01 JUIN 2021

STAGE CONTINENTAL

ACASYA: Automatic Control and Analysis System Application

PANNETIER-EXT, MARVYN (UIE42738)
CONTINENTAL
1 Avenue Paul Ourliac, Toulouse

ISSUES

Item	Type	Description	Solution	State
General Bugs/Ergo ACASYA				
001	Bug	Nothing happens when clicking on the quit button in the main menu	Add callback function that will quit the program but also close the process, by adding a hidden button (use example of the back button)	FIXED
002	Ergo	Add a popup when you click on the red quit button	Use popup function	FIXED
003	Bug\Ergo	Obligatory to add label to make the software works	Implement some functions to be able to let a label blank. If there is no first label the function will check from the beginning and if there is no last label, the function will analyse until the end	FIXED
004	Ergo	Code continuous parameter directly in the code	Add Continuous to the parameter list within the generateParameterTab function	FIXED
005	Bug	Excel exe are running despite we quit the application	Modify the path which didn't work due to the spaces! Look if there are forgotten cases that need to be correct to close excel program	FIXED
006	Bug	Ask again to select the log folder when modifying a check function	Correct the conditions to ask that	FIXED
007	Ergo	Load default conf even if we return to first menu	Implement an initial configuration load button	FIXED
008	Bug	Bugs due to multiple insertion of labels when we use several check functions	Implement the prohibition to insert the same label several times and correct the check function to let them work with this new implementation of labels	FIXED
009	Bug	Infinite loop when we start again a test	Modes.c : » while (!GetGbQuitter()) && (ongoingTest == 1))"	FIXED

			Line 13590	
010	Bug	Possibility to launch a test even if the security button is not green	Add variable and if to solve this problem !	FIXED
011	Bug	Impossible to send individual RF frames	Understand the DLL called IPC and correct the code to make it works	FIXED
012	Bug	Impossible to put more than one argument in a frame	Implement a new system of argument by adding a column in the test script and modifying the DLL	FIXED
013	Ergo/Bug	Almost all the column of the database was code in hard in the files, so it was impossible to change the columns in the database or add another one.	Created function to find the columns of the arguments we want	FIXED
014	Ergo	Implement the possibility to filter the frames by their RF protocols for the check function	Add if to check the RF protocol and passed only if it is the one we gave in parameter	FIXED
015	Bug	Soft crash randomly when a too long test is launch	Bug comes from the function RunUserInterface which stop the execution sometimes. In a first time I cleaned the code by removing the most of the warning possible and then I found a way to make the .exe compatible with windows10 and with that the bug disappear	
Bugs/Ergo Mode Creation ACASYA				
101	Ergo	Expected results: Value field red could be better	Dimmed the cross in the .uir file	FIXED
102	Ergo	Insert steps	Create a function to insert a step in a precise position **Function description:** inspire by create step but collect the number of the line select with the rect structure and use insertTableRow with this number	FIXED

103	Ergo	Implement Move up and Move down function using the one that already exist in script sequence	Create right click event and adapt the move up and down that already exist for the script panel	FIXED
104	Bug	Crash when insert a label in script	Comment the line that create the bug because this line is not good as it consider the label display as a list one.	FIXED
105	Ergo/bug	Enable to change values by hand which create a bug during execution	Remove SetActiveCtrl line and put the panels in indicator instead of hot	FIXED
106	Bug	Continuous crash	Add the line to the database if it isn't already in. Code in InitConfig and button_database functions	FIXED
107	Ergo	Load expected + script !	Load the script for the two panels. Copy and adapt in each panel the load script of the other one.	FIXED
108	Ergo	Modify a line or step !	Create functions that will collect the data from the table to the create script window or create check window.	FIXED
109	Ergo	WUid	Dimmed WUid when creating a step and we use broadcast command	FIXED
110	Ergo	Color lines selected	Create a function CouleurSelection that will color the line where the mouse is currently.	FIXED
111	Ergo	Load nouveau script ecrase l'ancien	Delete all lines in the both panels before loading the new test script	FIXED
112	Ergo	Insert pre et post cond couleurs	Colors fixed	FIXED
113	Bug	Save 2 fois d'affilé plante	Message d'erreur est en arrière plan	FIXED
114	Ergo	Security to be sure the label asked in check function exists	Go throw the table each time we create a check function and compare each label of the table to the one written on the expected results panel. If there any label correspond to the ones written, display a popup.	FIXED
115	Ergo	Modif put line in script mode even if it is not a script line	see error 118	FIXED
116	Ergo	Do not give the choice to choose	Change argument in the function FileSelectPopupEx	FIXED

