



Documentation: Informatique



Programme principal : Coupe 2013

Cette documentation a pour but de présenter le programme informatique mis en place dans le robot pour la coupe de France de robotique 2013

I. Présentation Générale

Le projet de dev Oufff2013 est composé de la manière suivante :

Oufff2013\	: Racine du projet
- Conception	: Dossier contenant les fichiers de conception
- Projets	: Dossier contenant les environnements des IDE
- MPLAB	: Environnement de dev de MPLAB
- VC2011	: Environnement de dev de MS VC2011
- src	: Code source
- BSP_v1	: Code relatif au BSP (code 2012)
- BSP_v2	: Code relatif au BSP (code 2013)
- DevCard	: Code relatif à la carte de dev (PIC32MX440H)
- Main	: Code relatif au programme principal
- Strategy	: Code relatif à toutes les stratégies du robot
- Microchip	: Code MPLAB et/ou µC-OSII
- uC-CPU	: Code MPLAB et/ou µC-OSII
- uC-LIB	: Code MPLAB et/ou µC-OSII
- uCOS-II	: Code MPLAB et/ou µC-OSII

Tous les projets contenus dans le répertoire « Projets » font appels aux fichiers du répertoire « src ». Seules les constantes d'environnement du projet peuvent varier d'un projet à l'autre. Dans la suite de ce document, nous allons nous concentrer sur les fichiers contenus dans le répertoire « src/Main ».



Documentation:

Informatique



Présentation du contenu du répertoire « src/Main » :

src\Main\	
- Strategy\	: Répertoire contenant les fichiers des stratégies
- StrategyColorA.c	: Stratégie pour la couleur bleue
- StrategyColorB.c	: Stratégie pour la couleur rouge
- PatternColorA.spattern	: Fichier modèle bleu pour StrategyGenerator V2
- PatternColorB.spattern	: Fichier modèle rouge pour StrategyGeneratorV2
- App.c	: Init et démarrage de toutes les tâches
- App.h	: Déclaration de toutes les variables globales
- AppCustomTypes.h	: Définition des types personnalisés
- AppGlobalConfig.h	: Définition des constantes de config communes
- AppGlobalFunc.*	: Fonctions communes à toutes les tâches
- AppGlobalVars.h	: Permet l'include de toutes les variables globales
- AppHooks.c	: Fichier contenant les actions pour le hook
- AppIncludes.h	: Contient tous les includes nécessaires du projet
- AppPrimaryConfig.h	: Définition des constantes de config du 1 ^{er} robot
- AppSecondaryConfig.h	: Définition des constantes de config du 2 ^e robot
- ArmsControl.*	: Fonctions pour le contrôle des bras
- HoopsControl.*	: Fonctions pour le contrôle de la pince
- LibMoving.*	: Lib de mouvement
- os_cfg.h	: Configuration de l'OS
- Strategy.*	: Déclaration des fonctions haut-niveau
- TaskAsser.*	: Code de la tâche Asservissement
- TaskDebug.*	: Code de la tâche Debug
- TaskMain.*	: Code de la tâche Principale
- TaskMvt.*	: Code de la tâche Mouvement
- TaskOdo.*	: Code de la tâche Odométrie
- TaskSensors.*	: Code de la tâche gérant tous les capteurs
- TaskTempo.*	: Code de la tâche gérant le temps
- TaskTest.*	: Code de la tâche Test
- TurbineControl.*	: Fonctions pour le contrôle des turbines
- XBEEControl.*	: Fonctions pour le contrôle des XBEE

NB : Les fichiers notés « NomDuFichier.* » symbolise les fichiers « NomDuFichier.c » et « NomDuFichier.h »



II. Définition des types personnalisés

a) Les Enumérations

EnumColor	Donne la couleur
Valeurs Possibles	
c_NotSet	Couleur non définie
c_ColorA	Couleur A. Pour la coupe 2013 c'est du bleu
c_ColorB	Couleur B. Pour la coupe 2013 c'est du rouge

EnumCmdType	Définit le type de chaque commande
Valeurs Possibles	
CmdType_NotSet	Type de commande non définie
CmdType_Blocking	La commande est bloquante
CmdType_NonBlocking	La commande est non bloquante

EnumCmd	Définit le type de la commande à passer entre toutes les tâches	
Valeurs Possibles		
NotSet	0	Commande non définie
Mvt_UseAngleOnly	10	Effectuer un mouvement avec le mode d'asservissement 1
Mvt_UseDistOnly	11	Effectuer un mouvement avec le mode d'asservissement 2
Mvt_UseMixedMode	12	Effectuer un mouvement avec le mode d'asservissement 3
Mvt_UsePivotMode	13	Effectuer un mouvement avec le mode d'asservissement 4
Mvt_UseSpline	14	Effectuer un mouvement en spline
MvtSimple_MoveInMM	15	Mvt simple (pas de division) : Aller en ligne droite
MvtSimple_RotateInDeg	16	Mvt simple (pas de division) : Tourner d'un angle en degré
MvtSimple_RotateToAngleInDeg	17	Mvt simple (pas de division) : Tourner pour se mettre dans un angle précis par rapport au repère du terrain
Mvt_Stop	18	Arrêter l'action en cours
App_Wait	30	Attendre
App_IfGoto_System	31	Test une condition système et agit en conséquence
App_IfGoto_Strategy	32	Test une condition stratégique et agit en conséquence
App_SetNewPos	33	Modifier la position du robot
App_SetStrategyFlags	34	Permet de modifier un flag de stratégie
Sensors_SetHoopLevel	40	Permet de lever ou abaisser le bras
Sensors_SetArmsStatus	41	Permet de définir le statut des bras (à l'avant)



b) Les Structures

StructPos		Structure utilisée par la TaskOdo pour donner la position actuelle du robot. Cette structure est aussi utilisée pour envoyer les ordres à la TaskAsser.
Contenu		
float	x	Position en x du robot
float	y	Position en y du robot
float	angle	Angle que fait le robot par rapport à l'origine (suivant x)
CPU_INT16U	right_encoder	Incrément de la roue droite pour un mouvement en pivot
CPU_INT16U	Left_encoder	Incrément de la roue gauche pour un mouvement en pivot
unsigned char	CurrentState	Flag pour indiquer le statut du mouvement (arrêté / en mvt)
Utilisation		
Pas d'information complémentaire		

StructMsg		Structure utilisée pour passer des messages entre tâches		
Contenu				
BOOLEAN	IsRead	Flag utilisé pour la gestion des messages		
EnumCmd	Cmd	Commande à envoyer via le message		
EnumCmdType	CmdType	Type de la commande		
Utilisation				
int	Param1	Paramètre 1 de la commande		
int	Param2	Paramètre 2 de la commande		
int	Param3	Paramètre 3 de la commande		
int	Param4	Paramètre 4 de la commande		
Utilisation				
Cmd	Param1	Param2	Param3	Param4
Sensors_SetHoopLevel	Position de la pince : HOOP_LEVEL_UP HOOP_LEVEL_DOWN			
Sensors_SetArmsStatus	Etat du bras droit : ARM_OPEN ARM_CLOSED ARM_FRONT	Etat du bras gauche : ARM_OPEN ARM_CLOSED ARM_FRONT		



Documentation:

Informatique



StructCmd		Structure utilisée pour passer des commandes d'une tâche à une autre		
Contenu				
<code>EnumCmd</code>	<code>Cmd</code>	Commande à exécuter		
<code>int</code>	<code>Param1</code>	Paramètre 1 de la commande		
<code>float</code>	<code>Param2</code>	Paramètre 2 de la commande		
<code>float</code>	<code>Param3</code>	Paramètre 3 de la commande		
<code>float</code>	<code>Param4</code>	Paramètre 4 de la commande		
<code>unsigned int</code>	<code>ActiveSensors Flag</code>	Indique quels sont les capteurs à utiliser pour l'anticollision lors de la prochaine commande		
<code>EnumCmdType</code>	<code>CmdType</code>	Action bloquante ou non		
Utilisation				
<code>Cmd</code>	<code>Param1</code>	<code>Param2</code>	<code>Param3</code>	<code>Param4</code>
NotSet				
Mvt_UseAngleOnly	Vitesse			Angle en °
Mvt_UseDistOnly	Vitesse	x	y	
Mvt_UseMixedMode	Vitesse	x	y	Angle en °
Mvt_UsePivotMode	Vitesse	RIGHT_WHEEL LEFT_WHEEL		Angle en °
Mvt_UseSpline	Vitesse	x	y	Angle en °
Mvt_Simple_MoveInMM	Vitesse	Distance en mm		
Mvt_Simple_RotateInDeg	Vitesse			Angle en °
Mvt_Simple_RotateToAngle_InDeg	Vitesse			Angle en °
Mvt_Stop				
App_Wait	Heure	Minute	Seconde	Milliseconde
App_IfGoto_System	Condition (Mettre -1 pour un test toujours vrai)	Goto si vrai	Goto si faux	
App_IfGoto_Strategy	Condition (Mettre -1 pour un test toujours vrai)	Goto si vrai	Goto si faux	
App_SetNewPos		x	y	Angle en °
App_SetStrategyFlags	Flag	OS_TRUE OS_FALSE		
Sensors_SetHoopLevel	HOOP_LEVEL_UP HOOP_LEVEL_DOWN			
Sensors_SetArmsStatus	Bras Droit ARM_OPEN ARM_CLOSED ARM_FRONT	Bras Gauche ARM_OPEN ARM_CLOSED ARM_FRONT		

