µA allows for usage in portable battery-operated applications.

Table 5. Pin Function Description

Pin No.	Mnemonic	Description
1	A	A Terminal.
2	B	B Terminal.
3	$\overline{CS}$/AD0	Chip Select ($\overline{CS}$) Input, Active Low. When $\overline{CS}$ returns high, data will be loaded into the DAC register. Programmable address bit 0 (AD0) for multiple package decoding.
4	SDO/NC	Serial Data Output (SDO). Open-drain transistor requires pull-up resistor. No Connect (NC).
5	SDI/SDA	Serial Data Input (SDI). Serial Data Input/Output (SDA).
6	CLK/SCL	Serial Clock Input. Positive edge triggered.
7	GND	Digital Ground.
8	DIS	Digital Interface Select (SPI/I²C Select). SPI when DIS = 0, I²C when DIS = 1.
9	V_{DD}	Positive Power Supply.
10	W	W Terminal.

Tournez la page S.V.P.

DT7 : extrait de la documentation du composant AD5161 (2 sur 5)

SPI INTERFACE

Table 6. AD5161 Serial Data-Word Format

B7	B6	B5	B4	B3	B2	B1	B0
D7 MSB 2^7	D6	D5	D4	D3	D2	D1	D0 LSB 2^0

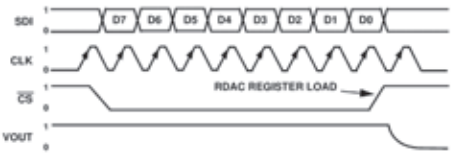


Figure 37. SPI Interface Timing Diagram (V_A = 5 V, V_B = 0 V, V_W = V_{OUT})

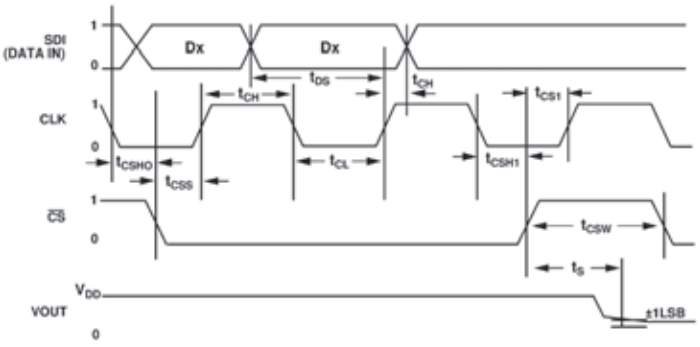


Figure 38. SPI Interface Detailed Timing Diagram (V_A = 5 V, V_B = 0 V, V_W = V_{OUT})

I²C INTERFACE

Table 7. Write Mode

S	0	1	0	1	1	0	AD0	$\overline{W}$	A	X	RS	SD	X	X	X	X	X	A	D7	D6	D5	D4	D3	D2	D1	D0	A	P
	Slave Address Byte									Instruction Byte										Data Byte									

S = Start Condition

P = Stop Condition

A = Acknowledge

X = Don't Care

$\overline{W}$ = Write

R = Read

RS = Reset wiper to Midscale 80_H

SD = Shutdown connects wiper to B terminal and open circuits A terminal. It does not change contents of wiper register.

D7, D6, D5, D4, D3, D2, D1, D0 = Data Bits.

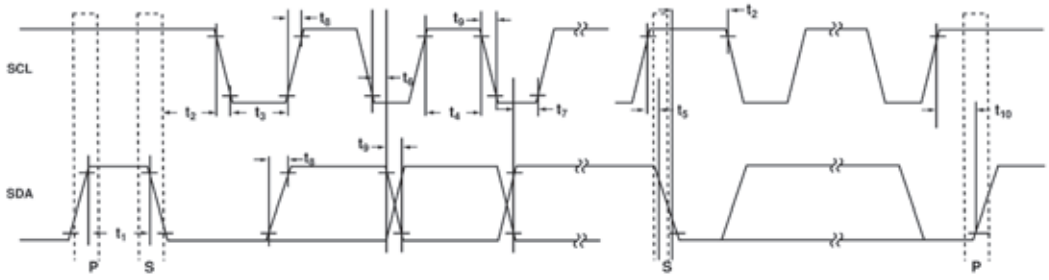


Figure 39. I²C Interface Detailed Timing Diagram

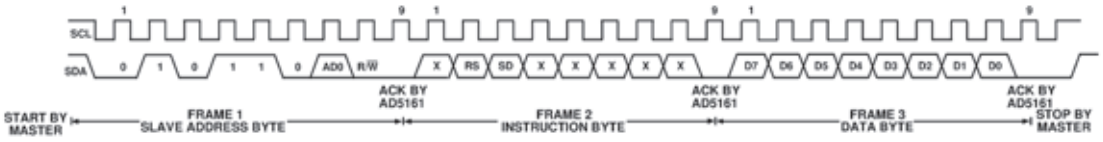


Figure 40. Writing to the RDAC Register

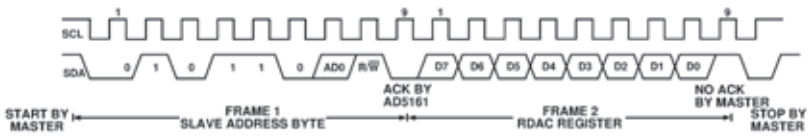


Figure 41. Reading Data from a Previously Selected RDAC Register in Write Mode

DT7 : extrait de la documentation du composant AD5161 (3 sur 5)

THEORY OF OPERATION

The AD5161 is a 256-position digitally controlled variable resistor (VR)[1] device.

An internal power-on preset places the wiper at midscale during power-on, which simplifies the fault condition recovery at power-up.

PROGRAMMING THE VARIABLE RESISTOR

Rheostat Operation

The nominal resistance of the RDAC between terminals A and B is available in 5 kΩ, 10 kΩ, 50 kΩ, and 100 kΩ. The final two or three digits of the part number determine the nominal resistance value, e.g., 10 kΩ = 10; 50 kΩ = 50. The nominal resistance (R_{AB}) of the VR has 256 contact points accessed by the wiper terminal, plus the B terminal contact. The 8-bit data in the RDAC latch is decoded to select one of the 256 possible settings. Assume a 10 kΩ part is used, the wiper's first connection starts at the B terminal for data 0x00. Since there is a 60 Ω wiper contact resistance, such connection yields a minimum of 60 Ω resistance between Terminals W and B. The second connection is the first tap point, which corresponds to 99 Ω ($R_{WB} = R_{AB}/256 + R_W = 39\ \Omega + 60\ \Omega$) for data 0x01. The third connection is the next tap point, representing 177 Ω ($2 \times 39\ \Omega + 60\ \Omega$) for data 0x02 and so on. Each LSB data value increase moves the wiper up the resistor ladder until the last tap point is reached at 9961 Ω ($R_{AB} - 1$ LSB + R_W). Figure 42 shows a simplified diagram of the equivalent RDAC circuit where the last resistor string will not be accessed; therefore, there is 1 LSB less of the nominal resistance at full scale in addition to the wiper resistance.