		.xlsx when we load a script sequence		
117	Bug	Impossible to add letters in check function for ID field for example	Put the letter always visible	FIXED
118	Bug	Allows all values (indiv, seq,range) when modifying a step	Add the line itemValue= commandExpResult; in the add exp results function	FIXED
119	Bug	Crash colors function sometime	Bug colors fixed, I read the previous color and put it on the right row	FIXED
120	Ergo	Does not Collect the RIM diameter when modify CheckCompareAcc function	Add line into modify expected result function	FIXED
121	Bug	Individual frames do not work	Look at the end of the document	FIXED
Bugs/Ergo Mode Execution ACASYA				
201	Ergo	Insert steps	Create a function to insert a step in a precise position **Function description:** inspire by create step but collect the number of the line select with the rect structure and use insertTableRow with this number	FIXED
202	Ergo	Implement moveup and movedown	Create right click event and adapt the move up and down that already exist for the script panel	FIXED
203	Ergo/bug	Enable to change values by hand which create a bug during execution	Remove SetActiveCtrl line and put the panels in indicator instead of hot	FIXED
206	Ergo	Modify a line or step !	Created a function which collect the data of the row selected and directly fill in the fields	FIXED
207	Ergo	Color	Add a color to the line selected	FIXED
208	Bug	Stop button when a test is running don't work	Solve this bug by adding security in check function if we don't have the log file we want	FIXED

First test of the Check Functions

Fonction	Travail et tests effectué	Fonctionnel ?
CheckFieldValue	**Test:** Test_Labels_CheckFieldValueFinal	Passed

	Description : Tests de toutes les combinaisons de labels dans le même script et dans des scripts différents. **Analyze** : fonctionne correctement ! **Test:** Test_otherFields_CheckFieldValue **Description :** Test différents check de fields avec différentes combinaison de labels **Result:** passed mais bug mineurs voir 117 **Correction tolerance 23/07/2021**	
CheckTimingInterBursts	**Test:** Test_Labels_CheckTimingInterBursts **Description :** Tests de toutes les combinaisons de labels dans le même script **Analyze** : fonctionne correctement ! **Correction tolerance 23/07/2021**	Passed
CheckTimingInterFrames	**Test:** Test_Labels_CheckTimingInterBursts **Description :** Tests de toutes les combinaisons de labels dans le même script **Analyze** : fonctionne correctement **Correction tolerance 23/07/2021**	Passed
CheckSTDEV	**Test:** **Description :** **Analyze** : ne fonctionne pas pour le moment ! Problem : Ne rentre pas dans la fonction :	Passed
CheckNbBursts	**Test:** Test_Labels_CheckNbBursts **Description :** Tests de toutes les combinaisons de labels dans le même script **Analyze :** passed	Passed
CheckNbFramesInBurst	**Test:** Test_Labels_CheckNbFramesInBurst **Description :** Tests de toutes les combinaisons de labels dans le même script **Analyze :** passed	Passed
CheckCompareP	**Test:** Test_Labels_CheckCompareP **Description :** Tests de toutes les combinaisons de labels dans le même script **Analyze : failed, ne donne aucun résultat** **Correction tolerance 23/07/2021** **Travail : correction de la fonction , lecture de la mauvaise colonne pour la WU et donc valeur de la pression vu par la WU toujours égale à zéro**	Passed

