AEREX AVIONIQUE INC.

36, du Ruisseau, Suite 102, Breakeyville (Québec), Canada, G0S 1E2
Téléphone: (418) 832-1040, télécopieur: (418) 832-9831, courriel: info@aerex.ca

Rapport AEREX No: 2005-44061-1 version 1.1

Rapport RDDC Valcartier No: CR 2005-166

EXPERIMENTATION DE CONTROLEURS EN MONTAGE HWIL

RAPPORT FINAL

remis à

M. Eric Gagnon, Ph.D.
autorité scientifique
R & D pour la Défense Canada Valcartier
2459, boulevard Pie-XI nord
Val-Bélair (Québec)
Canada, G3J 1X5

dans le cadre du contrat TPSGC numéro
W7701-4-4061
(AMA W7701-032428/002/QCA)

rédigé par

François Dinel
et
Christian Rouleau

mars 2005

Sans classification

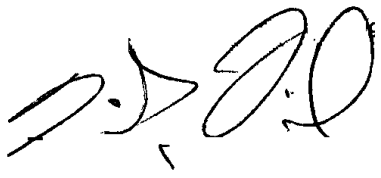
L'entrepreneur est entièrement responsable de la validité scientifique ou technique de ce rapport de contrat, et le contenu de ce rapport n'est pas nécessairement approuvé ni entériné par R et D pour la défense Canada.

Expérimentation de contrôleurs en montage HWIL

Rapport final

Rapport AEREX numéro: 2005-44061-1

Version: 1.1

Auteur et chargé de projet:  le 31 mars 2005

François Dinel, ing. jr., M. Sc.

Auteur et chargé de projet:  le 31 mars 2005

Christian Rouleau, B. Sc. A.

Approuvé pour publication:  le 31 mars 2005

Daniel Pomerleau, ing., Ph. D.

TABLE DES MATIÈRES

1. Introduction

L'objectif du projet est de réaliser une plate-forme expérimentale pour le développement d'algorithmes de contrôle et de guidage d'avions par un autopilote. La plate-forme que nous avons nommée HWIL (HardWare In Loop) est constituée de nombreux éléments électroniques et informatiques. L'autopilote est un MP2028g de Micropilot. Un ordinateur industriel roule un modèle d'avions 6DOF et un environnement aérien de la librairie Simulink Aerosim de Unmanned Dynamics. De plus, cet ordinateur est muni de cartes d'entrées/sorties analogiques et numériques de National Instruments pour réaliser l'interface matérielle avec l'autopilote. Ce projet est basé sur le projet de Myriam Manaï développé à l'Université Laval dans le cadre de son projet de maîtrise. Les travaux réalisés dans le cadre du présent contrat consistent à faire une réplique similaire à la plate-forme développée à l'Université Laval, en tenant compte des différences existantes entre le matériel utilisé dans les deux cas.

L'autopilote MP2028g de Micropilot est une carte électronique d'autopilote miniature disponible commercialement. Les capteurs directement présents sur la carte électronique sont: un capteur de pression statique (Motorola MPXA4115A), un capteur de pression dynamique (Motorola MPXV10GC), trois capteurs de vitesse angulaire (rate gyros) avec capteur de température (Analog Devices ADXRS150), deux accéléromètres à doubles axes (Analog Devices ADXL202E) et un GPS (Trimble Lassen SQ). Ces capteurs ont été enlevés de la carte d'autopilote pour pouvoir y injecter nos propres signaux, soit ceux provenant de l'ordinateur industriel par le biais des cartes d'interface.

L'ordinateur industriel utilisé est un Pentium IV HT cadencé à 3.08GHz et possédant 512Mo de mémoire. Cet ordinateur roule le système xPC Target développé par MathWorks. Deux cartes d'interface sont présentes dans cet ordinateurs: une carte PCI-6602 et une carte PCI-6703, toutes deux de National Instruments. La carte PCI-6602 est une carte de 8 compteurs/minuteries à haute vitesse de 32 bits alors que la carte PCI-6703 est une carte à 8 entrées/sorties analogiques. L'ordinateur industriel ainsi que les cartes d'interfaces simulent l'environnement atmosphérique et le comportement physique d'un avion 6DOF. Les actuateurs (commandes des servos) servent d'entrée au modèle alors que le modèle génère les informations de capteurs appropriées (capteurs de pression, de vitesse angulaire, etc.).

Le modèle d'avion et d'atmosphère de la librairie Aerosim de Unmanned Dynamics sont des modèles pour Simulink. Pour utiliser ces modèles dans le projet, un modèle Simulink d'interface avait été développé par l'Université Laval.

La Figure 1.A présente en schéma bloc l'architecture HWIL.

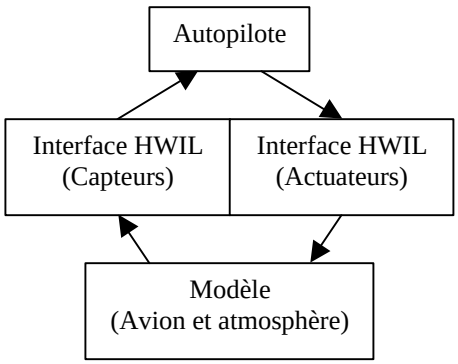


Figure 1.A Schéma bloc de l'architecture HWIL

Dans les chapitres qui suivent, les différences entre l'architecture HWIL originale développée à l'Université Laval et l'architecture répliquée pour RDDC Valcartier seront identifiées. Puis, les modifications apportées pour répondre au nouveau matériel utilisé sont décrites. Les améliorations apportées à l'architecture sont expliquées. Enfin, des suggestions de travaux futurs sont données.