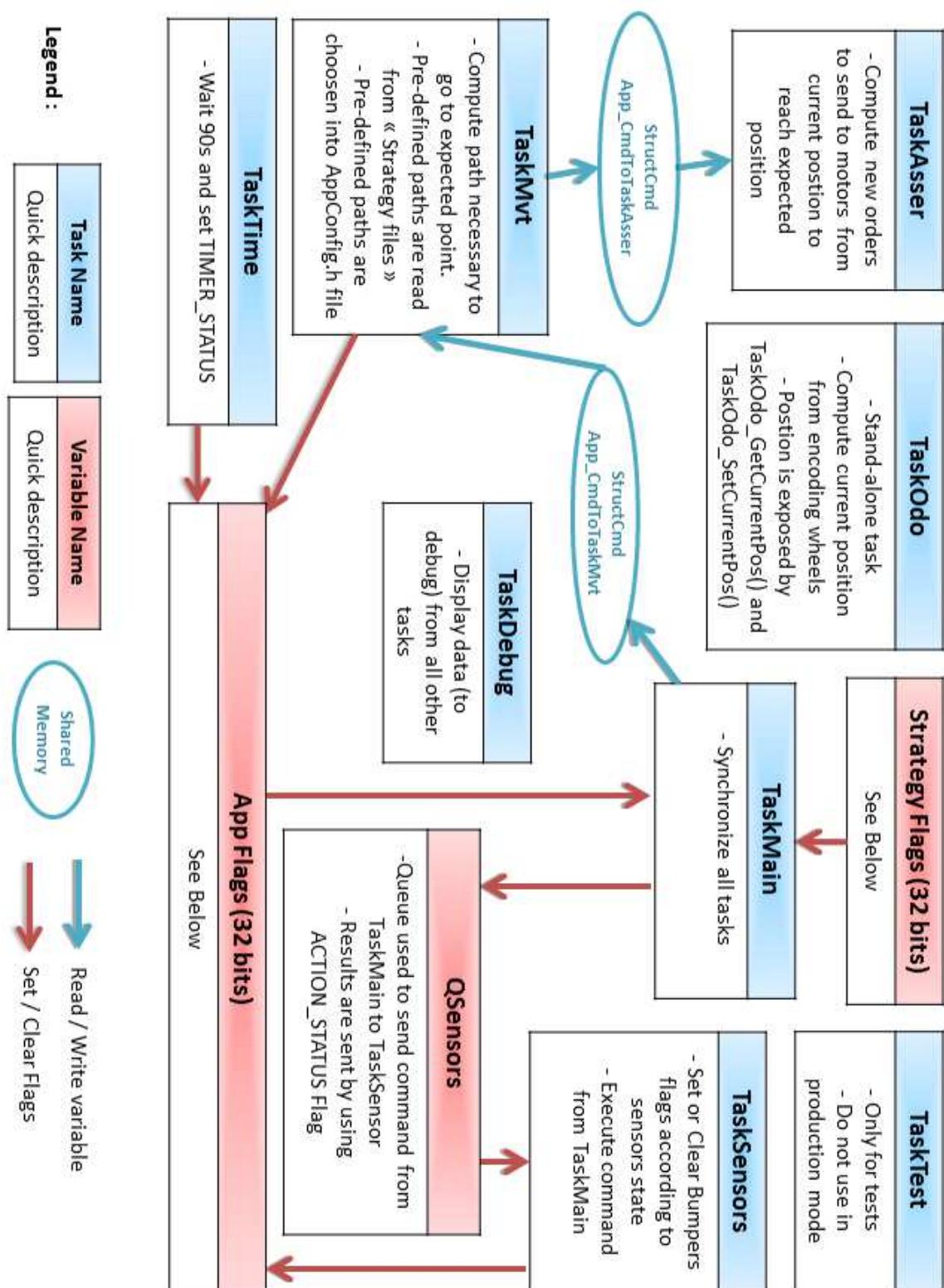


Documentation:

Informatique

III. Présentation Globale

Le programme se compose de la manière suivante :





Documentation:

Informatique



Flags Système : APP_PARAM_APPFLAG

Nom du Flag	Valeur	Description
Flag de fonctionnement		
APP_PARAM_APPFLAG_START_BUTTON	0x00000001	Etat de la tirette
APP_PARAM_APPFLAG_BIT01	0x00000002	Non utilisé
APP_PARAM_APPFLAG_BIT02	0x00000004	Non utilisé
APP_PARAM_APPFLAG_BIT03	0x00000008	Non utilisé
APP_PARAM_APPFLAG_BIT04	0x00000010	Non utilisé
APP_PARAM_APPFLAG_ACTION_TIMEOUT	0x00000020	Indique si la précédente action s'est terminée en timeout
APP_PARAM_APPFLAG_ACTION_STATUS	0x00000040	Indique l'état de l'action actuelle (terminée ou en cours)
APP_PARAM_APPFLAG_TIMER_STATUS	0x00000080	Indique si la fin du match est atteinte ou pas
Capteur GP2		
APP_PARAM_APPFLAG_GP2_FRONT	0x00000100	GP2 présent sur l'avant du robot
APP_PARAM_APPFLAG_GP2_INTERNAL_REAR	0x00000200	GP2 à l'intérieur de la pince
APP_PARAM_APPFLAG_GP2_REAR_HOOP	0x00000400	GP2 présent sur la pince
APP_PARAM_APPFLAG_GP2_REAR_LEFT_HOOP	0x00000800	GP2 sur la gauche de la pince (par rapport à l'avant du robot)
APP_PARAM_APPFLAG_GP2_REAR_RIGHT_HOOP	0x00001000	GP2 sur la droite de la pince (par rapport à l'avant du robot)
APP_PARAM_APPFLAG_GP2_INTERNAL_REAR_LONG	0x00002000	Flag pour la détection longue distance pour le premier mouvement
APP_PARAM_APPFLAG_GP2_FRONT_LEFT	0x00004000	Capteur situé sur l'avant gauche du robot
APP_PARAM_APPFLAG_GP2_FRONT_RIGHT	0x00008000	Capteur situé sur l'avant droite du robot
Capteur à contact		
APP_PARAM_APPFLAG_SW_1	0x00010000	Capteur à contact (Non utilisé)
APP_PARAM_APPFLAG_SW_2	0x00020000	Capteur à contact (Non utilisé)
APP_PARAM_APPFLAG_SW_3	0x00040000	Capteur à contact (Non utilisé)
APP_PARAM_APPFLAG_SW_4	0x00080000	Capteur à contact (Non utilisé)
APP_PARAM_APPFLAG_SW_5	0x00100000	Capteur à contact (Non utilisé)
APP_PARAM_APPFLAG_SW_6	0x00200000	Capteur à contact (Non utilisé)
APP_PARAM_APPFLAG_SW_7	0x00400000	Capteur à contact (Non utilisé)
APP_PARAM_APPFLAG_SW_8	0x00800000	Capteur à contact (Non utilisé)
Flag de fonctionnement		
APP_PARAM_APPFLAG_BIT24	0x01000000	Non utilisé
APP_PARAM_APPFLAG_BIT25	0x02000000	Non utilisé
APP_PARAM_APPFLAG_BIT26	0x04000000	Non utilisé
APP_PARAM_APPFLAG_BIT27	0x08000000	Non utilisé
APP_PARAM_APPFLAG_BIT28	0x10000000	Non utilisé
APP_PARAM_APPFLAG_BIT29	0x20000000	Non utilisé
APP_PARAM_APPFLAG_BIT30	0x40000000	Non utilisé
APP_PARAM_APPFLAG_BIT31	0x80000000	Non utilisé



Documentation:

