07/05/2021

IHM

A l’intention de Tachysséma

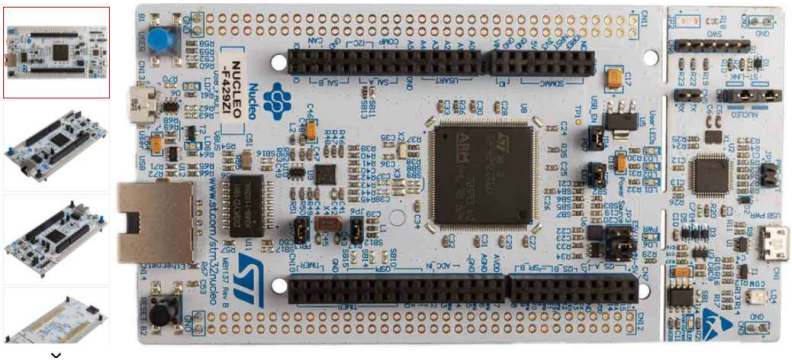
L’équipe IHM

Introduction

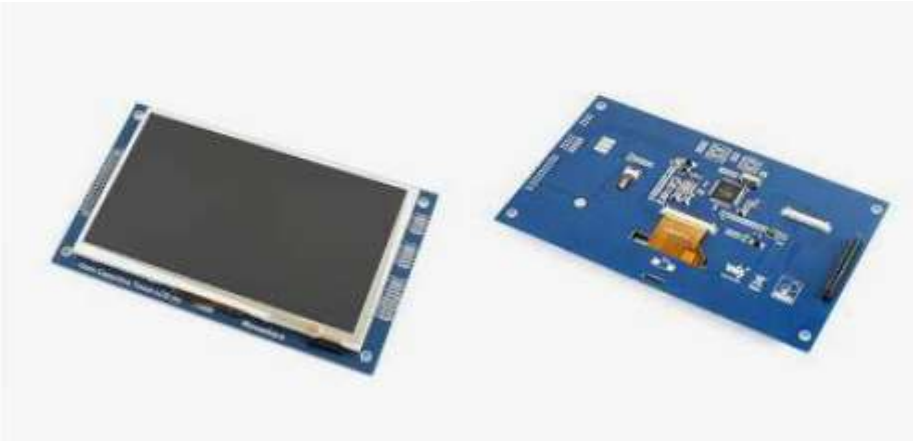
Dans ce court document l'équipe IHM, présentera le matériel utilisé pour la réalisation de l'IHM, les configurations ST nécessaires pour son fonctionnement, ainsi qu'une explication des différentes fonctions qui ont été rajoutées pour son utilisation.

Equipements utilisés

Carte de développement NUCLEO-F429ZI



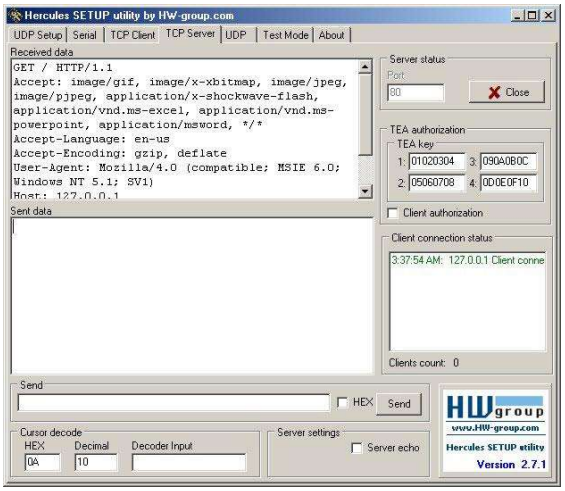
Waveshare 7inch Capacitive Touch LCD (C) 800x480



STM32CubeIDE

2 Câbles micro USB-USB mâles et des câbles mâles-femelles (Jumper)

Hercules SETUP utility



PC

Configurations ST

Bien que l'ensemble des configurations soit trouvable sur un « ST report » de notre projet, nous allons revenir sur les principales.

CRC : Le contrôle de redondance cyclique (CRC) est une technique utilisée pour détecter les erreurs dans les données numériques, mais sans apporter de corrections lorsque des erreurs sont détectées. Il est utilisé dans la transmission de données ou la vérification de l'intégrité du stockage des données.

DMA2D : Le DMA2D est spécialisé dédié à la manipulation d'images, de l'écran LCD. Il agit comme un contrôleur de mémoire statique flexible (FSMC) et est utilisé pour accéder à l'écran LCD-TFT.

FMC : Nécessaire pour interfacer l'écran, c'est un contrôleur de mémoire NOR/PSRAM.

I2C : Cela permettra de créer un descripteur de typedef I2C. Et utiliser les fonctions de l'API I2C.

NVIC : Gère les priorités entre interruptions, la sauvegarde et la restauration des registres.

RCC : Le périphérique RCC est utilisé pour contrôler les périphériques internes, ainsi que les signaux de réinitialisation et la distribution d'horloge.