	22/07/2021 : **Test:** Test_Labels_ CheckCompareP **Description :** Tests de toutes les combinaisons de labels dans le même script **Analyze :** passed **Correction tolerance 23/07/2021**	
CheckCompar Acc	**Test:** Test_Labels_CheckCompareAcc **Description :** Tests de toutes les combinaisons de labels dans le même script **Analyze : Failed, plante dans la fonction** 22/07/2021 : **Test:** Test_Labels_ CheckCompareAcc **Description :** Tests de toutes les combinaisons de labels dans le même script **Analyze :** passed **Correction tolerance 23/07/2021**	Passed
CheckNoRF	**Test:** Test_Labels_CheckNoRF **Description :** Tests de toutes les combinaisons de labels dans le même script **Analyze :** Passed, semble fonctionner correctement pour toutes les combinaisons de labels	Passed
CheckTimingFirstRF	**Test:** Test_Labels_CheckTimingFirstRF **Description :** Test avec un seul label **Analyze :** passed car ne ndétecte aucune trame RF (vérifier bug report par maxime) **Test:** Test_Labels_ CheckTimingFirstRF **Description :** Test de la fonction avec un label comme défini **Analyze :** passed **Correction tolerance 23/07/2021**	Passed

Passed	Failed
10	0

ANNEXE

Vocab:

MP: Mode Parking

MD: Mode Driving

MFB: Mode First Block

MI: Mode Interim

WU: Wheel Unit

ACASYA: Automatic Control and Analysis System Application

TTM: Truck Tool Management (Not sure)

BLE: Bluetooth Low Energy*

PID: process ID

COMPREHENSION

	What ?	Where ? (file)	How ?	state
1	Attribute function	Userint.h	Functions definitions with arguments	Understood
2	attribute	everywhere	Possible state or value of each command (button, list…etc.)	Understood
3	Config Anum file	Extern file	Configure the Anum with a script	Understood
4	Log creation	GstFiles.c(.h), ResultTextFil.h??	??	In process of comprehension
5	Re-analyse	Mode.c & Mode.uir	Function, button, reuse	Understood but can be study more
6	Parameters	Script definition panel, 1944 Modes.c, Modes.C 237, 350 Modes.h 18128 Modes.c!!! * 21828 Modes.c !!!!!!!!!!	Idea : collect in a variable time for example as a string ?	Understood but can be study more

7	Execution script	Execution window	Start the sequence of tests	Understood
8	Message error when use the keyboard	Modes.c	Event_keyboard or something like that in userint.h	Understood
9	File.uir -> generate .h	User Interface Files	Use graphic interface to create button...etc.	Understood
10	Pre-condition, script, post condition choice	Script definition interface	Precondition: what we want before the test Script: the different tests we will do Post condition: return to a basic config to be able to another test	Understood
11	Check function working	Modes.c	Fonctions	Understood
12	Progression bar	Execution window and modes.c	End Modes.c file	Understood
13	DLL	ResultTextFile	It is a lib with function We can use. Definition of these functions are in ResultTextFile.h	Understood
14	TextBox analyse mode	Modes.c & IhmModes.uir	Function which open the file and copy line by line	Understood
15	ShowCurrentScript	L7482 Modes.c	Use excel function to collect the different data	Understood
16	Time scripts	Modes.c	7337 Modes.c	Understood
17	QUIT button in main menu	Mode.c, IhmModes.uir IhmModes.h	Create a new button with a callback function that will be hide and use this button for the red button quit. ADD the switch event because this button probably always listens for events contrary to the back button which launches the callback function	Corrected and understood
18	Config files	Folders		In process of comprehension
19	LF data name	Script definition		In process of comprehension
20	Seq script	Modes.c 21957 22057 also 5291	First part goal is to create a log file for this comparison Then, we the files where seq are saved and we compare them	Understood but can be study more

