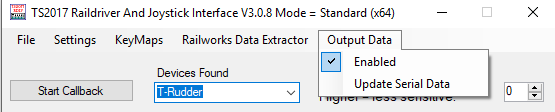
**Sortie des données vers les ports série**

Depuis la version 3.0.8, vous pouvez désormais envoyer la majorité des données affichées dans la fenêtre de superposition en mode avancé, vers n'importe quel port série relié à votre PC. Cette opération peut être effectuée dans l'un ou l'autre mode.

Pour ce faire, les éléments suivants ont été ajoutés au programme.

1.

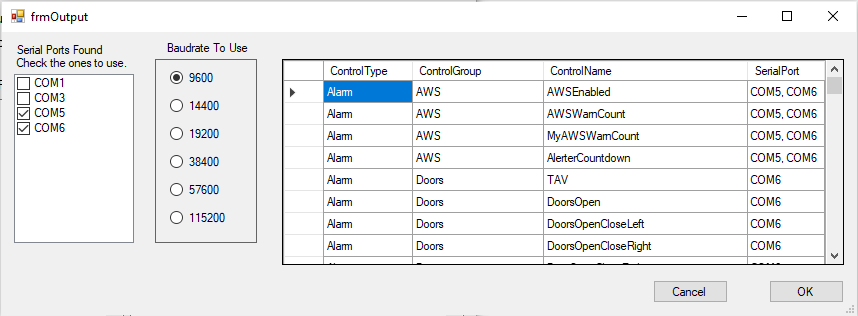


Si le programme détecte des ports série connectés, il ajoutera l'option de menu « Output Data » (Données de sortie) montrée ci-dessus.

Vous devez ensuite cocher la case à côté de «  Enabled » pour afficher l'option de menu « Update Serial Data », qui est utilisée pour attribuer diverses commandes à chaque port série (voir 2 ci-dessous).

Si vous ne cochez pas l'option de menu «  Enabled », rien ne sera envoyé aux ports série.

2. En cliquant sur l'option « Update Serial Data », vous obtiendrez les informations suivantes



La première boîte à gauche affiche une liste de tous les ports série présents sur votre PC.

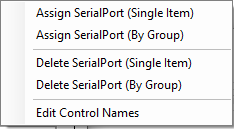
Vous devez cocher ceux vers lesquels vous souhaitez envoyer des données.

La deuxième case permet de sélectionner le débit en bauds à utiliser pour les ports série. J'ai constaté que les communications entre le PC et l'Arduino étaient correctes en utilisant la valeur par défaut de 9600, mais que des erreurs pouvaient apparaître sur l'écran LCD lorsque l'on augmentait la vitesse de transmission.

La dernière case montre la liste des contrôles tirés du fichier ControlNames.txt qui est utilisé tout au long du programme.

Vous avez le :  
**ControlType** qui est Alarm/Lever/Gauge etc., puis le   
**ControlGroup** qui est Reverser/Throttle/TrainBrake/Doors etc., ensuite vous avez   
**ControlName** qui est le nom réel du contrôle dans le train que vous conduisez et enfin vous avez le(s)   
**port(s) série** vers lequel (lesquels) les données seront envoyées.

Si vous cliquez avec le bouton droit de la souris sur l'une des lignes, le menu suivant s'affiche

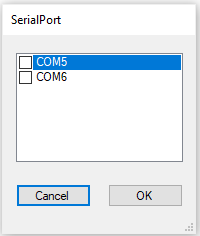


**Assign SerialPort (Single Item)** vous permet de sélectionner un ou plusieurs ports série pour cette ligne uniquement.

**Assign SerialPort (By Group)** vous permet de sélectionner un ou plusieurs ports série pour toutes les lignes qui ont le même ControlGroup que la ligne sur laquelle vous avez cliqué, c'est-à-dire comme dans toutes les AWS assignées à COM5 et COM6 dans l'image 2.

Les deux options de suppression de port série font la même chose mais suppriment le(s) port(s) série(s) attribué(s).

Lorsque vous cliquez sur l'une des options d'affectation ci-dessus, la fenêtre suivante s'affiche



La fenêtre ne répertorie que les ports série que vous avez cochés au point 2 ci-dessus, il vous suffit de sélectionner le(s) port(s) requis et de cliquer sur OK. Tous les réglages et affectations sont sauvegardés et prêts à être rechargés à chaque fois que le programme est exécuté.