Informatique



Flags de Stratégie : APP_PARAM_STRATEGYFLAG		
Nom du Flag	Valeur	Description
Actions réalisées		
APP_PARAM_STRATEGYFLAG_GLASS_1_DONE	0x00000001	Indique si la sous-stratégie « Verres 1 » a été réalisée
APP_PARAM_STRATEGYFLAG_GLASS_2_DONE	0x00000002	Indique si la sous-stratégie « Verres 2 » a été réalisée
APP_PARAM_STRATEGYFLAG_GIFT_1_TRY	0x00000004	Indique si le cadeau 1 a été essayé
APP_PARAM_STRATEGYFLAG_GIFT_1_DONE	0x00000008	Indique si le cadeau 1 a été ouvert
APP_PARAM_STRATEGYFLAG_GIFT_2_TRY	0x00000010	Indique si le cadeau 2 a été essayé
APP_PARAM_STRATEGYFLAG_GIFT_2_DONE	0x00000020	Indique si le cadeau 2 a été ouvert
APP_PARAM_STRATEGYFLAG_GIFT_3_TRY	0x00000040	Indique si le cadeau 3 a été essayé
APP_PARAM_STRATEGYFLAG_GIFT_3_DONE	0x00000080	Indique si le cadeau 3 a été ouvert
APP_PARAM_STRATEGYFLAG_GIFT_4_TRY	0x00000100	Indique si le cadeau 4 a été essayé
APP_PARAM_STRATEGYFLAG_GIFT_4_DONE	0x00000200	Indique si le cadeau 4 a été ouvert
APP_PARAM_STRATEGYFLAG_BIT10	0x00000400	Non utilisé
APP_PARAM_STRATEGYFLAG_BIT11	0x00000800	Non utilisé
APP_PARAM_STRATEGYFLAG_BIT12	0x00001000	Non utilisé
APP_PARAM_STRATEGYFLAG_BIT13	0x00002000	Non utilisé
APP_PARAM_STRATEGYFLAG_BIT14	0x00004000	Non utilisé
APP_PARAM_STRATEGYFLAG_BIT15	0x00008000	Non utilisé
APP_PARAM_STRATEGYFLAG_BIT16	0x00010000	Non utilisé
APP_PARAM_STRATEGYFLAG_BIT17	0x00020000	Non utilisé
APP_PARAM_STRATEGYFLAG_BIT18	0x00040000	Non utilisé
APP_PARAM_STRATEGYFLAG_BIT19	0x00080000	Non utilisé
APP_PARAM_STRATEGYFLAG_BIT20	0x00100000	Non utilisé
APP_PARAM_STRATEGYFLAG_BIT21	0x00200000	Non utilisé
APP_PARAM_STRATEGYFLAG_BIT22	0x00400000	Non utilisé
APP_PARAM_STRATEGYFLAG_SUBSTRATEGY_RESULT	0x00800000	Indique l'état de sortie de la précédente sous-stratégie
Flag de statut		
APP_PARAM_STRATEGYFLAG_COLLISION_FRONT	0x01000000	Flag de collision frontale
APP_PARAM_STRATEGYFLAG_COLLISION_REAR	0x02000000	Flag de collision arrière
APP_PARAM_STRATEGYFLAG_COLLISION_LEFT	0x04000000	Flag de collision gauche
APP_PARAM_STRATEGYFLAG_COLLISION_RIGHT	0x08000000	Flag de collision droite
APP_PARAM_STRATEGYFLAG_COLLISION_LONG_REAR	0x10000000	Flag de collision frontale (avec une longue détection)
APP_PARAM_STRATEGYFLAG_REAR_HOOPS_DOWN	0x20000000	Indique si la pince est baissée
APP_PARAM_STRATEGYFLAG_COLLISION_FRONT_LEFT	0x40000000	Flag de collision frontale gauche
APP_PARAM_STRATEGYFLAG_COLLISION_FRONT_RIGHT	0x80000000	Flag de collision frontale droite



Documentation: Informatique



Le fichier « AppIncludes.h » permet d'inclure toutes les ressources nécessaires à l'exécution du programme. Il faut ajouter ce fichier dans les en-têtes des fichiers du projet pour utiliser toutes les fonctionnalités (constantes et fonctions) de la OufffTEAM.

Le fichier « App.c » est le point d'entrée du programme. C'est à partir de ce fichier que l'on lance toutes les tâches de l'application. Le processus de démarrage est le suivant :

App.c – main(...)

- | | |
|---|-----------------|
| 1. Initialisation des registres | BSP_IntDisAll() |
| 2. Initialisation du kernel µC-OSII | OSInit() |
| 3. Initialisation des variables internes du programme | AppInitVar() |
| 4. Création des outils de synchronisation de tâches | AppCreateIPCS() |
| 5. Démarrage de toutes les tâches actives | AppTaskStart |
| 6. Démarrage de l'OS | OSStart() |

Le lancement des tâches s'effectue de la manière suivante :

App.c – AppTaskStart(...)

1. Si le flag AX12_REG_PROGRAMMING est activé, le processus de réglage des AX12 est démarré. ATTENTION, si ce processus est actif, toutes les autres tâches sont désactivées.
2. Si le flag APP_TASKODO_ENABLED est activé, le processus d'odométrie est actif.
3. Si le flag APP_TASKASSER_ENABLED est activé, le processus d'asservissement est actif.
4. Si le flag APP_TASKMVT_ENABLED est activé, le processus de gestion évolué des trajectoires est actif.
5. Si le flag APP_TASKSENSORS_ENABLED est activé, le processus permettant de récupérer les informations des capteurs et piloter les effecteurs est actif.
6. Si le flag APP_TASKMAIN_ENABLED est activé, le processus d'intelligence artificiel est actif.
7. Si le flag APP_TASKTEMPO_ENABLED est activé, le processus de gestion du temps de match est actif.

NB1 : Toutes les définitions des constantes APP_... sont faites dans le fichier « AppConfig.h ».

NB2 : La définition de la constante _TARGET_440H permet d'activer ou non le mode debug sur la carte de développement.



Documentation:

Informatique



Les variables communes à toutes les tâches sont définies ci-dessous. Elles sont déclarées dans le fichier « App.h » et utilisables via le fichier « AppGlobalVars.h ».

Queues		
OS_EVENT	*AppQueueSensors	File d'attente pour stocker les messages entre les processus TaskMain et TaskSensors
void	*AppQSensorsStk[...]	Pile pour stocker les pointeurs des messages pour la file d'attente AppQueueSensors. La taille maximale de cette pile est définie par la variable APP_QUEUE_SENSORS_SIZE.
StructMsg	AppMsgStack[...]	Pile pour stocker tous les messages de l'application. La taille maximale de l'application est définie par la variable APP_QUEUES_TOTAL_SIZE qui est la somme de toutes les tailles des files de messages.
Variables globales		
StructCmd	App_CmdToTaskMvt	Variable pour stocker la prochaine commande (en provenance de « TaskMain ») devant être exécutée par « TaskMvt ».
unsigned int	App_CmdToTaskMvtId	ID pour donner la validité du message. Il sert de révision de l'information. Si l'ID de l'ordre en cours dans « TaskMvt » est inférieur à App_CmdToTaskMvtId, cela signifie qu'un nouvel ordre est en attente.
StructCmd	App_CmdToTaskAsser	Variable pour stocker la prochaine commande (en provenance de « TaskMvt ») devant être exécutée par « TaskAsser ».
unsigned int	App_CmdToTaskAssertId	ID pour donner la validité du message. Il sert de révision de l'information. Si l'ID de l'ordre en cours dans « TaskAsser » est inférieur à App_CmdToTaskAsserId, cela signifie qu'un nouvel ordre est en attente.
Mutex / Sémaphores		
OS_EVENT*	App_MutexCmdToTaskMvt	Mutex pour limiter l'accès (RW) à la variable App_CmdToTaskMvt.
OS_EVENT*	App_MutexCmdToTaskAsser	Mutex pour limiter l'accès (RW) à la variable App_CmdToTaskAsser.
OS_EVENT*	App_MutexUART1	Mutex pour limiter l'accès (RW) à l'UART1
OS_EVENT*	App_MutexUART2	Mutex pour limiter l'accès (RW) à l'UART2
OS_EVENT*	App_MutexPMP	Mutex pour limiter l'accès (RW) au PMP
Enumeration		
EnumColor	AppCurrentColor	Couleur actuelle (lue depuis le BSP)
Flags		
OS_FLAG_GRP	*AppFlags	Contient tous les FLAGS (sur 32bits) de l'application

Toutes ces variables sont créées et initialisées par les fonctions :

App.c – AppInitVar(. . .)
App.c – AppCreateIPCS(. . .)



Documentation:

Informatique



IV. Présentation Détailée

a) TaskOdo

Description :

Cette tâche permet de calculer à tout moment la position du robot. Elle tourne en continu et calcule de manière cyclique la position actuelle du robot. A temps régulier, elle capture la valeur des encodeurs sur les roues « folles » et effectue ensuite les calculs nécessaires pour obtenir la position en (abscisse, ordonnée, angle).

