IUT ANNECY UNIVERSITÉ SAVOIE MONT BLANC

Remise à niveau

TP

v0

IUT d'Annecy, 9 rue de l'Arc en Ciel, 74940 Annecy

TP – Tri de pièces virtuel

0.1 Communication CS99

Dans cette section, nous allons créer les sous-programmes en charge de la communication avec la baie CS9. On propose, pour cela, d'ajouter deux sous-programmes à notre projet avec les noms et rôles suivants :

1. Une section `ReadEthIpRobot`, **première section à être exécutée**, elle aura pour rôle :
   (a) Lecture des données du robot via le bloc fonctionnel `VAL_ReadAxesGroup`
   (b) Lecture des informations depuis l'IHM via le bloc fonctionnel `BF_ReadHmi`
2. Une section `WriteEthIpRobot`, **dernière section à être exécutée**, elle aura pour rôle :
   (c) Écriture des informations vers l'IHM via le bloc fonctionnel `BF_WriteHmi`
   (d) Écriture des données du robot via le bloc fonctionnel `VAL_WriteAxesGroup`

Les blocs fonctionnels `VAL_ReadAxesGroup` et `VAL_WriteAxesGroup` sont fournis par la bibliothèque `UnivalPlc` et sont donc déjà disponibles. En revanche, les blocs fonctionnels `BF_ReadHmi` et `BF_WriteHmi` ont été créés par l'équipe enseignante et doivent être importés dans le projet.

Pour cela, ce TP suivra les étapes suivantes :

1. Importation des blocs fonctionnels nécessaires (HMI et CS9)
2. Réservation des variables
3. Instanciation des blocs fonctionnels
4. Création des sections `ReadEthIpRobot` et `WriteEthIpRobot`
   - Appel des blocs de lecture et d'écriture dans les sections respectives.
5. Vérification du bon fonctionnement de la communication

Manipulation 1 : Importation des blocs fonctionnels HMI

Étape 1 **Clique-droit->Importer** sur le dossier `Type FB dérivés` dans l'arborescence du projet.

Étape 2 Sélectionner les fichiers `FB_ReadHmi.xdb` et `FB_WriteHmi.xdb` dans le dossier `U:\Documents\BUT\GEII\ModulesS5\Automatisme_Pour_Robotique\Scara`

Étape 3 Si l'application vous le demande, **Garder tout** puis valider.

Manipulation 2 : Communication CS99

Étape 4 Créer les sous-programmes décrits ci-dessus et les ordonner afin qu'ils soient exécutés dans l'ordre voulu.

```
TYPE T_StaeubliRobot :
    STRUCT
        Status : T_Status ;
        Command : T_Command ;
        CommandInterface : T_CommandInterface ;
    END_STRUCT
END_TYPE


TYPE T_Status:
    STRUCT
        Initialized: BOOL ; (* True = The library is initialized *)
        Online: BOOL ; (* True = robot can receive commands *)
        ErrorPending: BOOL ; (* True = an error is pending *)
        IsMoving: BOOL ; (* True = robot is moving *)
        EStopActive: BOOL ; (* True = E-stop is pending *)
        DummyPlug: BOOL ; (* True = staubli teach pendant replaced by dummy
        plug *)
        ExternalMcp_Wms: BOOL ; (* True = Robot is properly configured to use both
        user- supplied WMS and Teach Pendant. *)
        ActualSpeed: REAL ; (* Cartesian speed of the current TCP *)
        ActualOverride: REAL ; (* Current override value [0.01 .. 100] *)
        ActualErrorNumber: UINT ; (* Total number of pending errors in robot *)
        ActualOperationMode: UINT ; (* Actual working mode 0= invalid , 1=manu,
        3=Auto, 4=remote(extaut) *)
        ActualErrorID: T_PendingErrors ; (* List of pending error on server side *)
        CartesianPos: T_CartesianPos ; (* Current cartesian position *)
        JointPos: T_JointPos ; (* Current joint position *)
        ActualCoordSystem: UINT ; (* Number of the user frame in which the position of
        the Tool Center Point is reported *)
        ActualTool: UINT ; (* Number of the Tool Center Point for which the
        position is reported *)
        RobotStateMachine: UINT ; (* State Machine of the robot *)
        MovementID: INT ; (* Identifier of motion currently executed *)
        MovementProgress: INT ; (* Percentage of actual movement that has been
        completed *)
        RobotModel: INT (* Robot model connected to controller *)
        ControllerModel: UINT ; (* Staubli controller 8=CS8C / 9=CS9 *)
        CS9Safety: T_CS9SftyFbk ; (* CS9 Only - Safety features *)
        Heartbeat: UINT ;
        ServerMajorVersion: UINT ; (* Major version of unival PLC server *)
        ServerMinorVersion: UINT ; (* Minor version of unival PLC server *)
        ServerEdit: UINT ; (* Edit of the uniVAL plc server *)
        ClientMajorVersion: UINT ; (* Major version of unival PLC client library *)
        ClientMinorVersion: UINT ; (* Minor version of unival PLC client library *)
        ClientEdit: UINT ; (* Edit of the unival PLC client library *)
    END_STRUCT
END_TYPE

TYPE T_Command :
    STRUCT
        EnableVerbose : BOOL ; (* TRUE=Enable tracing activity of the server. It is
        strongly recommended to enable trace for debugging
        purpose only. Starting with FALSE *)
        OverrideCmd : UINT ; (* Commanded Override [1..100] (monitor speed) Starting
        with 0 *)
        OperationModeCmd : UINT ; (* Commanded operation mode 0= invalid , 1= Manu,
        3=Auto, 4 remote(extAut) *)
        ToolCmd : UINT ; (* Select the Tool used for movement. Starting with 0*)
        CoordSystemCmd : UINT ; (* Select the coordinate system used for movement.
        Starting with 0 *)
        CS9Safety : T_CS9SftyCmd ; (* CS9 Only - Safety features *)
        LifebitPeriod : TIME ; (* Select the period of the internal lifebit
        (100ms<Period<1000ms) . starting with t#200ms *)
    END_STRUCT
END_TYPE
```