La dernière option du menu ci-dessus est « E**diter les noms de contrôle** », qui fonctionne de la même manière que l'option de menu « **Réglages - Editer les noms de contrôle** ».

Vous devez veiller à conserver le même format, sinon le programme échouera. Le format est le suivant : **ControlType=ControlGroup=ControlName=DisplayName=DisplayUnits** (MPH/KPH, PSI/Bar, Gallons/Litres etc.).

Le nom d'affichage et les unités d'affichage ne sont utilisés que dans la superposition qui s'affiche lorsque vous exécutez le programme en mode avancé. Ils peuvent être omis si vous utilisez le mode standard. Voici quelques exemples

**Levier=Renverseur=Renverseur virtuel  
Jauge=Pression chaudière=JaugePSI=Pression chaudière=PSI  
Alarm=Sifa=SiFaWarning=Sifa.**

Une fois que vous avez terminé la mise à jour des données, cliquez sur OK dans l'écran frmOutput au point 2 ci-dessus et lorsque vous cliquez sur le bouton « **Start Callback** », les données sélectionnées seront envoyées aux ports série correspondants.

Les données envoyées sont au format **<Groupe de contrôle:Nom de contrôle:valeur>**.

Par exemple, <Reverser:VirtualReverser:0.5> nous utilisons < et > pour indiquer le début et la fin des données et : pour diviser l'information.

Limite de vitesse suivante/limite de vitesse précédente ajoutée dans la V3.1.2

Depuis la version 3.1.2, si vous utilisez le mode avancé et que vous mettez l'inverseur en marche arrière, les données de **NextSpeedLimitBackSpeed** et **NextSpeedLimitBackDistance** seront envoyées au port série au lieu de **NextSpeedLimitSpeed** et **NextSpeedLimitDistance** mais avec un nombre négatif pour montrer que c'est la limite de vitesse derrière qui est affichée.

Si vous passez à la cabine arrière, les limites de vitesse devraient s'intervertir, de sorte que la limite de vitesse derrière devienne la limite de vitesse devant.

Malheureusement, cela ne fonctionne pas pour toutes les locomotives, c'est pourquoi j'ai inclus une macro appelée « **MACRO\_TOGGLE\_SPEEDLIMIT** » qui permute les limites de vitesse avant/arrière à chaque appel.

Pour l'utiliser, il suffit d'éditer la carte des boutons, de sélectionner « **Add New Command** » dans le menu du clic droit, de donner un nom à la case **Control Name** (j'ai utilisé « **Switch Speed Limits** ») et de sélectionner « **MACRO\_TOGGLE\_SPEEDLIMIT** » dans la case Key, c'est le dernier élément de la liste.Cela revient à appuyer sur les touches Shift+Alt+D lorsque la superposition est affichée.

Les données envoyées au port série seront les suivantes.Pour la limite de vitesse à venir,

**SpeedLimit:NextSpeedLimitSpeed:50.00** et **SpeedLimit:NextSpeedLimitDistance:50**

Et pour la limite de vitesse derrière,

**SpeedLimit:NextSpeedLimitSpeed:-50.00** et **SpeedLimit:NextSpeedLimitDistance:-50**

J'ai fourni un fichier script Arduino appelé « **ArduinoFromTS2019.ino** » dans le dossier « ArduinoFromTS2019 » qui démontre comment utiliser ces données pour allumer des leds et afficher la vitesse, les vitesses, l'inversion et les portes sur un lcd 20x4 I2c. J'ai placé une copie du code à la fin de ce document à partir de la page 4.

Voici une image du programme en action



Il se peut que certaines données ne donnent pas les résultats escomptés.

C'est le cas, par exemple, de **l'AWS**.Si vous voulez avoir un spectacle lumineux lorsque l'alarme **AWS** se déclenche, utilisez le groupe **Alarm=AWS**, mais si vous voulez qu'une lumière fonctionne de la même manière que le tournesol, vous devez utiliser le groupe **Warning=Sunflower=AWS** que j'ai ajouté au bas du fichier **ControlNames.txt**.

J'ai également ajouté deux autres avertissements au fichier

**Avertissement=DRA=DRA**

**Warning=DRA=DRAButton**

Cela vous permettra d'avoir une lumière led lorsque le DRA est allumé.