Fonctionnement :

Ce processus n'interagit avec aucun autre processus. Il met à jour de manière régulière la position actuelle (en interne). Les autres processus viennent lire cette variable (mutexée) pour obtenir la position courante. Pour éviter les risques de blocages concurrents, tous les accès à la variable de positionnement se font explicitement via l'utilisation de la fonction **TaskOdo.c** – `TaskOdo_GetCurrentPos(...)` qui prend en paramètre la structure à remplir avec les informations courantes. Sur le même principe, la fonction **TaskOdo.c** – `TaskOdo_SetCurrentPos(...)` permet de modifier la position actuelle depuis un autre processus.

Algo principal :

```
| Création des Mutex/Semaphores spécifiques à TaskOdo
| Initialisation de la position initiale (tout est mis à zéro)
| Initialisation et démarrage des Timers liés à cette tâche
| Faire(Toujours)
|   Récupération du Mutex
|   Si récupération ok
|     Calcul de la nouvelle position (ou calibration en fonction du mode choisi)
|     Si le compteur de debug est à 0
|       Afficher les informations de debug sur la sortie UART2
|       Incrémenter le compteur de debug
|       Le remettre à 0 si il est > à 50
|     FinSi
|   FinSi
| FinFaire
```

Algo pour calculer la nouvelle position [**TaskOdo.c** – `position_manager_process(...)`]:

```
| Init des variables de la fonction
| Attendre la fin de la capture des informations des codeurs
| Lecture des informations du codeur droit
| Lecture des informations du codeur gauche
| Calcul de l'écart sur la roue droite
| Calcul de l'écart sur la roue gauche
| Conversion des données en mm et radians (pour faire le calcul)
| Calcul de la nouvelle position
| Prendre le Mutex (bloquer l'accès à la variable)
|   Enregistrement de la nouvelle position dans la structure adéquate
|   Vérification et correction de l'angle
|   Mise à jour des incrémentations des codeuses (pour le mouvement pivot)
|   Mise à jour du flag de mouvement
| Lâcher le Mutex (libérer l'accès à la variable)
| Stocker la variable courante pour le prochain calcul
```

Auteur : [Fifi_one \(philippe.bechet@cpe.fr\)](mailto:Fifi_one (philippe.bechet@cpe.fr))

Team : [OufffTEAM \(oufffteam@gmail.com\)](mailto:OufffTEAM (oufffteam@gmail.com))

Coupe : 2013



Documentation:

Informatique



b) TaskAsser

Description :

Cette tâche permet de calculer les ordres à envoyer aux moteurs afin d'atteindre la position souhaitée.

Fonctionnement :

Cette tâche vient lire la position actuelle du robot via le processus TaskOdo et calcule en conséquence l'ordre à envoyer aux moteurs afin d'atteindre la position désirée. Elle se déroule en continu et se met à jour dès que la position du robot évolue. « TaskAsser » reçoit la position à atteindre à partir de la tâche « TaskMvt » avec l'utilisation de la variable App_CmdToTaskAsser (App_CmdToTaskAsserId permet de gérer la mise à jour de la commande).

Algo principal :

```
| Initialisation des variables de la tâche
| Faire(Toujours)
|   | Temporisation de ASSER_SAMPLING secondes
|   | Lecture de la position actuelle du robot
|   | Si(App_CmdToTaskAsserId est > à l'ID de la commande en cours de traitement)
|   |   | Prendre le Mutex (bloquer l'accès à la variable)
|   |   |   | Récupération du contenu de App_CmdToTaskAsser comme ordre courant
|   |   | Lâcher le Mutex (libérer l'accès à la variable)
|   |   | Mettre à jour l'ID de la commande en cours
|   | Sélectionner(PourLaCommandeCourante)
|   |   | Cas NouvellePosition :
|   |   |     | Mettre à jour les données internes à la tâche avec les nouvelles valeurs
|   |   |     | Mettre à jour les données de TaskOdo avec les nouvelles valeurs
|   |   | FinCas
|   |   | Cas MvtEnAngleSeulement :
|   |   |     | Mettre à jour les données internes à la tâche avec les nouvelles valeurs
|   |   | FinCas
|   |   | Cas MvtEnDistanceSeulement :
|   |   |     | Mettre à jour les données internes à la tâche avec les nouvelles valeurs
|   |   | FinCas
|   |   | Cas MvtSimple et MvtComplexe :
|   |   |     | Mettre à jour les données internes à la tâche avec les nouvelles valeurs
|   |   | FinCas
|   |   | Cas MvtStop :
|   |   |     | Lire les données de positionnement actuel (depuis TaskOdo)
|   |   |     | Mettre à jour les données internes avec les valeurs de TaskOdo
|   |   | FinCas
|   |   | Cas MvtPivot :
|   |   |     | Si le pivot est sur la roue de droite
|   |   |     |     | Calculer le nombre d'incrément à faire sur la roue droite
|   |   |     | Sinon Si le pivot est sur la roue de gauche
|   |   |     |     | Calculer le nombre d'incrément à faire sur la roue gauche
|   |   |     | FinSi
|   |   | FinCas
|   | FinSelectionner
```



```
FinSi
Lire la position courante depuis TaskOdo
Selectionner(ModeDeMouvement)
| Cas AngleSeulement :
|   Calculer l'ordre à envoyer aux moteurs en angle seulement
| FinCas
| Cas DistanceSeulement :
|   Calculer l'ordre à envoyer aux moteurs en distance seulement
| FinCas
| Cas MouvementMixte :
|   Calculer l'ordre à envoyer aux moteurs en mouvement mixte
| FinCas
| Cas MouvementPivot :
|   Calculer l'ordre à envoyer aux moteurs en pivot
| FinCas
FinSelectionner
Vérifier la validité des ordres à envoyer aux moteurs
Envoyer les ordres aux moteurs
FinFaire
```

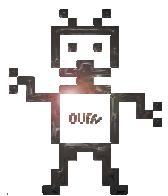
c) TaskMvt

Description :

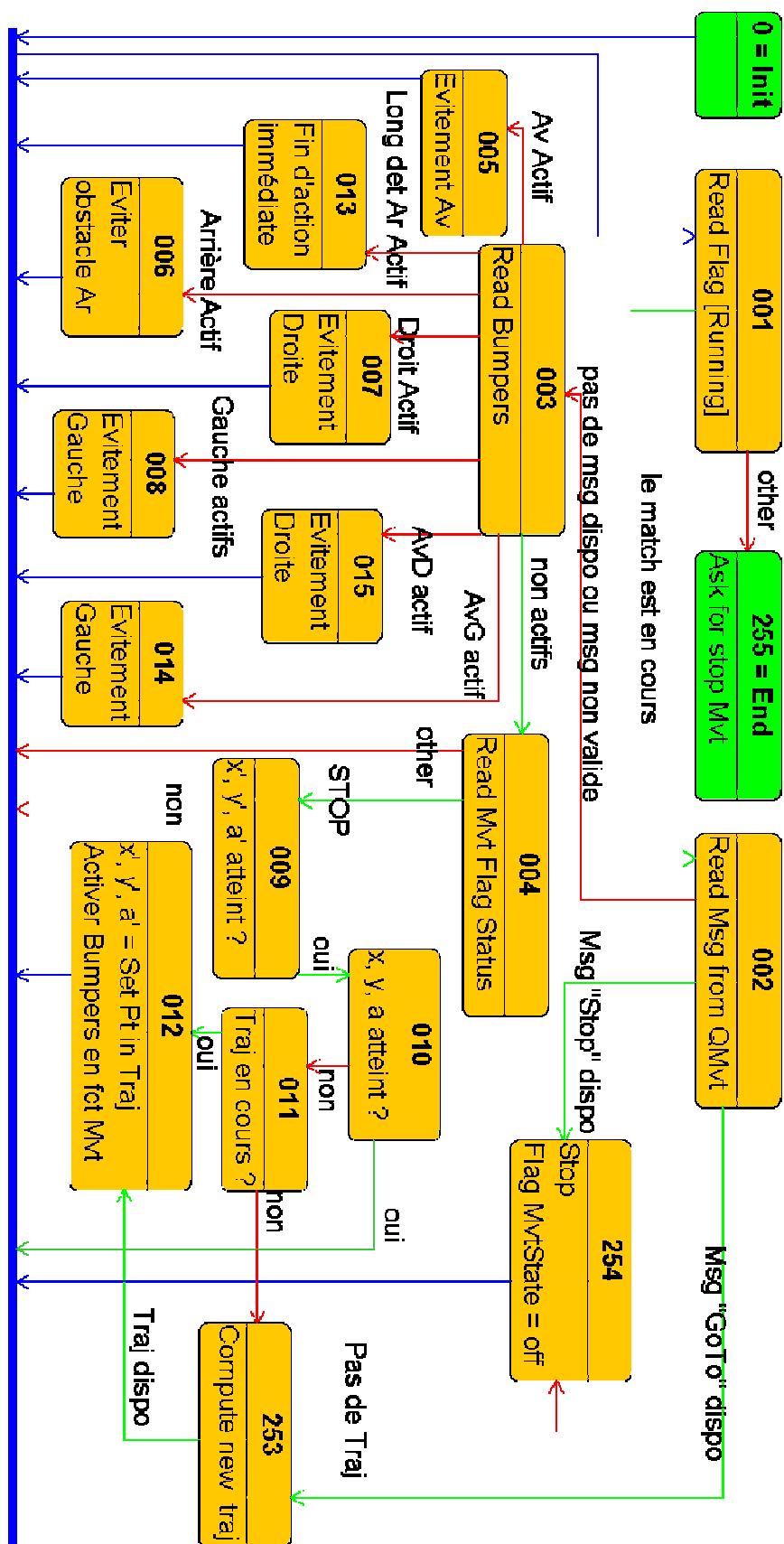
Cette tâche est en charge de toute la gestion du déplacement du robot. C'est elle qui génère et qui valide les trajectoires que doit réaliser le robot afin d'atteindre le point choisi. Elle embarque toute l'intelligence relative aux déplacements du robot. C'est cette tâche qui prend en compte les informations de positionnement et qui génère les nouvelles trajectoires lors d'une collision du système d'anticollision.