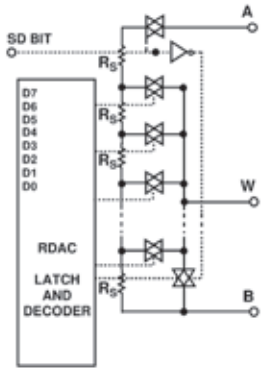


Figure 42. AD5161 Equivalent RDAC Circuit

[1] The terms digital potentiometer, VR, and RDAC are used interchangeably.

The general equation determining the digitally programmed output resistance between W and B is

$$R_{WB}(D) = \frac{D}{256} \times R_{AB} + R_W \quad (1)$$

where D is the decimal equivalent of the binary code loaded in the 8-bit RDAC register, R_{AB} is the end-to-end resistance, and R_W is the wiper resistance contributed by the on resistance of the internal switch.

In summary, if R_{AB} = 10 kΩ and the A terminal is open circuited, the following output resistance R_{WB} will be set for the indicated RDAC latch codes.

Table 9. Codes and Corresponding R_{WB} Resistance

D (Dec.)	R_{WB} (Ω)	Output State
255	9,961	Full Scale ($R_{AB} - 1$ LSB + R_W)
128	5,060	Midscale
1	99	1 LSB
0	60	Zero Scale (Wiper Contact Resistance)

Note that in the zero-scale condition a finite wiper resistance of 60 Ω is present. Care should be taken to limit the current flow between W and B in this state to a maximum pulse current of no more than 20 mA. Otherwise, degradation or possible destruction of the internal switch contact can occur.

Similar to the mechanical potentiometer, the resistance of the RDAC between the wiper W and terminal A also produces a digitally controlled complementary resistance R_{WA}. When these terminals are used, the B terminal can be opened. Setting the resistance value for R_{WA} starts at a maximum value of resistance and decreases as the data loaded in the latch increases in value. The general equation for this operation is

$$R_{WA}(D) = \frac{256 - D}{256} \times R_{AB} + R_W \quad (2)$$

For R_{AB} = 10 kΩ and the B terminal open circuited, the following output resistance R_{WA} will be set for the indicated RDAC latch codes.

Table 10. Codes and Corresponding R_{WA} Resistance

D (Dec.)	R_{WA} (Ω)	Output State
255	99	Full Scale
128	5,060	Midscale
1	9,961	1 LSB
0	10,060	Zero Scale

Typical device to device matching is process lot dependent and may vary by up to ±30%. Since the resistance element is processed in thin film technology, the change in R_{AB} with temperature has a very low 45 ppm/°C temperature coefficient.

Tournez la page S.V.P.

DT7 : extrait de la documentation du composant AD5161 (4 sur 5)

PROGRAMMING THE POTENTIOMETER DIVIDER

Voltage Output Operation

The digital potentiometer easily generates a voltage divider at wiper-to-B and wiper-to-A proportional to the input voltage at A-to-B. Unlike the polarity of V_{DD} to GND, which must be positive, voltage across A-B, W-A, and W-B can be at either polarity.

If ignoring the effect of the wiper resistance for approximation, connecting the A terminal to 5 V and the B terminal to ground produces an output voltage at the wiper-to-B starting at 0 V up to 1 LSB less than 5 V. Each LSB of voltage is equal to the voltage applied across terminal AB divided by the 256 positions of the potentiometer divider. The general equation defining the output voltage at V_W with respect to ground for any valid input voltage applied to terminals A and B is

$$V_W(D) = \frac{D}{256}V_A + \frac{256-D}{256}V_B \quad (3)$$

For a more accurate calculation, which includes the effect of wiper resistance, V_W, can be found as

$$V_W(D) = \frac{R_{WB}(D)}{256}V_A + \frac{R_{WA}(D)}{256}V_B \quad (4)$$

Operation of the digital potentiometer in the divider mode results in a more accurate operation over temperature. Unlike the rheostat mode, the output voltage is dependent mainly on the ratio of the internal resistors R_{WA} and R_{WB} and not the absolute values. Therefore, the temperature drift reduces to 15 ppm/°C.

PIN SELECTABLE DIGITAL INTERFACE

The AD5161 provides the flexibility of a selectable interface. When the digital interface select (DIS) pin is tied low, the SPI mode is engaged. When the DIS pin is tied high, the I²C mode is engaged.

SPI Compatible 3-Wire Serial Bus (DIS = 0)

The AD5161 contains a 3-wire SPI compatible digital interface (SDI, $\overline{CS}$, and CLK). The 8-bit serial word must be loaded MSB first. The format of the word is shown in Table 6.

The positive-edge sensitive CLK input requires clean transitions to avoid clocking incorrect data into the serial input register. Standard logic families work well. If mechanical switches are used for product evaluation, they should be debounced by a flip-flop or other suitable means. When $\overline{CS}$ is low, the clock loads data into the serial register on each positive clock edge (see Figure 37).

The data setup and data hold times in the specification table determine the valid timing requirements. The AD5161 uses an 8-bit serial input data register word that is transferred to the internal RDAC register when the $\overline{CS}$ line returns to logic high. Extra MSB bits are ignored.

Daisy-Chain Operation

The serial data output (SDO) pin contains an open-drain N-channel FET. This output requires a pull-up resistor in order to transfer data to the next package's SDI pin. This allows for daisy-chaining several RDACs from a single processor serial data line. The pull-up resistor termination voltage can be larger than the V_{DD} supply voltage. It is recommended to increase the clock period when using a pull-up resistor to the SDI pin of the following device because capacitive loading at the daisy-chain node SDO-SDI between devices may induce time delay to subsequent devices. Users should be aware of this potential problem to achieve data transfer successfully (see Figure 43). If two AD5161s are daisy-chained, a total of at least 16 bits of data is required. The first eight bits, complying with the format shown in Table 6, go to U2 and the second eight bits with the same format go to U1. $\overline{CS}$ should be kept low until all 16 bits are clocked into their respective serial registers. After this, $\overline{CS}$ is pulled high to complete the operation and load the RDAC latch. If the data word during the $\overline{CS}$ low period is greater than 16 bits, any additional MSBs will be discarded.