2. Différences entre l'architecture HWIL initiale et celle du présent projet

Le principal mandat du présent projet est de réaliser une réplique de l'architecture HWIL développée à l'Université Laval. Pour ce faire, une étude des différences entre l'architecture de l'Université Laval et le matériel disponible/acheté au RDDC Valcartier a été réalisée. Dans cette étude ont été constatées des différences au niveau de l'autopilote et au niveau des cartes d'interfaces.

2.1. Autopilote

L'autopilote utilisé dans le design original était un MP2000, lequel comporte des différences mineures par rapport à l'autopilote utilisé dans l'architecture développée pour RDDC Valcartier.

2.1.1. Capteurs de vitesse angulaire (rate gyros)

Les différences rencontrées dans l'autopilote MP2028g par rapport au MP2000 sont d'abord situées au niveau des gyroscopes. Les nouveaux gyroscopes sont des capteurs de vitesse angulaire ADXRS150 de Analog Devices. Ils sont environ 10 fois plus sensibles que ceux présents sur le MP2000.

2.1.2. Pattes de test des gyroscopes et des accéléromètres

Les accéléromètres et les gyroscopes sont munis de pattes de « self-test », lesquelles permettent de confirmer le bon fonctionnement électronique des composantes en question lors de leur activation. Les accéléromètres comprennent 2 pattes de test alors que les gyroscopes n'en comprennent qu'une seule. Les pattes de test changent le comportement des capteurs et génèrent une sortie différente selon ces signaux. La patte ST1 des gyroscopes change la sortie RATEOUT de -0.66V alors que la patte ST2 modifie la sortie de +0.66V. Dans le cas des accéléromètres, la patte de test cause, lorsque activée, une baisse de 10% dans le cycle d'activité (duty cycle) à la sortie.

Les pattes de test ST1 de tous les gyroscopes ainsi que la patte ST de tous les accéléromètres sont connectée ensembles alors que toutes les pattes ST2 des gyroscopes sont connectée ensemble. Ces pattes n'étaient pas utilisées pour les accéléromètres du MP2000 alors que les gyroscopes du MP2000 n'en avaient tout simplement pas. Lors de l'initialisation du MP2028g, les pattes de test des accéléromètres et la patte de test des gyroscopes sont activées dans un patron connu pour vérifier le bon fonctionnement de ces composantes. Le patron de test consiste d'abord à tester les gyroscopes en activant la patte ST1 et la patte ST2 en alternance 3 fois chacune pendant 1 seconde. Puis, la séquence de test vérifie le bon fonctionnement des accéléromètres en activant ST1 30 fois pour de courtes périodes.

2.2. Cartes d'interface

Les cartes d'interfaces utilisées dans l'architecture originale étaient deux cartes PCI-6601 et une carte PCI-6703. Puisque la carte PCI-6602 est une version améliorée comportant plus de canaux que la carte PCI-6601, une seule carte PCI-6602 a été achetée pour remplacer les deux cartes PCI-6601. Notons une différence de fréquence d'horloge dans les compteurs/minuteries des cartes PCI-6602 (80MHz) et PCI-6601 (20MHz).

3. Modifications et améliorations apportées à l'architecture HWIL

Pour parer aux différences entre l'architecture originale et celle du RDDC Valcartier, nous avons

dû apporter des modifications au design de l'Université Laval. Les sections suivantes présentent les changements apportés au matériel et au logiciel de l'architecture HWIL.

3.1. Modifications au matériel

3.1.1. Cartes d'interface

La première modification nécessaire pour réaliser l'architecture HWIL était de brancher les signaux aux cartes d'interfaces. Plutôt que d'avoir les signaux étendus sur 3 cartes (2xPCI-6601 + 1xPCI-6703), ceux-ci étaient étendus sur 2 cartes seulement (1xPCI-6602 + 1xPCI-6703).

3.1.2. Carte de conditionnement de signaux

Les signaux de pression statique et dynamique étant étendus sur une plage de ±10V, il faut effectuer une amplification et un conditionnement de signal pour les rendre en format nécessaire à l'autopilote. Pour ce faire, la carte de conditionnement de signal conçue par l'Université Laval a été reprise avec quelques modifications (ajout de filtres) et gravée sur place au RDDC Valcartier.

3.1.3. Source d'alimentation

Pour alimenter toute l'électronique de l'architecture (autopilote et carte de conditionnement de signaux), une source d'alimentation électrique multi-sorties a été intégrée à celle-ci. La source utilisée est une HTAA-16W-A de Power One. Cette source offre 3 sorties différentes, soit une sortie double à ±15V/0.4A et une sortie de +5V/2A.

3.1.4. Montage sur plaque en plexiglas

Pour une meilleure visibilité et une protection de l'électronique, les différentes plaquettes électroniques ont été fixées à une plaque en plexiglas, laquelle a été fixée au châssis de l'ordinateur industriel.

3.1.5. Ajout de signaux électroniques

Pour parer à l'ajout des séquences de tests sur l'autopilote, les signaux de tests ont été ajoutés et sont branchés sur les cartes d'interfaces sur des pattes d'entrées/sorties génériques.