Fonctionnement :

Cette tâche reçoit un point à atteindre depuis TaskMain. Elle crée une trajectoire adaptée et envoie les ordres de déplacement à TaskAsser. Toutes les méthodes liées aux calculs de trajectoire sont présentes dans le module LibMoving (détailé dans la suite de ce document)



Algo principal (schéma) :





Documentation:

Informatique



Algo principal (détailé) :

```
| Initialisation des variables de la tâche
| Faire
|   | Attente de 10 ms pour relâcher le processeur
|   | Récupération de l'EtatCourant (à partir de la variable NextState)
|   | Lecture des FLAGS système
|   | Selectionner(EtatCourant)
|   |   | Cas 0 (=EtatInitial) :
|   |   |     Si(BoutonStartEstPressé)
|   |   |       | NextState = 1
|   |   |     Sinon
|   |   |       | NextState = 0
|   |   |       | Si(App_CmdToTaskMvt définit une nouvelle position)
|   |   |         | Mettre à jour la position du robot
|   |   |       | FinSi
|   |   |     FinSi
|   |   | FinCas
|
|   | Cas 1 (=LectureEtatTimer) :
|   |   Si(LeTempsEstTerminé)
|   |     | NextState = 255
|   |   Sinon
|   |     | NextState = 2
|   |   FinSi
|   | FinCas
|
|   | Cas 2 (=LireMessageProvenantDuMain) :
|   |   Si(App_CmdToTaskMvtId est > à l'ID de la dernière commande exécutée)
|   |     | Récupération du message
|   |     | Mise à jour de l'ID de la dernière commande exécutée
|   |     | Selectionner(LaNouvelleCommandeReçue)
|   |     |   | Cas Stop :
|   |     |     | NextState = 254
|   |     |   FinCas
|   |     |   Cas Mvt_AngleSeulement :
|   |     |   Cas Mvt_Simple :
|   |     |   Cas Mvt_DistanceSimple :
|   |     |   Cas Mvt_ModeMixte :
|   |     |   Cas Mvt_Pivot :
|   |     |   Cas Mvt_DefinirNouvellePosition :
|   |     |     | NextState = 253
|   |     |   FinCas
|   |     |   CasParDéfaut :
|   |     |     | NextState = 3
|   |     |   FinCas
|   |     | FinSelectionner
|   |   Sinon
|   |     | NextState = 3
|   |   FinSi
|
| FinCas 3 (=Lecture des bumpers) :
|   | Lecture des états des bumpers
|   | Si(Bumpers de face activés)
|   |     | NextState = 5
```

Auteur : Fifi_one (philippe.bechet@cpe.fr)

Team : OufffTEAM (oufffteam@gmail.com)

Coupe : 2013

Documentation:

Informatique



```

Sinon Si(Bumpers frontal gauche activés)
|   nextState = 14
Sinon Si(Bumpers frontal droit activés)
|   nextState = 15
Sinon Si(Bumpers de gauche activés)
|   nextState = 7
Sinon Si(Bumpers de droite activés)
|   nextState = 8
Sinon Si(Bumpers de dos activés)
|   nextState = 6
Sinon Si(Bumpers de dos activés avec une longue détection)
|   nextState = 13
Sinon
|   nextState = 4
FinSi
FinCas

```

Cas 4 (=Lecture du FLAG de mouvement) :

```

Si(l'état du mouvement est STOP)
|   nextState = 9
Sinon
|   nextState = 1
FinSi
FinCas

```

Cas 5 (=Séquence d'évitement FRONT) :

```

Envoyer le message d'arrêt à TaskAser
Réinitialiser les compteurs de Timeout

```

```

Faire
|   Incrémenter le compteur de timeout
|   Tempo de 500ms
|   Lecture du FLAG des bumpers de FRONT
TantQue(Bumpers FRONT activés et compteur Timeout inférieur au max)

```

```

Si(compteur Timeout inférieur au Timeout max)
|   On lève le Flag pour indiquer que l'action s'est fini en timeout
|   On indique que l'action est terminée
|   On désactive le déplacement actuel
Sinon
|   On envoie de nouveau le point à atteindre
|   nextState = 9
FinSi
FinCas

```

Cas 6 (=Séquence d'évitement BACK) :

```

Envoyer le message d'arrêt à TaskAser
Réinitialiser les compteurs de Timeout

```

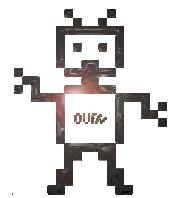
```

Faire
|   Incrémenter le compteur de timeout
|   Tempo de 500ms
|   Lecture du FLAG des bumpers de BACK
TantQue(Bumpers BACK activés et compteur Timeout inférieur au max)

```

Documentation:

Informatique



```

Si(compteur Timeout inférieur au Timeout max)
|   On lève le Flag pour indiquer que l'action s'est fini en timeout
|   On indique que l'action est terminée
|   On désactive le déplacement actuel
Sinon
|   On envoie de nouveau le point à atteindre
|   NextState = 9
FinSi
FinCas

Cas 7 (=Séquence d'évitement RIGHT) :
Envoyer le message d'arrêt à TaskAser
Réinitialiser les compteurs de Timeout

Faire
|   Incrémenter le compteur de timeout
|   Tempo de 500ms
|   Lecture du FLAG des bumpers de RIGHT
TantQue(Bumpers RIGHT activés et compteur Timeout inférieur au max)

Si(compteur Timeout inférieur au Timeout max)
|   On lève le Flag pour indiquer que l'action s'est fini en timeout
|   On indique que l'action est terminée
|   On désactive le déplacement actuel
Sinon
|   On envoie de nouveau le point à atteindre
|   NextState = 9
FinSi
FinCas

Cas 8 (=Séquence d'évitement LEFT) :
Envoyer le message d'arrêt à TaskAser
Réinitialiser les compteurs de Timeout

Faire
|   Incrémenter le compteur de timeout
|   Tempo de 500ms
|   Lecture du FLAG des bumpers de LEFT
TantQue(Bumpers LEFT activés et compteur Timeout inférieur au max)

Si(compteur Timeout inférieur au Timeout max)
|   On lève le Flag pour indiquer que l'action s'est fini en timeout
|   On indique que l'action est terminée
|   On désactive le déplacement actuel
Sinon
|   On envoie de nouveau le point à atteindre
|   NextState = 9
FinSi
FinCas

Cas 9 (=Vérification de la position du robot) :
Si(Le point actuel est-il atteint ?)
|   NextState = 10
Sinon
|   Si(Un point est en cours d'atteinte)

```

Documentation:

Informatique



```

Incrémenter le compteur de TimeOut
Sinon
| Annuler le TimeOut
FinSi
Si(Le compteur de TimeOut est > au TIMEOUT)
| Si(Le mouvement actuel est bloquant)
| | Libérer le FLAG du statut d'action
FinSi
Désactiver la trajectoire actuelle
NextState = 1
FinSi
NextState = 1
FinSi
FinCas

Cas 10 (=Est-ce que le point final est atteint) :
Lecture de la position actuelle
Si(L'un des paramètres est = à USE_CURRENT_VALUE)
| Mettre à jour la commande avec la position actuelle
FinSi
Si(Le point final est atteint)
| Si(Le mouvement actuel est bloquant)
| | Libérer le FLAG du statut d'action
FinSi
Désactiver la trajectoire actuelle
Réinitialiser tous les FLAGS
NextState = 1
Sinon
| NextState = 11
FinSi
FinCas

Cas 11 (=Existe-t-il un point suivant dans la trajectoire ?) :
Si(Il existe un point suivant)
| nextState = 12
Sinon
| nextState = 253
FinSi
FinCas

Cas 12 (=Utilisation du point suivant dans la trajectoire) :
Sélection du point suivant
Si(Il existe un point suivant)
| Envoyer le nouveau point à atteindre à TaskAsser
FinSi
NextState = 1
FinCas

Cas 13 (=detection d'un obstacle à longue dsitance) :
On stoppe immédiatement la trajectoire et on sort en timeout
NextState = 1
FinCas

Cas 14 (=Séquence d'évitement FRONT_LEFT) :
Envoyer le message d'arrêt à TaskAser

```

Documentation:

Informatique



Réinitialiser les compteurs de Timeout

Faire

- | Incrémenter le compteur de timeout
- | Tempo de 500ms
- | Lecture du FLAG des bumpers de FRONT_LEFT
- | TantQue(Bumpers FRONT_LEFT activés et compteur inférieur au max)

Si(compteur Timeout inférieur au Timeout max)

- | On lève le Flag pour indiquer que l'action s'est fini en timeout
- | On indique que l'action est terminée
- | On désactive le déplacement actuel

Sinon

- | On envoie de nouveau le point à atteindre
- | NextState = 9

FinSi

FinCas

Cas 15 (=Séquence d'évitement FRONT_RIGHT) :

Envoyer le message d'arrêt à TaskAser

Réinitialiser les compteurs de Timeout

Faire

- | Incrémenter le compteur de timeout
- | Tempo de 500ms
- | Lecture du FLAG des bumpers de FRONT_RIGHT
- | TantQue(Bumpers FRONT_RIGHT activés et compteur inférieur au max)

Si(compteur Timeout inférieur au Timeout max)

- | On lève le Flag pour indiquer que l'action s'est fini en timeout
- | On indique que l'action est terminée
- | On désactive le déplacement actuel

Sinon

- | On envoie de nouveau le point à atteindre
- | NextState = 9

FinSi

FinCas

Cas 253 (=Calculer une nouvelle trajectoire) :

Effacer la trajectoire courante

Calculer la nouvelle trajectoire

Si(Une trajectoire a été trouvée)

- | NextState = 12

Sinon

- | NextState = 254

FinSi

FinCas

Cas 254 (=Ordre de stopper le mouvement) :

Envoyer le message de STOP à TaskAsser

Si(Le mouvement actuel est bloquant)

- | Libérer le FLAG du statut d'action

FinSi

Désactiver la trajectoire actuelle

Inhiber tous les capteurs de mouvement du robot



Documentation:

Informatique



```
    NextState = 1
    FinCas

    Cas 255 (=Fin du match) :
        Désactiver la trajectoire actuelle
        Envoyer le message de STOP à TaskAsser
    FinCas

    Cas par défaut :
        Désactiver la trajectoire actuelle
        NextState = 254
    FinCas
    FinSelectionner
TantQue(L'état courant est différent de 255)

Faire(Toujours)
|   Attente de 1 heure
FinFaire
```

d) TaskSensors

Description :

Cette tâche est en charge de toute la gestion des capteurs et des actionneurs du robot. C'est elle qui fait l'interface entre le monde extérieur et le robot. C'est cette tâche qui se charge de lever les flags d'anticollision.

Fonctionnement :

Cette tâche reçoit un ordre depuis TaskMain (via QSensors) et l'exécute. Une fois l'action terminée, et si l'action en cours est bloquante, TaskSensor lève le flag ACTION_STATUS afin de signaler aux autres tâches que l'action est terminée.

Algo principal :

```
Si(le flag d'utilisation de la tirette est à VRAI)
    TantQue(La tirette est non présente)
        |   Lire la couleur
    FinTantQue
    TantQue(La tirette est présente)
        |   Lire la couleur
    FinTantQue
FinSi
Lire la couleur
Lever le Flag Tirette pour lancer toutes les tâches
Initialiser le Flag d'action
Initialiser la pince du robot
Faire
    |   Lire l'état des capteurs (collision + pince)
    EtatCourant = EtatSuivant
    Selectionner(EtatCourant)
        |   Cas 0 : Attente d'un message provenant de TaskMain
            Si(Un message est en cours d'utilisation)
                |       Libérer le message
            FinSi
            Récupérer un nouveau message
```

Auteur : Fifi_one (philippe.bechet@cpe.fr)

Team : OufffTEAM (oufffteam@gmail.com)

Coupe : 2013



Documentation:

Informatique



```
Si(Un message n'a pas été trouvé)
|   EtatSuivant = 0
Sinon
|   Selectionner(La commande depuis le message courant)
|       Cas Sensors_SetHoopLevel:
|           Modifier la hauteur à partir du paramètre 1 du message
|           Si(l'action courante est bloquante)
|               Lever le Flag pour indiquer la fin de l'action
|           FinSi
|       FinCas
|       Cas Sensors_SetArmsStatus:
|           Ouvrir/Fermer le bras en fonction des paramètres 1 et 2
|           Si(l'action courante est bloquante)
|               Lever le Flag pour indiquer la fin de l'action
|           FinSi
|       FinCas
|       Cas Sensors_SetHolderLevel :
|           Lever la pince en fonction du paramètre 1
|           Si(l'action courante est bloquante)
|               Lever le Flag pour indiquer la fin de l'action
|           FinSi
|       FinCas
|       Cas par défaut :
|           EtatSuivant = 0
|       FinCas
|   FinSelectionner
|   FinSinon
FinCas

Cas 255 : Etat final
    EtatSuivant = 255
FinCas

Cas par défaut
    EtatSuivant = 0
FinCas
FinSelectionner
TantQue(EtatCourant != 255)
Faire(Toujours)
|   Patienter 30 sec
FinFaire
```

Sous-fonctions de la tâche :

- TaskSensors_IsStartButtonPressed()
 - *Action* : Lecture de l'état de la tirette
 - *Retour* : **BOOLEAN**
 - OS_TRUE : La tirette est en place
 - OS_FALSE : La tirette n'est plus en place
 - *Paramètres* : Aucun
- TaskSensors_ReadColor()
 - *Action* : Lecture de la couleur
 - *Retour* : Via la variable **AppCurrentColor**



Documentation:

Informatique



- c_ColorA : Bleu
- c_ColorB : Rouge
- Paramètres : Aucun
- TaskSensors_CheckBumpers()
 - Action : Vérifier l'état des bumpers
 - Retour : Mise à jour des infos GP2 du Flag AppFlags
 - Paramètres : Aucun
- TaskSensors_GenerateStrategyFlags()
 - Action : Crée les flag de stratégies en fonction des flags des bumpers
 - Retour : Mise à jour des infos de stratégie du Flag AppStrategieFlag
 - Paramètres : Aucun
- TaskSensors_FunnyAction()
 - Action : Gonfle le ballon pendant 10 sec
 - Retour : Aucun
 - Paramètres : Aucun
- TaskSensors_SetHoopLevel(...)
 - Action : Change la hauteur de la pince
 - Retour : Aucun
 - Paramètres :
 - HOOP_LEVEL_DOWN: La pince est basse
 - HOOP_LEVEL_UP : La pince est haute
- TaskSensors_SetArmsStatus(...)
 - Action : Définit le statut du bras
 - Retour : Aucun
 - Paramètre 1 : Bras Gauche
 - ARM_OPEN : Le bras est ouvert
 - ARM_CLOSED : Le bras est fermé
 - ARM_FRONT : Le bras est ouvert sur l'avant du robot
 - Paramètre 2 : Bras Droit
 - ARM_OPEN : Le bras est ouvert
 - ARM_CLOSED : Le bras est fermé
 - ARM_FRONT : Le bras est ouvert sur l'avant du robot



Documentation: Informatique



e) TaskTempo

Description :

Cette tâche est en charge de la gestion du temps. C'est elle qui indique à toutes les autres tâches le statut actuel.

Fonctionnement :

Dès que la tirette est enlevée, le compte à rebours commence. Une fois le temps de match écoulé, TaskTempo lève le Flag TIMER_STATUS pour indiquer à toutes les tâches de se mettre en pause.