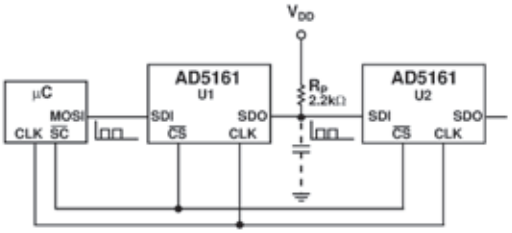


Figure 43. Daisy-Chain Configuration

I²C Compatible 2-Wire Serial Bus (DIS = 1)

The AD5161 can also be controlled via an I²C compatible serial bus with DIS tied high. The RDACs are connected to this bus as slave devices.

The first byte of the AD5161 is a slave address byte (see Table 7 and Table 8). It has a 7-bit slave address and a R/$\overline{W}$ bit. The six MSBs of the slave address are 010110, and the following bit is determined by the state of the AD0 pin of the device. AD0 allows the user to place up to two of the I²C compatible devices on one bus.

The 2-wire I²C serial bus protocol operates as follows:

1. The master initiates data transfer by establishing a START condition, which is when a high-to-low transition on the SDA line occurs while SCL is high (see Figure 40). The following byte is the slave address byte, which consists of the 7-bit slave address followed by an R/$\overline{W}$ bit (this bit determines whether data will be read from or written to the slave device).

DT7 : extrait de la documentation du composant AD5161 (5 sur 5)

The slave whose address corresponds to the transmitted address responds by pulling the SDA line low during the ninth clock pulse (this is termed the acknowledge bit). At this stage, all other devices on the bus remain idle while the selected device waits for data to be written to or read from its serial register. If the R/$\overline{W}$ bit is high, the master will read from the slave device. On the other hand, if the R/$\overline{W}$ bit is low, the master will write to the slave device.

2. A write operation contains an extra instruction byte that a read operation does not contain. Such an instruction byte in write mode follows the slave address byte. The first bit (MSB) of the instruction byte is a don't care.

   The second MSB, RS, is the midscale reset. A logic high on this bit moves the wiper to the center tap where $R_{WA} = R_{WB}$. This feature effectively writes over the contents of the register, and thus, when taken out of reset mode, the RDAC will remain at midscale.

   The third MSB, SD, is a shutdown bit. A logic high causes an open circuit at terminal A while shorting the wiper to terminal B. This operation yields almost 0 Ω in rheostat mode or 0 V in potentiometer mode. It is important to note that the shutdown operation does not disturb the contents of the register. When brought out of shutdown, the previous setting will be applied to the RDAC. Also, during shutdown, new settings can be programmed. When the part is returned from shutdown, the corresponding VR setting will be applied to the RDAC.

   The remainder of the bits in the instruction byte are don't cares (see Table 7).

3. After acknowledging the instruction byte, the last byte in write mode is the data byte. Data is transmitted over the serial bus in sequences of nine clock pulses (eight data bits followed by an acknowledge bit). The transitions on the SDA line must occur during the low period of SCL and remain stable during the high period of SCL (see Table 7).

4. In the read mode, the data byte follows immediately after the acknowledgment of the slave address byte. Data is transmitted over the serial bus in sequences of nine clock pulses (a slight difference with the write mode, where there are eight data bits followed by an acknowledge bit). Similarly, the transitions on the SDA line must occur during the low period of SCL and remain stable during the high period of SCL (see Figure 41).

5. When all data bits have been read or written, a STOP condition is established by the master. A STOP condition is defined as a low-to-high transition on the SDA line while SCL is high. In write mode, the master will pull the SDA line high during the tenth clock pulse to establish a STOP condition (see Figure 40). In read mode, the master will issue a No Acknowledge for the ninth clock pulse (i.e., the SDA line remains high). The master will then bring the SDA line low before the tenth clock pulse which goes high to establish a STOP condition (see Figure 41).

A repeated write function gives the user flexibility to update the RDAC output a number of times after addressing and instructing the part only once. During the write cycle, each data byte will update the RDAC output. For example, after the RDAC has acknowledged its slave address and instruction bytes, the RDAC output will update after these two bytes. If another byte is written to the RDAC while it is still addressed to a specific slave device with the same instruction, this byte will update the output of the selected slave device. If different instructions are needed, the write mode has to start again with a new slave address, instruction, and data byte. Similarly, a repeated read function of the RDAC is also allowed.

Readback RDAC Value

The AD5161 allows the user to read back the RDAC values in the read mode. Refer to Table 7 and Table 8 for the programming format.

Multiple Devices on One Bus

Figure 44 shows two AD5161 devices on the same serial bus. Each has a different slave address since the states of their AD0 pins are different. This allows each RDAC within each device to be written to or read from independently. The master device output bus line drivers are open-drain pull-downs in a fully I²C compatible interface.

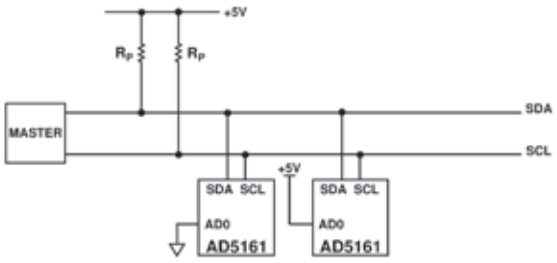


Figure 44. Multiple AD5161 Devices on One I²C Bus

Tournez la page S.V.P.

DT8 : le modèle de programmation (API) mbed (1 sur 4)

1. Broche de port d'E/S configurée en sortie

Instanciation
DigitalOut nom_donné_à_la_broche(*n° de broche*);

Syntaxe
Positionner la broche au niveau logique haut : nom_donné_à_la_broche=1;
Positionner la broche au niveau logique bas : nom_donné_à_la_broche=0;

Autre syntaxe (C++) : nom_donné_à_la_broche.write(*int valeur*); où *valeur* est égale à 0 ou 1
Positionner la broche au niveau logique haut : nom_donné_à_la_broche.write(*1*);
Positionner la broche au niveau logique bas : nom_donné_à_la_broche.write(*0*);

Exemple
```
DigitalOut     Cde_buzzer(p14);
Cde_buzzer=1;       // Mise au niveau haut de la broche p14 de commande de buzzer
```

2. Broches de port d'E/S configurées en bus de sortie

Instanciation
BusOut nom_donné_au_bus(*n° des broches séparées par des virgules*);

Syntaxe pour positionner les différentes broches du bus aux NL haut ou bas
nom_donné_au_bus=int valeur
Attention : le n° de broche qui suit la parenthèse ouvrante correspond au LSB de la valeur !