21	Modes.c function AffResult2 not finished and graph	Modes.c 15641 and IhmModes.uir PANELGRAPH	Find a way to implement that	
22	Modes.c & IhmModes.uir right_click		GST_SCRIPT edit	Understood but can be study more
24	Load 3 last files when it is launched	12094 Modes.c	Ecrire le nom des fichiers dans un fichier et venir chercher ces noms	Corrected and understood
25	Load 3 last files when it is launched		Bonus : faire un script windows qui vient chercher le nom du dernier fichier enregistrer dans un dossier ECRIRE LES chemin dans fichier et venir les lires Première idée à reprendre Regarder condition activation des modes Ajout lessage explication	DONE
26	generateCoverageMatrix	21284 Modes.c		Understood
27	Move up/move down insert	Modes.c run table callback		Understood
28				

Fonction	Avancée
CheckFieldValue	Filtre protocoles RF : OK Test : OK Return 1 : oui
CheckTimingInterBursts	Filtre protocoles RF : OK Test : OK Return 1 : oui
CheckTimingInterFrames	Filtre protocoles RF : OK Test : OK Return 1 : oui
CheckSTDEV	Pas besoin donc field dimmed
CheckNbBursts	Filtre protocoles RF : OK Test : OK Return 1 : non
CheckNbFramesInBurst	Filtre protocoles RF : OK Test : OK
CheckCompareP	Filtre protocoles RF : OK Test : OK
CheckCompareAcc	Filtre protocoles RF : OK Test : OK
CheckNoRF	Filtre protocoles RF : OK Test :
CheckTimingFirstRF	Filtre protocoles RF : OK Test :

Amélioration/MAJ : ***DONE***

Rendre la lecture des différentes trames de la database possible et les charger dans la liste des interfaces. Coder dans la fonction parametersdata dans le fichier Modes.c. J'ai déjà commencé une fonction pour récupérer la lettre de la colonne correspondant à l'interface donné en argument (appeler cette fonction dans parametersdata). Ensuite l'idée serait de chargé les différents types de trames dans la liste des interfaces et non de les coder en dur. Pour ça l'idée serait d'écrire une fonction permettant de trouver la case interface et récupérer le nom de l'interface toujours placé sur la droite de cette case. A la fin on aurait un tableau avec les interfaces et on l'utiliserait avec la fonction InsertListItem pour charger le boutton/liste. Il faudrait cependant modifier la fonction parametersdata qui traite les cas avec l'index correspondant au type de la trame codé en dur, le traitement peut être fait avec des if et le nom en chaine de caractère de chaque interface plutôt que l'index dans la liste. Le but de cela est d'avoir un logiciel robuste qui va continuer de fonctionner si la colonne d'une interface change ou si on en rajoute un ce qui n'est pas le cas actuellement.

Faire un document qui explique les chose à respecter dans la database pour que ça fonctionne !

Explication BUG trames individuelles : DONE

Démarche :

J'ai trouvé la fonction appelée pour exécuter les fonctions ANUM :

```
//modif carolina
void ProcessAnumCmd(int index)   //this function is called when there is a comand anum in the script
{
    CDotNetHandle exception ;
    int Command;
    unsigned int NumVal;
    char *StrVal;
    char *ID;
    //char *StorageFolder;
    unsigned int Param;

    WuAutoCheck_LFComand__Get__Command(TabAnumCmd[index],&Command,&exception);
    WuAutoCheck_LFComand__Get__StrVal(TabAnumCmd[index],&StrVal,&exception);
    WuAutoCheck_LFComand__Get__NumVal(TabAnumCmd[index],&NumVal,&exception);
    WuAutoCheck_LFComand__Get__Param(TabAnumCmd[index],&Param,&exception);
    WuAutoCheck_LFComand__Get__ID(TabAnumCmd[index],&ID,&exception);

    printf("Command = _%d_ and StrVal = _%s_ and NumVal = _%d_ and Param = _%d_ and ID = _%s_\n",Command,StrVal,NumVal,Param
    ExecuteAnumCmd(Command,StrVal,NumVal,Param,ID); //command anum

}
```