Algo principal :

```
| Attente de la tirette
| Calcul du temps de match
| Faire
|   | Se mettre en pause durant APP_PARAM_TEMPO_RESOLUTION millisecondes
|   | Lire le TempsActuel
| TantQue(TempsActuel < temps de match)

| Lever le Flag TIMER_STATUS
| Faire(Toujours)
|   | Attendre 1 heure
| FinFaire
```

f) TaskMain

Description :

Cette tâche est la tâche principale. C'est elle qui est en charge de toute l'intelligence du robot. C'est cette tâche qui se charge de coordonner les ordres envoyés aux tâches TaskSensors et TaskMvt.

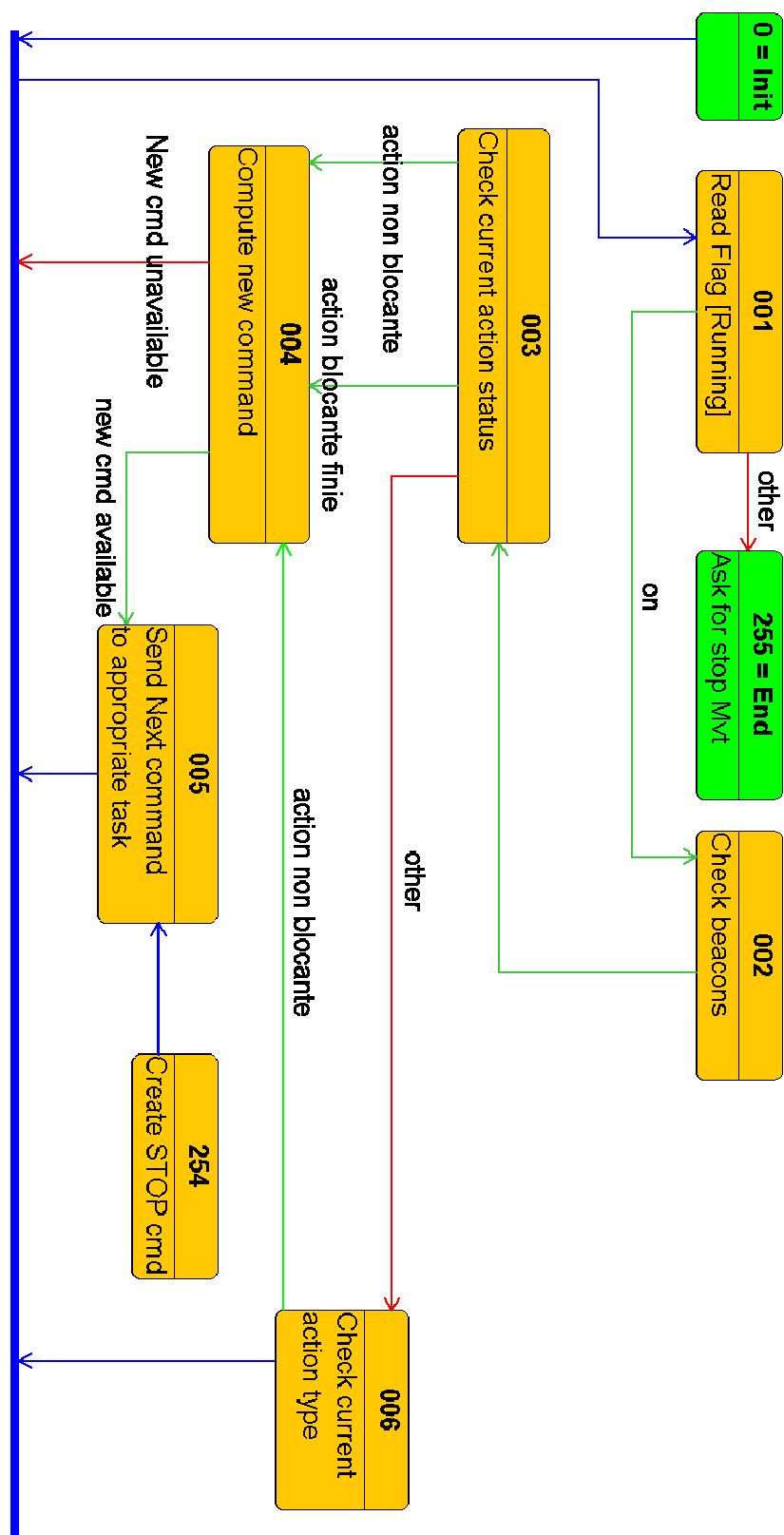
Fonctionnement :

TaskMain coordonne l'envoi des ordres entre toutes les tâches. Elle est en charge de surveiller l'évolution des Flags afin de générer les ordres en conséquence. C'est le point central du programme.

Algo principal (schéma) :

Documentation:

Informatique



Algo principal :

| Attente de 1 sec (attente des autres tâches)

Auteur : Fifi_one (philippe.bechet@cpe.fr)

Team : OufffTEAM (oufffteam@gmail.com)

Coupe : 2013

Documentation:

Informatique



TantQue(La couleur n'est pas valide)

| Temporisation

FinTantQue

Faire

| Si(CouleurCourante est différente de CouleurPrécédente)

| | CouleurPrécédente = CouleurActuelle

| | Si(Une position est disponible pour CouleurActuelle)

| | | Envoyer la position à TaskMvt

| | FinSi

| FinSi

| Lire les Flags actuels

TantQue(Le Flag de démarrage n'est pas présent dans les Flag système)

Faire

| EtatCourant = EtatSuivant

| Lire les Flags systèmes

| Selectionner(EtatCourant)

| | Cas 0 : Initialisation

| | | EtatSuivant = 1

| | FinCas

| | Cas 1 : Vérification du Timer

| | | Si(Le Flag de TIMER est activé)

| | | | EtatSuivant = 255

| | | Sinon

| | | | EtatSuivant = 2

| | | FinSi

| | FinCas

| | Cas 2 : Vérification des balises

| | | EtatSuivant = 3

| | FinCas

| | Cas 3 : Vérification du statut courant

| | | Si(L'action en cours est non bloquante)

| | | | Si(Le Flag de fin d'action est levé)

| | | | | EtatSuivant = 4

| | | Sinon

| | | | | EtatSuivant = 6

| | | | FinSi

| | | Sinon

| | | | | EtatSuivant = 4

| | | | FinSi

| | Cas 4 : Récupérer l'action suivante

| | | Si(Une nouvelle action est dispo)

| | | | EtatSuivant = 5

| | | Sinon

| | | | EtatSuivant = 1

| | | FinSi

| | Cas 5 : Envoyer l'action suivante

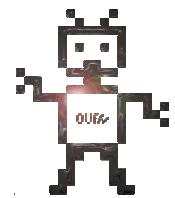
| | | Si(L'action actuelle est définie)

| | | | Réinitialiser les variables d'échanges



Documentation:

Informatique



Selectionner(L'action actuelle)

- Cas MvtAngleSeulement :
- Cas MvtDistanceSeulement :
- Cas MvtModeMixte :
- Cas MvtPivot :
- Cas MvtSpline :
- Cas MvtSimple :
- Cas MvtStop :
- Cas MvtAttente :
- Cas MvtDefinirNouvellePosition :
 - Envoyer la nouvelle commande à TaskMvt

FinCas

Cas DéfinirHauteurPince :

- Cas DéfinirStatutBras :
 - Préparer le message
 - Envoyer le message à TaskSensors

FinCas

Cas Attendre :

- Effectuer l'attente
- Lever le flag bloquant de l'action en cours

FinCas

Cas ModifierLeFlagStrategie :

- Création du masque à partir du param 1 du message
- Si(Param2 est à Faux)
 - Mettre à 0 le flag de stratégie en fonction du masque
- Sinon
 - Mettre à 1 le flag de stratégie en fonction du masque

FinSi

- Lever le flag bloquant de l'action en cours

FinCas

Cas AgirEnFonctionDuFlagDeStratégie :

Cas AgirEnFonctionDuFlagSystem :

Cas CommandeNonDéfinie :

- Lever le flag bloquant de l'action en cours

FinCas

Cas par défaut :

- Afficher un message d'erreur
- Lever le flag bloquant de l'action en cours

FinCas

FinSelectionner

Abaissier le Flag d'action

FinSi

EtatSuivant = 1

FinCas

Cas 6 : Verifier le Flag d'action

Selectionner(Le type de l'action actuelle)

- Cas ActionBloquante :
 - EtatSuivant = 1

FinCas

Auteur : Fifi_one (philippe.bechet@cpe.fr)

Team : OufffTEAM (oufffteam@gmail.com)

Coupe : 2013



```

Cas ActionNonBloquante :
    EtatSuivant = 4
FinCas

Cas par défaut :
    EtatSuivant = 1
FinCas
FinSelectionner

Cas 254 : Action à faire si les balises sont activées
    Préparer l'action à exécuter
    EtatSuivant = 5
FinCas

Cas 255 : Etat Final
    EtatSuivant = 255
FinCas

Cas par défaut
    EtatSuivant = 1
FinCas
FinSelectionner
TantQue(EtatCourant = 255)

Faire(Toujours)
|   Attendre 1 heure
FinFaire

```

g) LibMoving

Description :

Ce module contient toutes les méthodes de base qui permettent le calcul des trajectoires pour le robot.