RTC : C'est une horloge en temps réel (**R**eal **T**ime **C**lock) est un élément de chronométrage dédié à la conservation de l'heure. Ou à la réalisation d'action à des instants précis.

SPI1 : Cela permettra de créer un descripteur de typedef SPI. Et utiliser les fonctions de l'API SPI.

STMicroelectronic TouchGFX : Afin de faciliter la création de l'interface homme machine vous utilisez ce GUI Graphical User Interface, qui vous générera du code représentatif des widgets, des images et pages de l'écran.

USB_OTG_FS : L'interface USB « On-the-go » « Full speed », permet de considérer la carte nucleo comme un hôte et d'utiliser sa liaison USB à 12 Mbits/s.

Descriptions des fonctions de la communication ECRAN-STM32

L'ensemble de ces fonctions se trouve dans TouchGFX > gui > include, dans les dossiers gui vous trouverez les librairies pour chaque affichages (screens) et dans les dossiers src vous trouverez les fichiers sources associés aux screens.

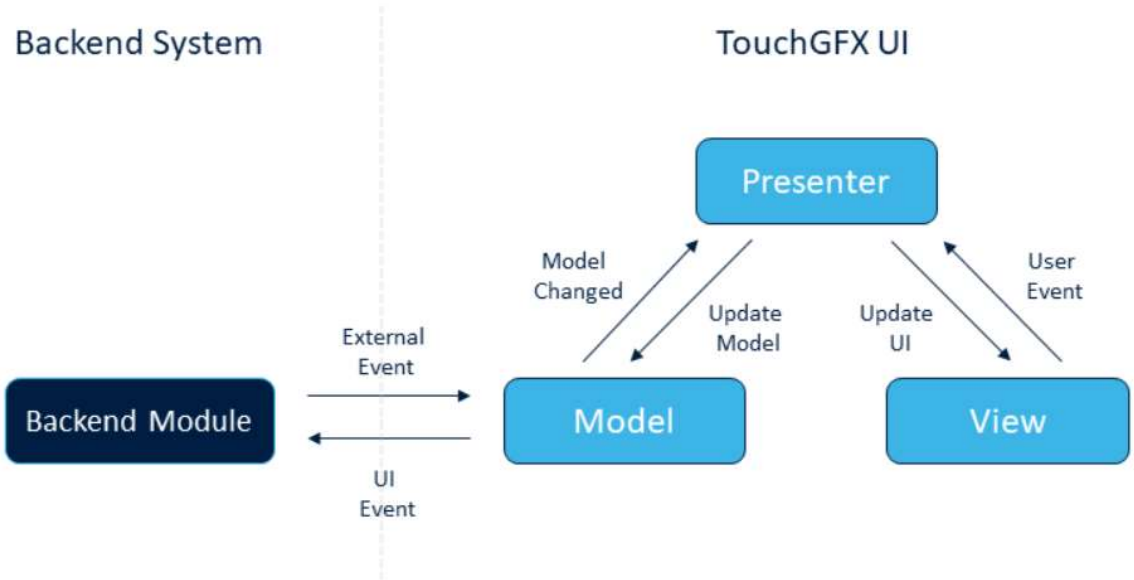
Model

Dans le Model.C, on récupère les variables globales du projet les données envoyées et reçus de la génération commençant par **IHM_G** et de la réception commençant par **IHM_R.**

On y déclare des accesseurs commençant par **setRec** et des mutateurs commençant par **setGen,** pour communiquer avec les variables de l'IHM.

Principes du Model-View-Controller

Pour interagir avec le main on utilise le model, pour communiquer avec ce dernier on utilise le presenter et enfin pour communiquer avec le presenter on utilise la view.
La view contient les parties de codes utilisés, pour réaliser les fonctions communicante avec les screens affichés.



Screen Analyse

Pour communiquer avec les fonctions du modèle dont les accesseurs on fait model-> nomFonctionduModèle.

Dans les presenters des screens analyses on y fait des nouveaux accesseurs pour que la view accède aux variables de l'IHM.

Dans les views on initialise une demande d'observation des variables de l'IHM du bus ou du flux (IHM_R_commencer, IHM_R_data_dispo_DVI ou IHM_R_data_dispo_SDI à 1) dans la fonction d'initialisation de l'affichage setupScreen().
On y a aussi initialisé une variable cpt représentatif d'un écoulement de temps, au bout d'à peu près 30 secondes on y modifie les variables. Cela a été fait ainsi pour éviter l'affichage de valeurs non modifiés.

Screen Génération Bus

Pour communiquer avec les fonctions du modèle dont les mutateurs on fait model-> nomFonctionduModèle.

Dans les presenters des screens analyses on y fait des nouveaux accesseurs pour que la view accède aux variables de l'IHM.

Dans les views des screens de générations il y a des préprocesseurs de nettoyage qui renvoie aux fonctions clean pour effacer les lignes ainsi nettoyage1 correspondra à clean1() ; .