Exemple
```
BusOut         data_bus(p10,p7,p9,p6,p8,p11,p18,p16);
data_bus=45;    // Mise à 1 des broches p10, p9, p6 et p11 et à 0 des autres broches
```

3. Interface SPI (en mode maître exclusivement)

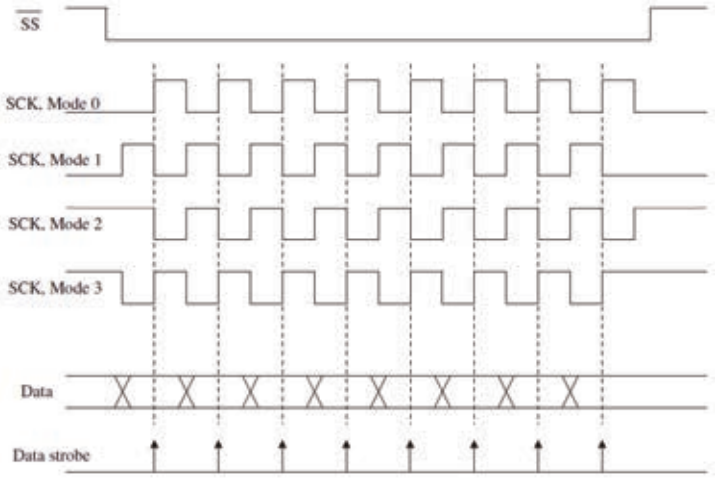
Une interface **SPI** (**Serial Peripheral Interface**) est une interface série synchrone autorisant des communications série de type full duplex, et présentant 4 broches appelées **mosi**, **miso**, **sclk** et **/ss**.
Lorsque l'interface SPI est configurée en mode maître, ce qui est le cas dans cette application :

- **mosi** est une broche de sortie transmettant à l'extérieur les données sous forme série,

DT8 : le modèle de programmation (API) mbed (2 sur 4)

- **miso** est une broche d'entrée recevant de l'extérieur les données sous forme série,
- **sclk** est une broche de sortie délivrant le signal de synchronisation des données,
- **ssel** est une broche de sortie passant au niveau bas lorsqu'il y a communication.

L'interface peut être configurée selon 4 modes de fonctionnement distincts (0 à 3) en fonction du type de synchronisme souhaité, dicté par les spécifications des composants avec lesquels elle communique.
On précise que les 2 interfaces SPI du module LPC4088 QuickStart Board ne comportent que 3 broches : mosi et miso (représentées ci-contre par le terme générique « Data ») ainsi que sclk (notée SCK).



Le format des données transmises et/ou reçues est paramétrable de 4 à 16 bits, de même que la fréquence d'horloge (30 MHz maximum).
Mettre en œuvre une interface SPI s'effectue en 2 phases successives : déclaration puis configuration.

Instanciation
SPI nom_donné_à_l'interface(*n° broche mosi,n° broche miso,n° broche sclk*);

Configuration
nom_donné_à_l'interface.format(*int nombre de bits,int n°de mode*);
nom_donné_à_l'interface.frequency(*int fréquence souhaitée en Hz*);

Syntaxe pour transmettre et recevoir une donnée
int variable_de stockage_de_la_donnée_reçue nom_donné_à_l'interface.write(*int donnée*);

Exemple
SPI com_spi(p39,p38,p32); //déclaration de la SPI « com_spi » et des broches associées

com_spi.format(16,3) // émission/réception de mots de 16 bits selon le mode 3
com_spi.frequency(5000000) // fréquence d'horloge d'émission/réception de 5 MHz

int donnee_recue=com_spi.write(0xabcd);
➔ transmet la donnée de 16 bits 0xabcd et mémorise la donnée reçue de 16 bits dans la variable « donnee_recue », le tout avec un débit de 5 Mbits/s et selon le mode 3.

Tournez la page S.V.P.

DT8 : le modèle de programmation (API) mbed (3 sur 4)

4. Temporisateur programmable (timer)

Le µC possède plusieurs temporisateurs programmables dont nous présentons ici une seule fonctionnalité (en liaison avec notre application) : le mode ***ticker***.
Le mode « ticker » consiste à déclencher une requête d'interruption à intervalles de temps réguliers.

Instanciation
Ticker nom_donné_au_ticker;

Validation
nom_donné_au_ticker.attach(*&nom de la fonction d'interruption à appeler,périodicité en s*);
ou bien
nom_donné_au_ticker.attach_us(*&nom de la fonction d'interruption à appeler,périodicité en µs*);

Invalidation
nom_donné_au_ticker.detach;

Exemple

```
void led_switch(void);
Ticker clignotant;
DigitalOut led(p14);

int main()
{
   led=0;
   clignotant.attach(&led_switch,0.25);
   while(1);
}

void led_switch()
{
   led=!led;
}
```

DT8 : le modèle de programmation (API) mbed (4 sur 4)

5. Le convertisseur analogique numérique CAN (ADC)

La µC possède un convertisseur analogique numérique de résolution 12 bits à 6 entrées multiplexées. Le temps de conversion Tc de la tension appliquée sur une entrée est de 2,5 µs.

Instanciation
AnalogIn nom_donné_à_l'entrée_analogique(*n° de broche*);

Syntaxes permettant de lire le résultat de conversion analogique numérique

a) Résultat réel codé entre 0,0 et 1,0

```
float variable_de_stockage_du_résultat=nom_donné_à_l'entrée.read();
```

➔ La valeur retournée est un réel codé entre 0,0 et 1,0.

b) Résultat entier codé sur 16 bits

```
unsigned short variable_de_stockage_du_résultat=nom_donné_à_l'entrée.read_u16();
```

➔ La valeur retournée est codée sur 16 bits et alignée à gauche donc 16 fois supérieure au résultat de conversion analogique numérique.

Exemples

```
AnalogIn can_3(p17);
AnalogIn can_6(p20);

float resultat_can_3_en_Volts=3.3*can_3.read();                  // résultat compris entre 0,0V et 3,3V
unsigned short resultat_can_6_en_binaire=can_6.read_u16()>>4;    // résultat compris entre 0 et 4095
```

6. La fonction sscanf() permettant de lire une chaine de caractères formatée

La fonction sscanf() analyse une chaine de caractères *texte* conformément au format *format* et stocke les résultats dans les espaces mémoire adressés par les pointeurs donnés en argument à la suite du format.
Le format traduit la forme attendue de la chaine de caractères, où les données à extraire sont identifiées par des balises :

- %d pour un entier ;
- %f pour un flottant ;
- etc.