Fonctionnement :

Les méthodes sont utilisées dans la tâche TaskMvt pour le calcul des trajectoires.

Méthodes :

```
void LibMoving_MoveInMM(int Dist, INT8U Speed, StructCmd *NextSetpoint);
    - Param1 (Integer) : Distance en mm à parcourir par le robot en ligne droite
    - Param2 (INT8U) : Vitesse pour le mouvement (entre 0 et 100%)
    - Param3 (StructCmd*) : Structure contenant le point à atteindre
        o Le type de mouvement est un mouvement simple
        o L'action est bloquante
    - Retour : La valeur renvoyée dans NextSetpoint n'est pas contrôlée (par rapport à la taille du terrain) avant d'être renvoyée.
```



Documentation:



Informatique

```
void LibMoving_RotateInDeg(int AngleInDeg, INT8U Speed, StructCmd *NextSetpoint);
    - Param1 (Integer) : Angle en degré à parcourir
    - Param2 (INT8U) : Vitesse pour le mouvement (entre 0 et 100%)
    - Param3 (StructCmd*) : Structure contenant le point à atteindre
        o Le type de mouvement est un mouvement simple
        o L'action est bloquante
    - Retour : La valeur retournée dans NextSetpoint n'est pas contrôlée (par rapport à la taille du terrain) avant d'être retournée.
```

```
void LibMoving_MoveToAngleInDeg(int AngleToGoInDeg, INT8U Speed, StructCmd *NextSetpoint);
    - Param1 (Integer) : Angle en degré à atteindre dans le repère du terrain
    - Param2 (INT8U) : Vitesse pour le mouvement (entre 0 et 100%)
    - Param3 (StructCmd*) : Structure contenant le point à atteindre
        o Le type de mouvement est un mouvement simple
        o L'action est bloquante
    - Retour : La valeur retournée dans NextSetpoint n'est pas contrôlée (par rapport à la taille du terrain) avant d'être retournée.
```

```
void LibMoving_ComputeNewPath(StructCmd *ExpectedCmd, StructCmd *NewPath, INT8S *NewPathLength);
    - Param1 (StructCmd*) : Point à atteindre dans le repère du terrain
    - Param2 (StructCmd*) : Tableau pour recevoir tous les points de la trajectoire
    - Param3 (INT8S*) : Entier pour recevoir le nombre d'ordres de la trajectoire
    - Retour : Les valeurs retournées sont faites via les variables NewPath et NewPathLength.
```

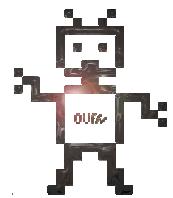
```
void LibMoving_CreateEscapeSeq(INT8U EscapeSeqType, INT8U Speed, StructCmd *NewPath, INT8S *NewPathLength);
    - Param1 (INT8U) : Type de mouvement d'esquive à réaliser
        o APP_MOVING_ESCAPE_SEQ_RIGHT : Passer par la droite
        o APP_MOVING_ESCAPE_SEQ_LEFT : Passer par la gauche
        o APP_MOVING_ESCAPE_SEQ_BACK : Reculer
    - Param2 (INT8U) : Vitesse pour le mouvement (entre 0 et 100%)
    - Param3 (StructCmd*) : Tableau pour recevoir tous les points de la trajectoire
    - Param4 (INT8S*) : Entier pour recevoir le nombre d'ordres de la trajectoire
    - Retour : Les valeurs retournées sont faites via les variables NewPath et NewPathLength.
```

```
BOOLEAN LibMoving_IsSetpointReached(StructCmd *SetpointToTest);
    - Param1 (StructCmd*) : Point que le robot doit atteindre
    - Retour : Retourne OS_TRUE si le point est atteint (si la position actuelle du robot est comprise dans un cercle d'approche défini par les coefficients APP_MOVING_APPROACH_PRECISION_COEF et APP_MOVING_DIST_APPROACH_PRECISION), OS_FALSE sinon. Le type d'approche est calculé en fonction du type de mouvement.
```



Documentation:

Informatique



h) AppGlobalFunc

Description :

Ce module contient toutes les fonctions génériques de bases.

Méthodes :

void AppDebugMsg(**char** *DebugMsg);

- Param1 (*char**): Chaine de caractère à afficher sur la sortie debug (serial port)
- Retour : Pas de retour.

BOOLEAN AppPostQueueMsg(**OS_EVENT** *PtrQueue, **StructMsg** *PtrMsgToPost);

- Param1 (*OS_EVENT**): Pointeur vers la queue où il faut poster le message
- Param2 (*StructMsg**): Message à poster. Le message est copié puis la copie est ajoutée dans la queue
- Retour : **OS_TRUE** si le message est posté correctement, **OS_FALSE** sinon.

float AppConvertRadInDeg(**float** ValueInRad);

- Param1 (*float*): Angle (en radian) à convertir en degré
- Retour : Retourne la valeur en degré

float AppConvertDegInRad(**float** ValueInDeg);

- Param1 (*float*): Angle (en degré) à convertir en radian
- Retour : Retourne la valeur en radian

float AppCheckAngleInRad(**float** ValueToCheck);

- Param1 (*float*): Angle (en radian) à vérifier
- Retour : Retourne un angle en radian compris dans l'intervalle $]-\pi ; \pi]$

float AppCheckAngleInDeg(**float** ValueToCheck);

- Param1 (*float*): Angle (en degré) à vérifier
- Retour : Retourne un angle en degré compris dans l'intervalle $]-180 ; 180]$

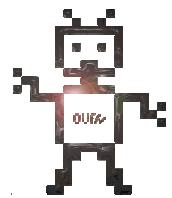
INT8U AppGetCurrentPos(**StructPos** *CurrentPos);

- Param1 (*StructPos**): Pointeur vers une structure pour recevoir la position actuelle du robot
- Retour : La valeur actuelle du robot (en provenance de TaskOdo) est stockée dans CurrentPos



Documentation:

Informatique



i) Strategy

Description :

Le répertoire Strategy contient les fichiers de stratégies du robot.

Fonctionnement :

Tous les fichiers implémentent les fonctions de bases (déclaré dans le fichier Strategy.h) :

- Strategy_GetInitCmd(...)
- Strategy_GetNextAction(...)

Chaque fichier est lié à une constante d'utilisation qui permet de sélectionner la stratégie à utiliser. Seulement une seule constante d'utilisation peut être activée à la fois (une erreur de compilation se produit sinon). Les constantes actuelles sont :

- | | |
|----------------------------|--|
| - DEFAULT_STRATEGY_ENABLED | : Stratégie par défaut (DefaultStrategy.c) |
| - HOMOL_STRATEGY_ENABLED | : Stratégie d'homologation (Homol.c) |
| - STRATEGY_1_ENABLED | : Stratégie n°1 (Strategy.c) |

Méthodes :

INT8U Strategy_GetInitCmd(*EnumColor* CurrentColor, *StructCmd* *InitCmd)

Récupérer la commande initiale en fonction de la couleur sélectionnée.

- Param1 (*EnumColor*) : Couleur actuelle
- Param2 (*StructCmd**) : Pointeur pour récupérer la commande initiale
- Retour : Retourne la commande initiale dans InitCmd

INT8U Strategy_GetNextAction(*EnumColor* CurrentColor, *StructCmd* *NextAction)

Récupérer la commande suivante en fonction de la couleur sélectionnée.

- Param1 (*EnumColor*) : Couleur actuelle
- Param2 (*StructCmd**) : Pointeur pour récupérer la commande suivante
- Retour : Retourne la commande suivante dans NextAction

La fonction *Strategy_GetNextAction(...)* est structurée de la manière suivante :

- Une variable statique (*CurrentActionID*) permet de suivre toutes les étapes de la stratégie.
- Tous les cas de la stratégie doivent contenir les paramètres suivants :
 - o [Optionnel] OSTimeDlyHMSM(...) pour temporiser l'exécution de la tâche
 - o [Obligatoire] Définition de la commande (*NextAction->Cmd = Command*)
 - o [Optionnel] Définition des paramètres de la commande
 - o [Obligatoire] Définition du type de la commande (Bloquante ou non bloquante)
 - o [Obligatoire] Incrémenter le compteur *CurrentActionID* pour passer à l'étape suivante