Le signal de capteur d'altitude à ultrason (AGL) a aussi été ajouté à l'architecture. Ce signal est une modulation en largeur d'impulsion (PWM) qui a été modélisé grossièrement en laboratoire. La fonction de transfert associée à ce capteur est la suivante: $PWM_s = 0.001(5.96Alt_m + 0.11)$. Ce capteur est développé par Micropilot mais une feuille technique n'est pas disponible publiquement. Il a été branché sur un canal libre la carte PCI-6602.

Enfin, le signal de température étant disponible et n'étant pas complexe, il a aussi été ajouté à l'architecture, branché à un canal libre de la carte PCI-6703.

3.2. Modifications au logiciel

3.2.1. Nouveau bloc Simulink de gyroscope

Pour parer au nouveau type de capteurs de vitesse angulaire, un nouveau bloc Simulink a été créé en se basant sur la feuille technique de la pièce ADXRS150. De plus, ce nouveau bloc tient compte des signaux de tests pour y réagir conformément aux spécifications de la pièce lors du mode de test.

3.2.2. Modification du bloc Simulink d'accéléromètre

Le bloc Simulink original des accéléromètres ADXL202E a été modifié pour tenir compte des signaux de tests. Il réagit maintenant conformément à ces signaux.

3.2.3. Ajout d'un bloc Simulink pour le capteur d'altitude à ultrason (AGL)

Le capteur à ultrason a été modélisé et intégré dans un bloc Simulink. Ce bloc génère les bons signaux en modulation de largeur d'impulsion pour répliquer ce capteur.

3.2.4. Ajout d'un bloc d'interface avec le simulateur de vol X-Plane

Afin d'ajouter une valeur visuelle, le simulateur de vol X-Plane à été intégré à l'architecture. Ce simulateur affiche un avion en vol et il est possible de lui envoyer des données pour faire déplacer l'avion. Dans l'architecture du RDDC Valcartier, X-Plane sert de visualisateur de l'avion simulé. Il permet de voir en 3D le comportement de cet avion lors du vol ce qui facilite grandement l'analyse. Pour utiliser X-Plane dans cette architecture, un bloc Simulink de création de paquet réseau a été réalisé. Ce bloc converti les données de vol en paquet de données compatible avec X-Plane et ce paquet est transmit à un autre ordinateur roulant X-Plane par le biais d'un bloc de transmission UDP de xPC Target.

4. Procédures d'utilisation de l'architecture

Pour utiliser l'architecture HWIL, il est préférable de suivre une certaine procédure pour le démarrage et l'arrêt des diverses composantes. Ce chapitre présente brièvement la procédure à utiliser dans ces deux cas. Notons que lors de la modification du modèle, la procédure de démarrage doit être reprise à partir de l'étape 10.

4.1. Démarrage

1. Allumage de toutes les composantes électroniques sauf de l'autopilote
2. Démarrage de MatLab
3. Ajout au PATH des différents répertoires contenant des composantes du modèle (ex: librairie Aerosim, librairie personnelle, etc.)
4. Initialisation des variables nécessaires au modèle par le script MatLab « aerosondeconfigR »
5. Compilation par la commande mex des différentes S-Functions nécessaires au projet (inversesingle, padding6, RemoveIrregularPoints, dionipci660x et XPlane_VEHA+XPlane_VEHA_wrapper)
6. Ouverture du modèle aerosondecompletr.mdl dans simulink
7. Configuration du bloc UDP Send pour contenir l'adresse IP de l'ordinateur roulant X-Plane
8. Démarrage de X-Plane sur l'ordinateur de visualisation
9. Configuration de X-Plane pour recevoir les données de vol de l'ordinateur industriel (inscrire l'adresse IP de l'ordinateur industriel dans le panneau de configuration Data Inputs/Outputs de X-Plane)
10. Compilation du modèle par RTW pour envoi sur l'ordinateur industriel
11. Activation de la connexion MatLab avec l'ordinateur (choisir le mode d'exécution « external » pour le modèle et cliquer sur le bouton de connexion)
12. Initialisation de la variable On/Off à 0
13. Démarrage de la simulation dans MatLab: l'affichage de X-Plane devrait déplacer l'avion au lieu d'initialisation du modèle

14. Démarrer l'autopilote et observer les signaux de servos. Lorsqu'ils apparaissent, c'est que l'autopilote est prêt à recevoir des communications
15. Démarrer Horizon GCS et créer une connexion avec l'autopilote
16. Attendre que les cases rouges (Ready, Fixs et Waiting for GPS to Lock) deviennent vertes: autopilote est maintenant initialisé et prêt à être volé
17. Cliquer sur « Arm », puis sur « Takeoff », ce qui indique au GCS qu'il sera normal que l'avion quitte le sol.
18. Changement de la variable On/Off du modèle à 1: toutes les composantes de l'architecture HWIL sont maintenant démarrées et la simulation roule

4.2. Arrêt

1. Peu importe l'ordre: fermeture de X-Plane et déconnexion/fermeture de Horizon GCS
2. Éteindre l'autopilote
3. Stopper la simulation MatLab
4. Éteindre toutes les composantes électroniques
5. Peu importe l'ordre: fermeture de l'ordinateur industriel et fermeture de MatLab

5. Suggestions de travaux futurs

À la suite du présent projet, plusieurs travaux pourront être envisagés. Ce chapitre décrit quelques travaux pouvant être réalisés dans les suites de ce projet.