Syntaxe permettant de lire une chaine de caractère formatée
int résultat = sscanf(const char **texte*, const char **format*, &*var1*, &*var2*, etc);

Exemples

```
int var1 ;
float var2 ;
char monTexte[10] = "12;15.34" ;
sscanf(monTexte, "%d;%f", &var1, &var2) ;

int tab[3] ;
char maLigne[10] = "xx5;6:7yy" ;
int resultat = sscanf(maLigne, "xx%d;%d:%dyy", &tab[0], &tab[1], &tab[2]) ;
if (resultat == 3) printf("Tableau reconnu.") ;
else printf("Tableau non reconnu.") ;
```


Tournez la page S.V.P.

DT9 : code source du programme de commande de la carte (1 sur 2)

```
int main(int argc, char *argv[])
{
   //Création du controleur
   Controleur *ctl=new Controleur();
   // Lancement de la méthode de fonctionnement normal run()
   ctl->run();
}

Controleur::Controleur()
{
   //Création et initialisation de l'objet DDS
   dds=new DDS;
   PinName pinDDS[6]= {p29,p28,p27,p26,p25,p24};    //Définition des broches de communication série
                                                    //"data" des DDS
   dds->init_DDS(p30,p31,p32,pinDDS);
   dds->envoi_f_phi();

   //Création et initialisation des deux objets Potentiometre
   pot=new Potentiometre[2];
   pot[0].init_Potentiometre(p11,p13,p14);
   pot[0].envoi_E();
   pot[1].init_Potentiometre(p11,p13,p15);
   pot[1].amplitude=2;
   pot[1].envoi_E();

   // Création et initialisation de l'objet Bluetooth
   bluetooth=new Interface_Bluetooth();
   bluetooth->init_Bluetooth(P4_22, P4_23);
}

void Controleur::run()
{
   while(true) {
         //Boucle attente d'une consigne en BT
         while(!bluetooth->get_reception_bt());
         // Remise à zéro du flag de réception d'un message
         bluetooth-> sr_receotion_bt(FALSE);
         bluetooth->CPT_I=0;
         bluetooth->acquisition_consignes(dds,pot);
         // Programmation des composants
         dds->envoi_f_phi();
         pot[0].envoi_E();
         pot[1].envoi_E();
   }
}
```

DT9 : code source du programme de commande de la carte (2 sur 2)

Extrait des méthodes de l'objet DDS :

```
void DDS::init_DDS (PinName CLK, PinName Update, PinName Rst, PinName *Data)
{
   // Définition des broches
   CLK_DDS = new DigitalOut(CLK) ;
   Update_DDS = new DigitalOut(Update,0) ;
   Rst_DDS = new DigitalOut(Rst,0) ;
   for (int i=0 ; i<6 ; i++) Data_DDS[i] = new DigitalOut(Data[i],0) ;
   //Initialisation des fréquences, phases et mot de configuration
   config=0x00 ;
   freq=1000 ;
   for (int i=0 ; i<6 ; i++) {
         tab_phi[i]=0 ;
   }
}

void DDS::pulse_CLK_DDS(void)
{
   wait_us(Tclock);       //Pause
   *CLK_DDS=1;            //Mise à 1 du signal d'horloge
   wait_us(Tclock);       //Pause
   *CLK_DDS=0;            //Mise à 0 du signal d'horloge
   wait_us(Tclock);       //Pause
}

// Génération d'un top de reset pour la transmission série du DDS
void DDS::pulse_Rst_DDS(void)
{
   wait_us(Tclock);       //Pause
   *Rst_DDS=1;            //Mise à 1 du signal de reset
   wait_us(Tclock);       //Pause
   *Rst_DDS=0;            //Mise à 0 du signal de reset
   wait_us(Tclock);       //Pause
}

// Génération d'un top de mise à jour pour la transmission série du DDS
void DDS::pulse_Update_DDS(void)
{
   wait_us(Tclock);       //Pause
   *Update_DDS=1;         //Mise à 1 du signal de mise à jour
   wait_us(Tclock);       //Pause
   *Update_DDS=0;         //Mise à 0 du signal de mise à jour
   wait_us(Tclock);       //Pause
}
```

Extrait des méthodes de l'objet Potentiometre :

```
void Potentiometre::init_SPI(void)
{
   pot_num->format();
   pot_num->frequency();
}
```


Tournez la page S.V.P.

DT10 : Fichier Python de l'interface graphique

```
import android, sys, time
from fullscreenwrapper2 import *

class Layout(Layout):
    def __init__(self):
        super(Layout,self).__init__(xmldata,"Application Doriane")

    def on_show(self):
        self.add_event(key_EventHandler(handler_function=self.close_app))
        self.views.but_exit.add_event(click_EventHandler(self.views.but_exit, self.close_app))
        self.views.but_connect.add_event(click_EventHandler(self.views.but_connect, self.connectBT))
        self.views.but_disconnect.add_event(click_EventHandler(self.views.but_disconnect, self.disconnectBT))
        self.views.but_envoi.add_event(click_EventHandler(self.views.but_envoi, self.envoi_donnees))
        self.est_fermee=False

    def on_close(self):
        pass

    def close_app(self,view,event):
        self.est_fermee=True
        FullScreenWrapper2App.exit_FullScreenWrapper2App()

    def connectBT(self,view, event):
        NON DETAILLEE

    def disconnectBT(self,view, event):
        NON DETAILLEE

    def envoi_donnees(self,view, event):
        Freq = self.views.txt_freq.text
        Amp1 = self.views.txt_amplitude14.text
        Amp2 = self.views.txt_amplitude56.text
        Pha1=self.views.spinner_phase1.text
        Pha2=self.views.spinner_phase2.text
        Pha3=self.views.spinner_phase3.text
        Pha4=self.views.spinner_phase4.text
        Pha5=self.views.spinner_phase5.text
        Pha6=self.views.spinner_phase6.text

        data = '!'+Freq+'|'+Pha1+'|'+Pha2+'|'+Pha3+'|'+Pha4+'|'+Pha5+'|'+Pha6+'|'+Amp1+'|'+Amp2+'?'

        droid.bluetoothWrite(data,self.connID)



# Identifiants du module Bluetooth
BT_DEVICE_ID = '20:15:07:20:94:86'
ssp_uuid = '00001101-0000-1000-8000-00805F9B34FB'

# Lancement de l'application
if __name__ == '__main__':
    droid = android.Android()
    FullScreenWrapper2App.initialize(droid)
    FullScreenWrapper2App.show_layout(Layout())
    FullScreenWrapper2App.eventloop()
```

Modèle CMEN-DOC v2 ©NEOPTEC

Nom de famille :
(Suivi, s'il y a lieu, du nom d'usage)

Prénom(s) :

Numéro Inscription :

(Le numéro est celui qui figure sur la convocation ou la feuille d'émargement)

Né(e) le : ☐☐ / ☐☐ / ☐☐☐☐

(Remplir cette partie à l'aide de la notice)

Concours / Examen : …….. **Section/Spécialité/Série :** ……

Epreuve : …….. **Matière :** …….. **Session :** ……..