On remarque que dans cette fonction on vient récupérer les paramètres utiles à l'exécution de cette fonction (index pour le switch, nom de la commande, nombre de trames émises, intertrame, ID). J'ai vérifié avec un printf que les valeurs sont celles attendues, c'est le cas. Ensuite j'ai été voir l'implémentation de la fonction ExecuteAnumCmd exécutée en dernier dans cette fonction :

```
        break;
    }
    case 8:
    {

    /*  strcpy(aCommand, "SENDF");
        sAPILFRF.ByteIn1=(unsigned char)numVal;
        sAPILFRF.ByteIn2=(unsigned char)param;
        strcpy(sAPILFRF.StringIn1, strVal);
        //printf("sAPILFRF.StringIn1 = %s\n",sAPILFRF.StringIn1);
        cAPI=myFUnc(aCommand, (void*)&sAPILFRF);*/

        strcpy(aCommand, "SENDF");
        sAPILFRF.ByteIn1=(unsigned char)numVal;
        sAPILFRF.ByteIn2=(unsigned char)param;
        strcpy(sAPILFRF.StringIn1, strVal);
        strcpy(sAPILFRF.StringIn2, ID);
        cAPI=myFUnc(aCommand, (void*)&sAPILFRF);

        if (cAPI!=0)
        {
        printf("error\n");
        return;
        }
```

Dans cette fonction il y a des « case » pour les différentes commandes ANUM. En faisant des tests j'ai remarqué que le case 8 correspond au « case » des trames individuelles. En investiguant j'ai trouvé le document IPC.h dans lequel on a les infos suivantes :

```
//---------------------------------------------------------------------------
// INTERFACE DEFINITION
//---------------------------------------------------------------------------
typedef unsigned char __stdcall (*TAPI)(char *Command, void *Structure);

//---------------------------------------------------------------------------
// STRUCTURES DEFINITION
//---------------------------------------------------------------------------
// Structure for all ANUMLFRF commands
#define API_STRING_SIZE 1024
struct TAPILFRF
{
    char          StringIn1[API_STRING_SIZE];
    char          StringIn2[API_STRING_SIZE];
    char          StringIn3[API_STRING_SIZE];
    char          StringIn4[API_STRING_SIZE];
    unsigned char ByteIn1;
    unsigned char ByteIn2;
    unsigned char ByteIn3;
    unsigned char ByteIn4;

    char          StringOut1[API_STRING_SIZE];
    char          StringOut2[API_STRING_SIZE];
    char          StringOut3[API_STRING_SIZE];
    char          StringOut4[API_STRING_SIZE];
    unsigned char ByteOut1;
    unsigned char ByteOut2;
    unsigned char ByteOut3;
    unsigned char ByteOut4;
};
```

```
   SENDF : Send a frame (yet defined in the list)
   ----------------------------------------------
       IN  : StringIn1  = Frame name
             StringIn2  = 1st parameter (if defined)
             StringIn3  = 2nd parameter (if defined)
             StringIn4  = 3rd parameter (if defined)
             ByteIn1    = Number of frames  [1..255]
             ByteIn2    = Interframe [0..255]ms
       OUT : -
       RET : OK or ERROR if the frame is not defined
```

On peut donc remarquer que le code du « case 8 » pour les trames individuelles est cohérent avec ceux qui est écrit dans ce fichier API/IPC.

À partir de là j'ai fait une série de tests pour comprendre si ces commandes étaient envoyée, reçu par la WU… .Ci-dessous la liste des tests effectués :

- Changer dans la database l'ID de 4 octets par celui de 3
- Ajouter ou enlever le h à la fin de l'id
- Coder en dur l'id sur 4 et 3 octets
- Envoyer les commandes indiv puis regarder avance un user1 si elles ont été prise en compte, ce qui voudrait dire que c'est la récupération des trames RF qui ne serait pas bonne
- Affiché les paramètres à différents endroits, vérifier leurs tailles etc...
- Testé avec le .exe anum du projet d'envoyer une trame individuelle. Ce qui a fonctionné.

Ensuite j'en ai parlé à Rémi et il est venu me donner un coup de main. On a donc refait tout ça avec Rémi pour que je lui explique ou j'en étais et qu'ils puissent avoir une idée claire de comment ça fonctionne. Ensuite Rémi m'a expliqué plus exactement comment anum concatène l'ID et s'est rappelé que les fonctions utilisées sont les mêmes que celles utilisées pour les scripts dans ANUM. Or dans ANUM on ne peut pas choisir l'ID dans les scripts, il faut les coder en dur.