- Intégration d'un ordinateur embarqué permettant la modification et la création d'algorithmes complexes de guidage
- Étude de la possibilité de modifier le programme contenu dans l'autopilote pour plus de flexibilité
- Intégration de plusieurs systèmes HWIL dans une même simulation, permettant les travaux de guidages interdépendants entre les différents véhicules

6. Avenues possibles pour l'intégration d'un ordinateur embarqué

Dans le cadre des travaux futurs, un ordinateur embarqué de type PCI-104 sera intégré à l'architecture pour permettre le développement d'algorithmes de guidages évolués. L'intégration de cet ordinateur peut-être réalisé différemment selon les objectifs recherchés et les ressources disponibles. Ce chapitre tente de présenter plusieurs alternatives possibles de développement pour permettre de faire un meilleur choix sur l'avenue à prioriser.

L'ordinateur embarqué PCI-104 est un Cool RoadRunner 4 de Lippert. Il s'agit d'une carte comprenant un processeur Pentium M de Intel cadencé à un maximum de 2GHz. La carte comprend aussi la plupart des interfaces standard d'un ordinateur PC de bureau. Une carte ajoutant 4 ports série au système est nécessaire pour la communication avec l'autopilote puisque le Cool RoadRunner 4 n'en comporte pas.

6.1. Spécifications à rencontrer

Lors de l'intégration du Cool RoadRunner 4 à l'architecture, cette partie du système devra répondre aux spécifications suivantes:

- L'ordinateur embarqué doit servir de passerelle entre l'ordinateur au sol roulant l'application de Ground Control Station (GCS), camouflant sa présence aux « yeux » du GCS
- L'ordinateur embarqué doit pouvoir rouler des algorithmes de guidage complexes et collaborer avec l'autopilote pour réaliser le guidage voulu
- L'ordinateur embarqué doit pouvoir utiliser la carte de ports série disponible, soit une carte Extreme 104 de ConnectTech
- L'ordinateur embarqué doit pouvoir être utilisé en vol, embarqué dans l'avion miniature du RDDC Valcartier

6.2. Alternatives de développement

Plusieurs alternatives s'offrent quant au développement du système embarqué PCI-104. Cette section décrit deux alternatives possibles. L'analyse de ces alternatives est reportée à la section suivante puisque cette section met l'emphase à décrire objectivement chaque alternative. De plus, ces alternatives ne sont pas les seules alternatives possibles et de nouvelles alternatives ou même des solutions hybrides sont envisageables.

6.2.1. Alternative 1: Une grande flexibilité

Une première solution met l'emphase sur la flexibilité de développement des algorithmes de guidage. Cette solution vise à utiliser un maximum de ressources et de langages de programmation pour développer les algorithmes de guidage désirés sans contrainte de développement. Pour ce faire, cette solution ne doit pas se retreindre à un seul environnement de développement, mais plutôt permettre la coopération de ces environnements. L'algorithme de guidage développé doit donc être une application en soi, ou être roulé par le biais d'une application spécifique très flexible.

Cette solution consiste à utiliser un système d'exploitation multi-processus roulant des applications spécifiquement conçues pour ce système. Par exemple, le système d'exploitation linux pourrait être utilisé. Une application pourrait être développée pour prendre en charge les communications entre le GCS et l'autopilote et il serait facile d'y intégrer aussi des commandes additionnelles pour contrôler le système embarqué directement. Cette application pourrait être développée en langage C ou C++ avec support pour les librairies dynamiques (.dll sous windows, .so sous linux). De cette façon, il serait possible de faire rouler l'application en permanence et d'y charger à volonté les algorithmes de guidage désirés qui serait contenues dans des librairies dynamiques.

6.2.2. Alternative 2: Un prototypage rapide

La deuxième solution proposée vise à réduire le temps mit à développer le système embarqué et ainsi pouvoir allouer plus de ressources au développement d'algorithmes de guidage, ou à développer d'autres aspects du projet.

Le système embarqué pourrait être bâti autour de xPC Target, lequel supporte directement l'implantation de modèles MatLab/Simulink. Le modèle implanté dans le système embarqué devra toujours contenir un certain squelette consistant en la partie nécessaire à la passerelle série entre le GCS et l'autopilote. Cette solution maintient tout de même une flexibilité puisqu'il est possible dans

MatLab/Simulink de programmer des S-Functions en plusieurs langages.

6.3. Analyse des alternatives

Les deux alternatives proposées présentent des avantages et des inconvénients. Cette section vise à comparer les deux alternatives pour aider à choisir la solution la plus appropriée. La comparaison est faite sous forme de tableau, lequel présente d'abord les spécifications, puis des critères d'analyse se voulant les plus variés possible. Le Tableau 6.1 ci-dessous présente la comparaison des 2 alternatives.

Caractéristiques	**Alternative 1**	**Alternative 2**
Passerelle GCS ⇔ Autopilote	Facile	Facile
Algorithmes complexes	Oui	Oui
Utilisation carte ConnectTech	Peu d'effort (pilote disponible)	Effort pour la création du bloc
Embarquement pour vol	Oui	Oui
Système d'exploitation	Linux, QNX, Windows	xPC Target
Type d'application	Application C/C++	Modèle compilé par RTW
Type de passerelle	Noyau de l'application	Squelette à inclure au modèle
Implantation d'algorithmes	Librairies dynamiques	Nouveau modèle
Communication série	Programmation de l'interface	Création d'un bloc d'interface
Flexibilité	Grande	Grande
Robustesse	Grande	Faible
Rapidité de prototypage	Faible	Grande
Coût (système embarqué)	Moyenne	Moyenne
Coût (algorithmes de guidage)	Moyenne	Faible
Performance	Grande	Faible

Tableau 6.1 Analyse des alternatives de développement du système embarqué

7. Conclusion

L'architecture HWIL présente à l'Université Laval a été répliquée avec succès au RDDC Valcartier, et certaines fonctionnalités y ont été ajoutées. Plusieurs modifications telles l'ajout de nouveaux signaux électroniques, l'installation d'un panneau de démonstration, l'ajout de nouveaux blocs au modèle et l'intégration d'un visualisateur à l'architecture améliore l'architecture du RDDC Valcartier par rapport à son prédécesseur.