CONSIGNES

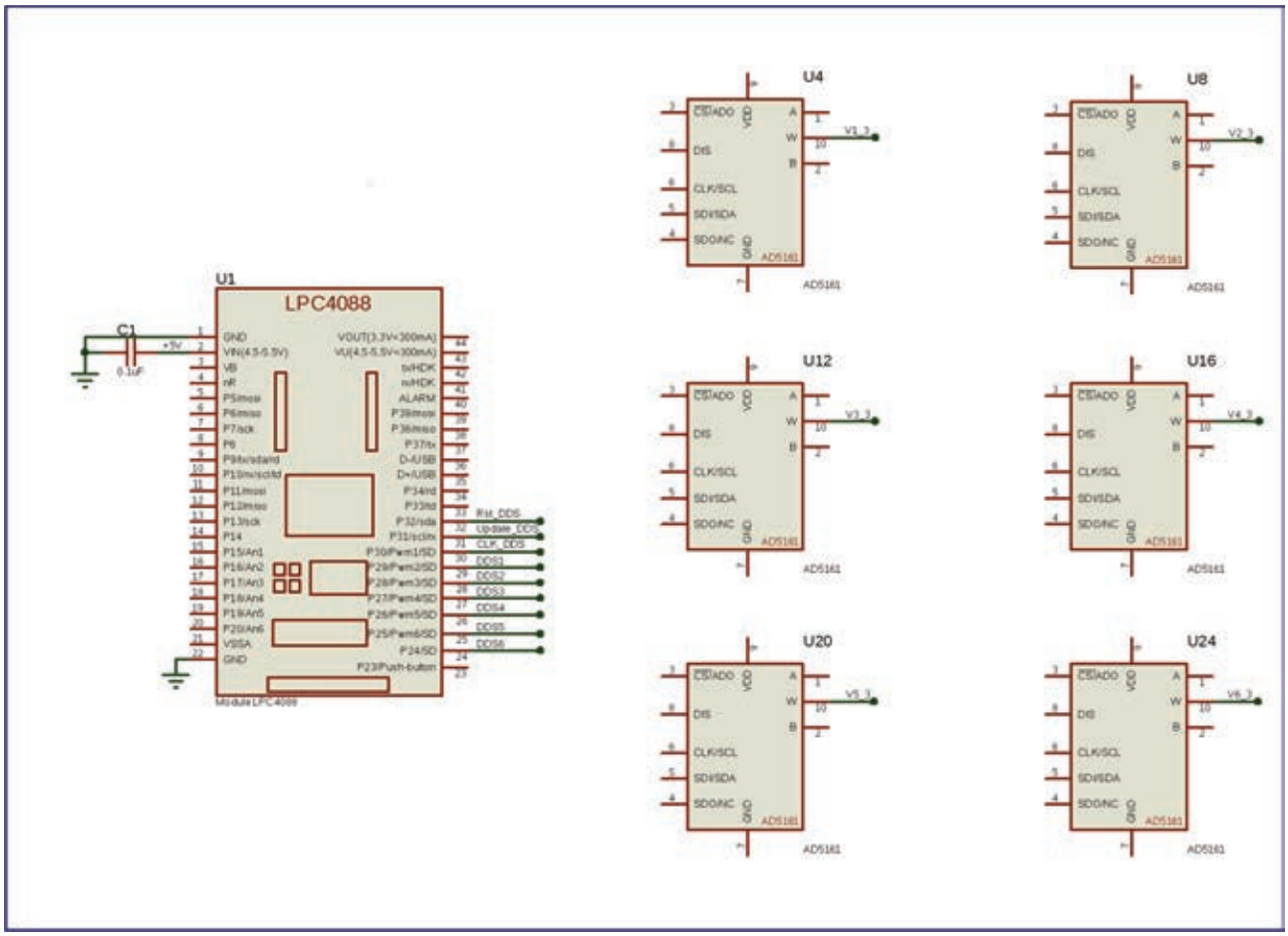
- *Remplir soigneusement, sur CHAQUE feuille officielle, la zone d'identification en MAJUSCULES.*
- *Ne pas signer la composition et ne pas y apporter de signe distinctif pouvant indiquer sa provenance.*
- *Numéroter chaque PAGE (cadre en bas à droite de la page) et placer les feuilles dans le bon sens et dans l'ordre.*
- *Rédiger avec un stylo à encre foncée (bleue ou noire) et ne pas utiliser de stylo plume à encre claire.*
- *N'effectuer aucun collage ou découpage de sujets ou de feuille officielle. Ne joindre aucun brouillon.*