Par la fonction init_val() à chaque initialisation des pages il y a une remise à 0 des valeurs précédemment entrées.

Les fonctions l1_ renvoient aux boutons de récupération d'un appui sur la ligne 1, le numéro ou mot suivant cet entête permet d'identifier les boutons associés, er pour effacer avec le logo < et enter pour entrer avec le logo v.

Les fonctions l2_ renvoient aux boutons de récupération d'un appui sur la ligne 2. l3_ ligne 3 et l4_ ligne 4 sur les mêmes principes.

A chaque ligne est associé un buffer (un tableau) qui s'incrémente par la variable (ilNumligne) à chaque appuie d'un bouton et qui se réinitialise à 0 en cas d'effacement, ou de dépassement de la taille du buffer ou d'appuiement d'entrer.

l1_enter, l2_enter, l3_enter, l4_enter, renvoie les valeurs des buffers (des chiffres saisies par lignes) les converties en nombres aux mutateurs, accédant aux variables de l'IHM.

Exemple d'utilisation :
Voir vidéo du PowerPoint jointe à ce document.

Descriptions des fonctions de la communication PC-STM32

L'ensemble de ces fonctions se trouve dans USB_DEVICE > APP, dans les fichiers pcstm32.c et pcstm32.h

void confirmationCMD(uint8_t cmd, **void*** str);
Cette fonction reçoit un chiffre, lui indiquant quelle commande a tapé l'utilisateur, s'il a tapé genFlux il recevra 1 dans cmd, void* str permet la récupération de n'importe quel type de variable dans le cas d'utilisation de la commande genFlux, il recevra le type de la structure du flux en génération (s_gen_flux), c'est-à-dire la résolution, le type de mire et le standard. Il affichera à l'utilisateur la signification de sa commande.

void envoiePCSTM(uint8_t* Buf, uint16_t Len);
Cette fonction nous permet d'envoyer des informations du STM32 au PC, il reçoit la chaine de caractère Buf et sa taille Len. Puis il utilise une fonction de la librarie « sbd_cdc_if.h » pour transmettre la chaine de caractère Buf via USB au PC. Un délai y a été ajouté pour que lors d'envoi de plusieurs données rapidement le logiciel de réception Hercules ne plante pas.

void s_gen_flux_config(s_gen_flux* gf);
Cette fonction sert à paramétrer les informations qu'on veut générer, si un utilisateur saisi genFlux dans le buffer_verif qui est le buffer de toutes les données envoyer depuis Hercules, alors on copie la chaine envoyée dans un buffer secondaire nommé test. Si les paramètres sont saisis correctement c'est-à-dire qu'après un -r ou -symbole on a bien retrouvé la valeur désirée. On continue la vérification de toutes les commandes saisies, si la valeur souhaitée n'a pas été retrouvé on renvoie erreur.

void s_gen_bus_config(s_gen_bus* gb);
Sous le même principe, cette fonction sert à paramétrer les informations qu'on veut générer, si un utilisateur saisi genBus dans le buffer_verif qui est le buffer de toutes les données envoyer depuis Hercules, alors on copie la chaine envoyée dans un buffer secondaire nommé test. Si les paramètres sont saisis correctement c'est-à-dire qu'après un -r ou -symbole on a bien retrouvé la valeur désirée. On continue la vérification de toutes les commandes saisies, si la valeur souhaitée n'a pas été retrouvé on renvoie erreur.

void s_rec_bus_config(s_rec_bus* rb);
Cette fonction sert à indiquer qu'on veut recevoir, la fréquence, l'octet et le mot binaire du bus de communication. On copie la chaine de caractère du buffer_verif dans dr(une chaine de caractère d'observation), et si un utilisateur saisi recBus, cela signifie qu'il veut les informations de la structure passé en paramètre. La fonction appel confirmationCMD pour exécuter l'ordre associé à cette commande recBus, qui est l'affichage des paramètres de la structure s_rec_bus* rb.

void s_rec_flux_config(s_rec_flux* rf);
Cette fonction sert à indiquer qu'on veut recevoir, la résolution (largeur et hauteur), le blanking (horizontal et vertical) ainsi que le framerate du flux vidéo. On copie la chaine de caractère du buffer_verif dans dfl(une chaine de caractère d'observation), et si un utilisateur saisi recFlux, cela signifie qu'il veut les informations de la structure passé en paramètre. La fonction appel confirmationCMD pour exécuter l'ordre associé à cette commande recFlux, qui est l'affichage des paramètres de la structure s_rec_flux* rf.

void help(**void**);
Cette fonction envoie au PC la description de toutes les commandes, si dans le buffer_verif(dans Hercules), un utilisateur a tapé help.

void clear_buffer(uint8_t * buffer_verif);
Cette fonction affecte la valeur de base '\0' à notre buffer_verif en faisant ceci il vide le buffer_verif.

Des exemples d'utilisation

Voir vidéo du PowerPoint jointe à ce document.