L'architecture permet de simuler le vol d'un avion avec un autopilote matériel dans la boucle, ce qui permettra de développer des algorithmes de guidage plus près de la réalité puisque certaines composantes ne sont plus simulées mais bien réelles. De plus, puisque la partie matérielle du montage est indépendante de la partie simulée, il sera possible de tester ces algorithmes en vol à l'aide d'un avion miniature déjà disponible.

8. Références

[1] Manaï, M. Modélisation et identification du UAV: Rapport préliminaire No. 1. Janvier 2004.
[2] Manaï, M. Le système « hardware in the loop »: Rapport préliminaire No. 2. Décembre 2004.
[3] MicroPilot. Mp2028g Installation and Operation. Janvier 2004.
[4] Motorola. MPXA4115A: Semiconductor Technical Data. 2004.
[5] Motorola. MPXV10GC: Semiconductor Technical Data. 2001.
[6] Analog Devices. ADXRS150: Semiconductor Technical Data. 2003.
[7] Analog Devices. ADXL202E: Semiconductor Technical Data. 2000.

Annexe A : Configuration du modèle

Configuration parameters → Solver

- Stop time: inf
- Solver options: type: Fixed-step
- Solver options: Fixed-step size: 0.01

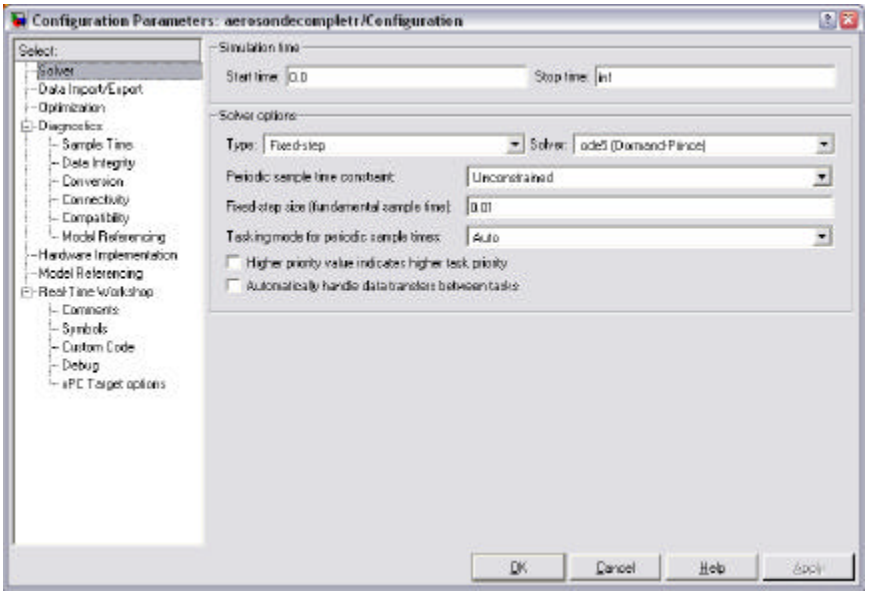
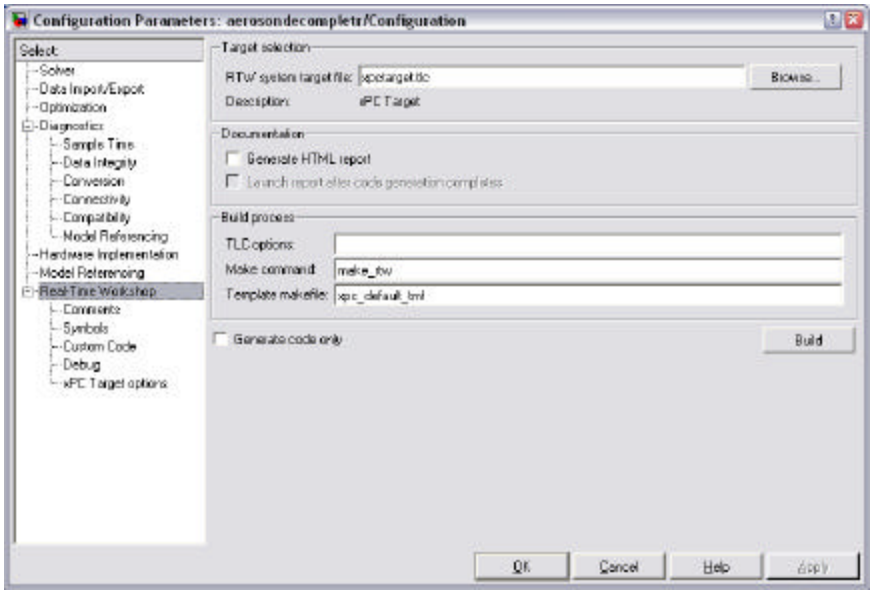