On a ensuite eu l'idée de coder en dur l'ID dans la config ANUM et voir si quand on appelle la trame individuelle ça fonctionne, oui ! J'ai eu l'idée de laisser le @ID(24) avant l'ID codé en dur :

```
L2=Name:LFD_Ind_Force_MP¬Freq:125.0¬Content:0000h 111000101100110010b FADEh 31h @ID(24) 819ADFh-
```

Et on voit bien les trames RF que la WU a répondues à cette trame LF. Or on a laissé la ligne qui attribue l'ID dans la structure TAPILFRF. Conclusion : cette ligne ne fait rien, le champ @ID(24) n'est pas remplacé et est même ignoré (taille 0). Il est donc logique que la WU n'ait jamais renvoyé de trame RF puisque la trame LF reçu n'était pas de la taille attendue. On comprend donc que la partie paramètre de la fonction SENDF n'a pas été implémentée.

Plusieurs solutions nous sont venues en tête :

- Trouver le fichier source de la DLL et implémenter cette fonction. L'idée est de concaténer la variable ID qui est bien stocké dans l'attribut StringIn2 de la structure TAPILFRF qui correspond au paramètre 1, avec le reste de la trame LF. Autrement dit de remplacer le @ID(24) par cette valeur
- Si on n'a pas accès à la DLL peut-être que quelqu'un peut l'implémenter même si peu probable.
- Enfin la solution de secours est de créer une trame individuelle par commande et par ID, exemple :
  L2=Name:LFD_Ind_Force_MP_ID1 et L3=Name:LFD_Ind_Force_MP_ID2
  Et ça pour chaque commande

  Bonus : j'ai retrouvé une partie du rapport de Carolina où elle explique qu'elle a eu à modifier une DLL avec une plateforme .NET. C'est peut-être une piste pour corriger ce problème :

Un concept très important dans cette application est l'utilisation de dll (Dynamic-Link Library). La dll est un code en langage intermédiaire généré par Visual Studio. Le fichier en format .dll est une bibliothèque qui peut contenir du code, des données et des ressources et elle peut être utilisé pour plusieurs programmes en même temps. Autrement dit, des différents programmes peuvent avoir accès au code du projet en Visual Studio. Les codes de la dll sont accessibles grâce à une plateforme appelé .NET qui est la responsable de la gestion d'accès d'un programme à l'autre.

Cette interface utilise plusieurs dlls, mais une en particulier (nommé WuAutoCheck.dll) a été construite pour créer le dossier et les fichiers résultats et séparer les commandes ANum de la commande du banc. Ce programme est codé en C# sur Visual Studio. J'ai dû modifier ce code à deux moments. Pour ça, j'ai pris une semaine pour faire connaissance de la langage C# et du code.

Une des exigences du projet était l'introduction d'un tag (avec n'importe quelle nom). Ce tag sert à déterminer quand l'opérateur veut commencer la vérification et quand il veut la finir. Par exemple, si dans le fichier test script il y a deux tags, sur le fichier expected results il doit écrire quels sont les tests de vérification qu'il souhaite faire entre ces deux tags. Cependant, quand la dll lisait le TestCase, le programme ne reconnaissait pas le tag, car elle ne fait pas partie de liste de commandes. Donc, j'ai modifié le code de façon que le tag soit transparent. Comme toutes les commandes font partie d'une liste et qu'elles ne sont pas modifiables, tout ce qu'il n'est pas dans cette liste de commande est un Label, ANNEXE C : Code C#. Dans l'ANNEXE D : Error.Log, vous pouvez trouver un exemple du fichier ErrorLog.txt généré par la dll. Ce fichier est censé faire un suivi des erreurs qu'il peut avoir pendant la lecture des fichiers, et en cas d'une erreur de syntaxe, par exemple, les commandes ne sont pas exécutées et le test sera « Error ».