DR1 - DR2 - DR3 - DR4 - DR5

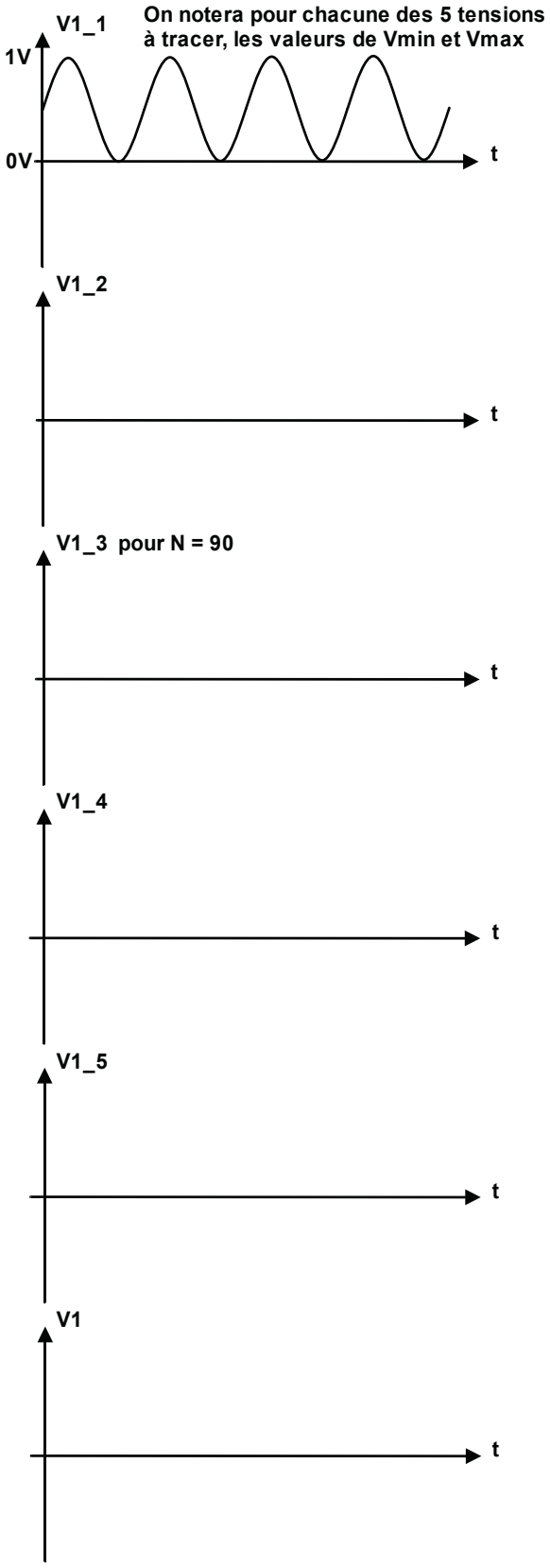
NE RIEN ECRIRE DANS CE CADRE

Documents réponse

DR4 (Q25)

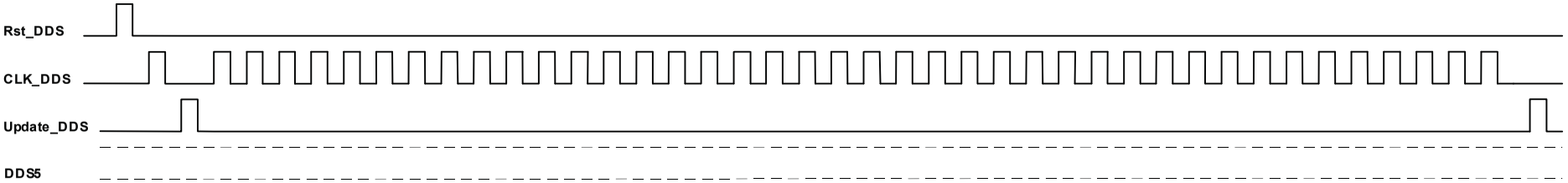


DR2 (Q15)

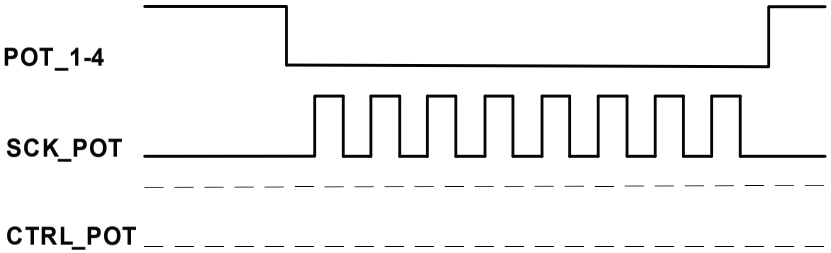


Tournez la page S.V.P.

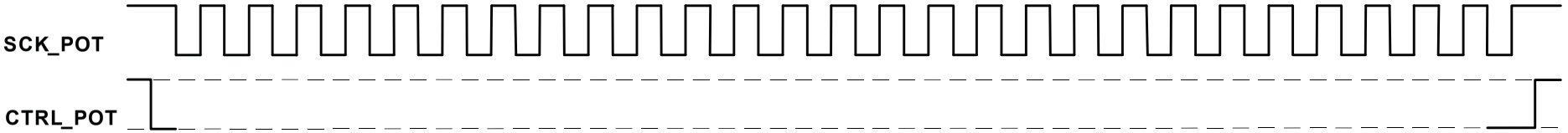
DR1 (Q6)



DR3 (Q22)



DR5 (Q26)



Modèle CMEN-DOC v2 ©NEOPTEC

Nom de famille :
(Suivi, s'il y a lieu, du nom d'usage)

Prénom(s) :

Numéro Inscription :

Né(e) le : / /

(Le numéro est celui qui figure sur la convocation ou la feuille d'émargement)

(Remplir cette partie à l'aide de la notice)

Concours / Examen : …….. **Section/Spécialité/Série :** ……

Epreuve : …….. **Matière :** …….. **Session :** ……..