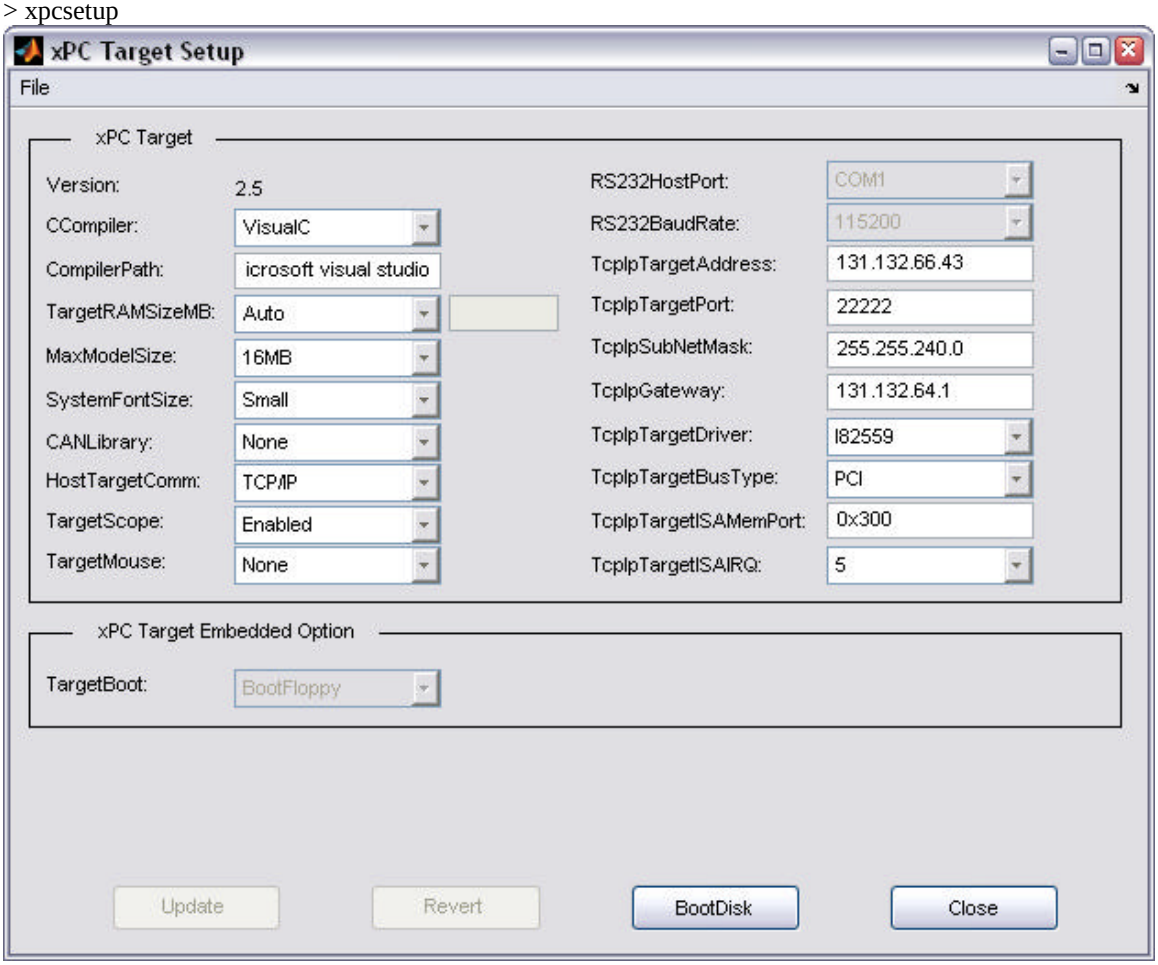
Figure A.1 Configuration du solveur pour usage avec xPC Target

Configuration parameters → Real-time Workshop:

- RTW system target file: xpctarget.tlc



> xpcsetup


xPC Target Setup
File
xPC Target
Version: 2.5
CCompiler: VisualC
CompilerPath: icrosoft visual studio
TargetRAMSizeMB: Auto
MaxModelSize: 16MB
SystemFontSize: Small
CANLibrary: None
HostTargetComm: TCP/IP
TargetScope: Enabled
TargetMouse: None
RS232HostPort: COM1
RS232BaudRate: 115200
TcpIpTargetAddress: 131.132.66.43
TcpIpTargetPort: 22222
TcpIpSubNetMask: 255.255.240.0
TcpIpGateway: 131.132.64.1
TcpIpTargetDriver: I82559
TcpIpTargetBusType: PCI
TcpIpTargetISAMemPort: 0x300
TcpIpTargetISAIRQ: 5
xPC Target Embedded Option
TargetBoot: BootFloppy
Update
Revert
BootDisk
Close

Annexe B : Correspondance technique

From: Paul Chambers [mailto:pchambers@MICROPILOT.com]
Sent: Thursday, March 31, 2005 12:34 PM
To: Rouleau, Christian (AEREX)
Subject: Description of P8 Connector

Hello Mr. Rouleau,

Attached is a description of the pins on the P8 connector. Pins on the side of the connector closest to the pressure sensors are A; the row of pins closer to the edge of the board is B. Pin one for each row is the pin closest to the P1 connector.

Best Regards,

Paul Chambers

Senior Technician
Micropilot
paul@micropilot.com
(204) 344-5558 Ext.227 - Office
(204) 292-2155 - Mobile

-----Original Message-----
From: Rouleau, Christian (AEREX) [mailto:Christian.Rouleau.aerex@drdc-rddc.gc.ca]
Sent: Wednesday, March 30, 2005 9:28 AM
To: Support
Subject:

Hello M. Wakeman,
I'm working for AEREX Avionics Inc and we bought a sensorless MP2028. For our interfacing needing, we need the 16 pins P8 connector pinout.
Thank you very much
Christian Rouleau
AEREX Avionics Inc.