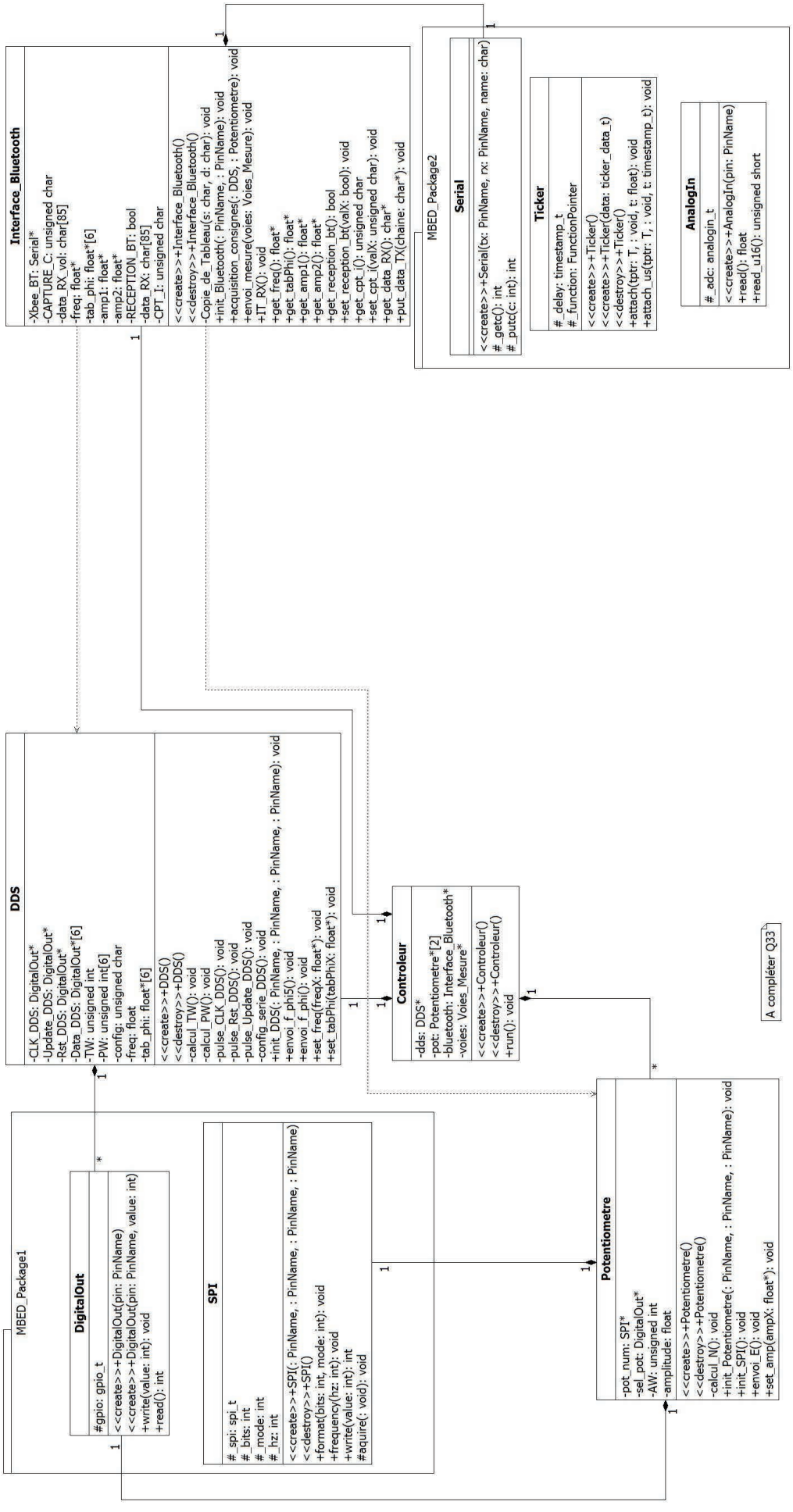
CONSIGNES

- *Remplir soigneusement, sur CHAQUE feuille officielle, la zone d'identification en MAJUSCULES.*
- *Ne pas signer la composition et ne pas y apporter de signe distinctif pouvant indiquer sa provenance.*
- *Numéroter chaque PAGE (cadre en bas à droite de la page) et placer les feuilles dans le bon sens et dans l'ordre.*
- *Rédiger avec un stylo à encre foncée (bleue ou noire) et ne pas utiliser de stylo plume à encre claire.*
- *N'effectuer aucun collage ou découpage de sujets ou de feuille officielle. Ne joindre aucun brouillon.*

DR6 - DR7

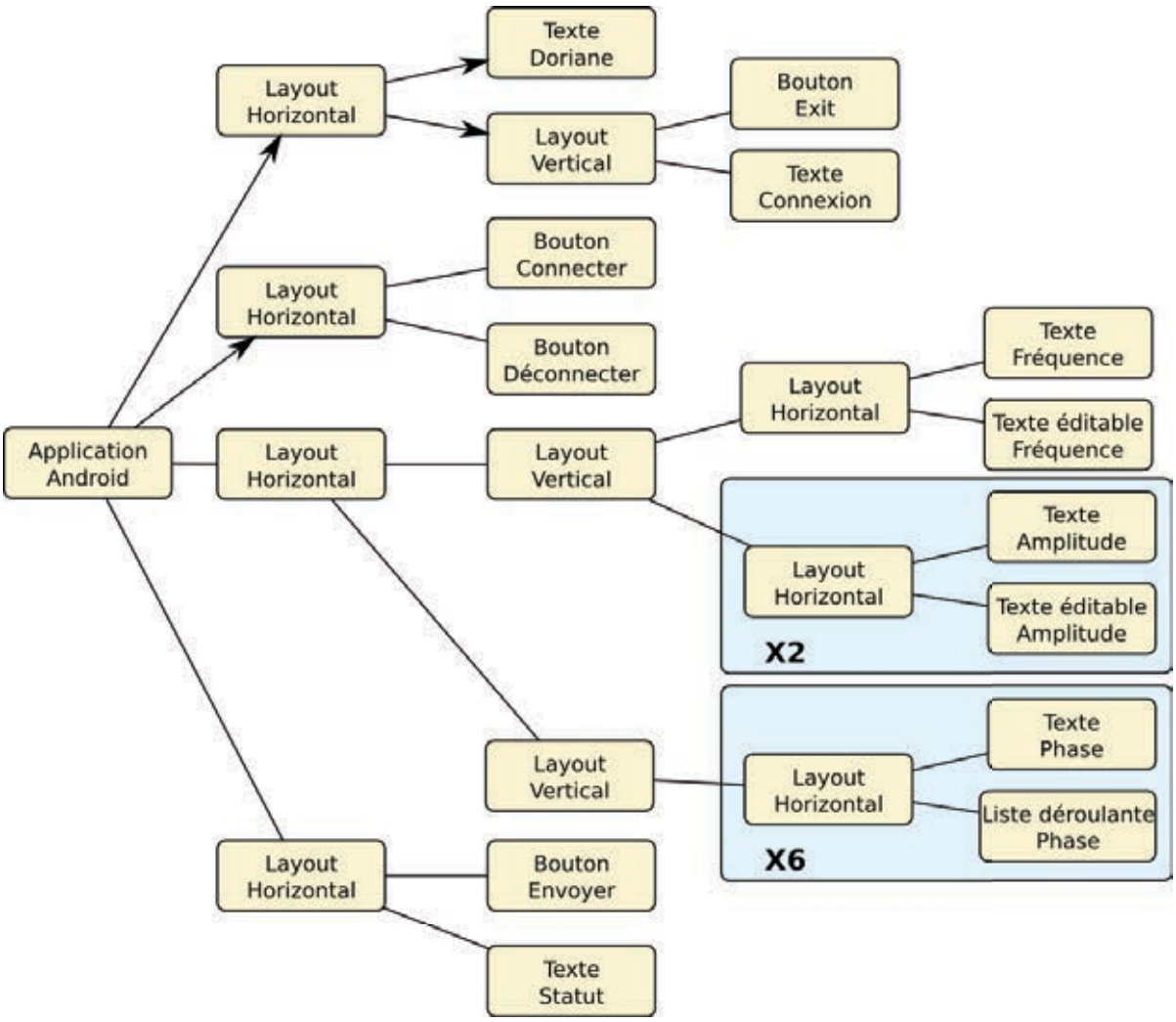
NE RIEN ECRIRE DANS CE CADRE

DR6 Diagramme de classes (Q33)



Tournez la page S.V.P.

DR7 (Q39)


Application Android
Layout Horizontal
Texte Doriane
Layout Vertical
Bouton Exit
Texte Connexion
Layout Horizontal
Bouton Connecter
Bouton Déconnecter
Layout Horizontal
Layout Vertical
Layout Horizontal
Texte Fréquence
Texte éditable Fréquence
Layout Horizontal
Texte Amplitude
Texte éditable Amplitude
X2
Layout Vertical
Layout Horizontal
Texte Phase
Liste déroulante Phase
X6
Layout Horizontal
Bouton Envoyer
Texte Statut