Dinel, Francois (AEREX) wrote:

>   Creating library HWILAerosonde.lib and object HWILAerosonde.exp
>HWILAerosonde.obj : error LNK2001: unresolved external symbol _sfunwmm
>HWILAerosonde.dll : fatal error LNK1120: 1 unresolved externals NMAKE :
>fatal error U1077: 'link' : return code '0x460'
>Stop.
>
>
This is where your link error is. The "sfunwmm" S-function module has not been linked in. The source code for that module is available in the Aerosim Blockset sources directory. So you have two options - either find a way to add-in the source for this module when the code is compiled and linked, or if you don't need magnetic field output, create a version of the aircraft model which does not have the WMM-2000 magnetic field model block.

Thanks,
Marius

Hello Mr. Dinel,

Sorry for the delay in my response, but I had to discuss this problem with our hardware engineer, who is extremely busy working on the next generation of our autopilot hardware.

These failures that you are seeing are likely caused by the fact that you are not duplicating the self-test procedure that is performed by the autopilot when it is first powered up. You will need to recreate the test signals send by the gyros and accelerometers in response to the signals that the autopilot sends to each sensor to initiate the test sequence.

Accelerometers

The accelerometer test is conducted simultaneously on all three accelerometers. Begin by duplicating the pulsewidth that each accelerometer would generate when the board is leveled. Monitor the output of ST1. When the value on ST1 changes from 0 to 1, the signal from each accelerometer should change by 10% from the initial signal. Any change of the output between 7% and 15% of the resting pulsewidth will pass the test. When ST1 changes from 1 to 0, the outputs from the accelerometers return to their normal values. Thirty cycles of this test are run on each accelerometer.

Gyros

Similarly, the gyro test is performed on all three gyros simultaneously. Begin by generating the resting signal (2.5 volts) for each gyro. The gyros have two test pins, ST1 and ST2. When ST2=1 (and ST1=0) for a particular gyro, it must generate a voltage roughly 0.66 volts higher than the resting voltage (a range of +0.3 to +1.1 volts will pass the test; +0.66 is ideal). When ST1=1 (and ST2=0) for a particular gyro, it must generate a voltage roughly 0.66 volts lower than the resting voltage (a range of -0.3 to -1.1 volts will pass the test; -0.66 is ideal). Three positive and three negative tests are performed on each gyro.

I believe that you already have the information required to find the ST1 and ST2 pins that I refer to. In case you do not have these yet, I am attaching the spec sheets for the gyros and accelerometers. If there is other information that you are missing to recreate the signals generated by the attitude sensors during these tests, please do not hesitate to ask.

Best Regards,

Paul Chambers

Senior Technician
Micropilot Inc.
paul@micropilot.com
(204) 344-5558 Ext.227 - Office
(204) 292-2155 - Mobile

-----Original Message-----
From: Dinel, Francois (AEREX) [mailto:Francois.Dinel.AEREX@drdc-rddc.gc.ca]
Sent: Monday, April 11, 2005 4:10 PM
To: Paul Chambers
Cc: Rouleau, Christian (AEREX)
Subject: Sensorless MP2028g

Hi Mr. Chambers,

I'm Mr. Rouleau's collegue and I also work on the integration of a sensorless MP2028g in a simulation environment. We are currently facing a problem while injecting signals. We feed it "reasonable physical zeros" at initialization while we power on the autopilot. However, we get error messages like "Pitch gyro failed" and "X accelerometer failed".

We don't know what is causing those errors because we think our initialization signals are ok. The signals we are injecting at initialization are:
- All gyros (simulating ADXRS150): 2.5V signal with 2.5V reference
- Both accelerometers (simulating ADXL202E): 5V pwm signal at 720Hz (1.390ms period) with a 50% duty cycle

Do you know what is causing those error messages? Should special care be taken while injecting signals at initialization?

Thanks for your help!

François Dinel
|Computers & electronic specialist
|Aerex Avionics inc.
|fdinel@aerex.ca

Hello Dinel,

Can you check to see what the autopilot is reading for airspeed? This error generally appears when the current airspeed exceeds the maximum scheduled airspeed on one or more feedback loops. You could circumvent this by removing gain scheduling from all feedback loops (set all schedule ranges for each loop to zero) but this does not really solve the problem - it sounds like the airspeed is not reading correctly.

Have you adjusted any of the gains for navigation? What does the datalog show in regards to desired heading, current heading, desired roll, actual roll, etc?

Regards,

Paul Chambers
Senior Technician
Micropilot Inc.
paul@micropilot.com
(204) 344-5558 Ext.227 - Office
(204) 292-2155 - Mobile

-----Original Message-----
From: Dinel, Francois (AEREX) [mailto:Francois.Dinel.AEREX@drdc-rddc.gc.ca]
Sent: Thursday, April 14, 2005 3:10 PM
To: Paul Chambers
Subject: RE: Sensorless MP2028g

Hello,

Thanks for your precise answer. We are now facing another error message:
"Gain schedule error". Do you know what it means?

We are currently trying simulating a flight for the autopilot, but it only do a circle motion. It doesn't seem to respond to programmed trajectories, even though we are sure to program it with other ones.

Thanks!

François Dinel
|Computers & electronic specialist
|Aerex Avionics inc.
|fdinel@aerex.ca

Hello,

There are several things that must happen before you can enter RPV mode, whether in simulation or in actual flight:

1. The aircraft must be 'off ground' (check that the flight status indicator in the bottom left indicates 'FLY', not 'GND' - this transition usually takes place during or just after the takeoff).

2. The .fly file must contain a 'pattern gcsFailed' failure handler.

3. The .vrs file must contain a non-zero value for the 'GCS Fail Timeout'

Without these three conditions satisfied, Horizon will return an error (unfortunately not a very specific one). These safeties were put in place to prevent the possibility of someone flying out of range of the datalink while in RPV mode with no failsafe in place.

I a awaiting some input from the software development department concerning your licensing question, and I will get back to you as soon as I can.

Regards,

Paul Chambers
Senior Technician
Micropilot Inc.
paul@micropilot.com
(204) 344-5558 Ext.227 - Office
(204) 292-2155 - Mobile

-----Original Message-----
From: Dinel, Francois (AEREX)
[mailto:Francois.Dinel.AEREX@drdc-rddc.gc.ca]
Sent: Monday, May 02, 2005 4:47 PM
To: Paul Chambers
Cc: Gagnon, Eric; 'fdinel@aerex.ca'
Subject: Troubles using Horizon GCS...

Hi again,

we are trying to use the Horizon GCS with RPV mode but each time we click on the "UAV" button (which is supposed to switch UAV/RPV mode), we get an error message saying "invalid command". Is there something we could be doing wrong? Do we need to take care with the electronic for this to work? The Autopilot/RemotePilot wire is currently floating but we also tried wiring it to ground and VCC without success...

Thanks!

Francois Dinel
computer & electronic specialist
Aerex Avionics inc.
fdinel@aerex.ca

Hello Mr. Dinel,

What directory do you run your custom GCS program from? For it to work properly, your program must be in one of these following directories (depending on what versions of Xtender and Horizon you have installed):

C:\Program Files\Micropilot\Horizon3

C:\Program Files\Micropilot\Horizon3.1

C:\Program Files\Micropilot\Xtender3\bin

C:\Program Files\Micropilot\Xtender3.1\bin

Please verify that the full path of your program's current directory corresponds to one of the above.

Best Regards,

Paul Chambers

> -----Original Message-----
> From: Dinel, Francois (AEREX) [mailto:Francois.Dinel.AEREX@drdc-rddc.gc.ca]
> Sent: Mon 5/2/2005 12:21 PM
> To: Paul Chambers
> Cc: Gagnon, Eric; 'fdinel@aerex.ca'
> Subject: xTender License Key...
>
> Hello M. Chambers,
>
> we're trying to develop a custom GCS with xTender, but some calls (eg: mpInitLink) to the library ends up returning an MPERR_AUTH_FAILED error code. However, when we launch the license manager application for xTender, it warns us that we currently have a valid license file and that we should not change it... Do you know about this problem and what could be causing it?
>
> I also noticed that every call to the library takes a long time to complete (more than 5-10 seconds). Is this normal?
>
> Our current application requires that we put a companion computer aboard our development plane. We ended up with a design that includes a PC104 that would act as a bridge between the autopilot and the Horizon GCS, apart from executing special hardware and software control & monitoring. However, we clearly don't want the PC104 to run Windows as it will need realtime scheduling. The current xTender version we have seem only to be compatible with windows targets. If that's really the case, it would be preferable for us to communicate directly with the MP2028g autopilot by RS232 port so we would need the autopilot protocol definition. Is it possible for you to share the protocol with us?
>
> Thanks for your help!
>
> Francois Dinel
> computers & electronics specialist
> Aerex Avionics inc.
> fdinel@aerex.ca

Hello Mr.Dinel,

There is no easy way to run your custom GCS from another location. That said, it is possible and if this is something that you absolutely require we should be able to instruct you on how do it. You would have to copy the license, as well as a number of other files, and also setup the license program to recognize the new location as valid.

In regards to communication protocol, this information is in chapter 5 of the Xtender manual. If your Xtender manual is outdated and does not contain this information, please let me know and we can mail you an updated version (sorry - we do not send out electronic copies of Xtender documentation).

Best Regards,

Paul Chambers
Senior Technician
Micropilot Inc.
paul@micropilot.com
(204) 344-5558 Ext.227 - Office
(204) 292-2155 - Mobile

-----Original Message-----
From: Dinel, Francois (AEREX) [mailto:Francois.Dinel.AEREX@drdc-rddc.gc.ca]
Sent: May 4, 2005 9:50 AM
To: Paul Chambers
Subject: RE: xTender License Key...

You are right, we aren't running our custom GCS in one of those. I'll try it!

Is there a way not to have to put it in the same directory as those? For example by copying a file (license?) to our custom GCS directory or something like this? It would be cleaner for us to keep our work in a directory seperated from Micropilot work...

And what about sharing the communication specs (protocol) with us for our custom development needs? (as I asked in my mail below...)

Thanks!

François Dinel
|Computers & electronic specialist
|Aerex Avionics inc.
|fdinel@aerex.ca

Hello Mr. Dinel,

In regards to your question on how to run your custom GCS from a different location, there are instructions on how to do this included with Xtender. You can find them in one of the following locations:

For XTENDER 3.0
c:\program files\micropilot\xtender3\redist\redist.txt

For XTENDER 3.1
c:\program files\micropilot\xtender3.1\redist\redist.txt

I have also attached a copy of this file for your convenience.

Best Regards,

Paul Chambers
Senior Technician
Micropilot Inc.
paul@micropilot.com
(204) 344-5558 Ext.227 - Office
(204) 292-2155 - Mobile