**Si vous constatez qu'une fonction ne s'affiche pas lorsque vous affectez les ports série, procédez comme suit.**

1. Lancez TS et le programme et conduisez la locomotive.

2. Mettez TS en pause puis Alt+Tab sur mon programme et faites défiler le panneau de droite pour voir si vous pouvez voir le nom de la commande.

3. Si c'est le cas, passez au point 5.

4. Si vous ne connaissez pas le nom du contrôle que vous recherchez, le moyen le plus simple de le trouver est de sélectionner tout le texte dans le panneau et de le copier dans un éditeur de texte comme Notepad++, puis de mettre TS en pause, de déplacer le contrôle, de mettre à nouveau en pause et de copier le texte mis à jour dans le panneau de droite dans une deuxième fenêtre de Notepad++. Comparez ensuite les deux fichiers, en notant les contrôles qui ont changé, ce qui devrait vous donner le nom du contrôle.

5.Utilisez maintenant l'option Paramètres - Modifier les noms de contrôle et ajoutez le contrôle au bas de la liste.

Le ControlType et le ControlGroup peuvent être ce que vous voulez, j'ai seulement utilisé Warnings parce que cela le place en bas de la liste et le rend plus facile à trouver et j'ai utilisé Sunflower parce que cela rend évident ce que le contrôle AWS contrôle. Veillez cependant à respecter le format.

6. Sélectionnez maintenant l'option de menu Output Data (Données de sortie) - Update Serial Data (Mise à jour des données série) et attribuez le port série que vous souhaitez utiliser. Vous pouvez trier les colonnes en cliquant sur leur nom.

**Voici le code pour l'Arduino.**

/\*

\* Ceci est un sketch de test pour un Arduino qui permet d'autoriser les données reçues via son port série.

\* du programme « TSClassic Raildriver And Joystick Interface » pour contrôler des LEDs et des écrans LCD.

\* des LEDs et des écrans LCD. Les données sont au format

\* « <ControlGroup:ControlName:value> ».

\* Par exemple, <Reverser:VirtualReverser:0.5>.

\* Nous utilisons < et > pour indiquer le début et la fin des données et : pour diviser l'information.

\*/

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

const byte AwsDRA\_Pin = 13;

const byte Doors\_Pin = 2;

const byte Dsd\_Pin = 3;

const byte Emergency\_Pin = 4;

const byte Engine\_Pin = 5;

const byte Handbrake\_Pin = 6;

const byte Sifa\_Pin = 7;

//Here we add the details regarding the lcd screen width/height

const byte lcdWidth = 20;

const byte lcdRows = 4;

//Buffer to hold data from serialport

//data[0] = controlgroup

//data[1] = controlname

//data[2] = value

char data[3][80];

// byte data to hold the position of the current char in the data array.

byte cnt = 0;

byte pos = 0;

//Marker set when start of data received

boolean started = false;

//Marker set when all data received

boolean finished = false;

//Set I2C LCD Address

byte Lcd\_Address = 0x27;

//Initialise the lcd.

LiquidCrystal\_I2C lcd(Lcd\_Address, 20, 4);

char ButtonMap[80];

char LocoName[80];

void setup() {

// Set the pins

pinMode(AwsDRA\_Pin, OUTPUT);

pinMode(Doors\_Pin, OUTPUT);

pinMode(Dsd\_Pin, OUTPUT);

pinMode(Emergency\_Pin, OUTPUT);

pinMode(Engine\_Pin, OUTPUT);

pinMode(Handbrake\_Pin, OUTPUT);

pinMode(Sifa\_Pin, OUTPUT);

Serial.begin(9600);

//Setup lcd

lcd.begin();

lcd.backlight();

lcd.clear();

//Display a message to the user to show the code is working

// and waiting for data.

lcd.setCursor(0,0);

lcd.print("Waiting for TS2019");

}

void loop() {

// Check we have something in the serial buffer

while(Serial.available() > 0) {

char nextChar = Serial.read();

if(nextChar == '<') { //Check for start of command marker

started = true;

finished = false;

pos = 0;

cnt = 0;

//Reset data to nothing

data[0][0] = '\0';

data[1][0] = '\0';

data[2][0] = '\0';

}

else if(nextChar == '>') { //End of command marker

finished = true;

break; //We have our data so exit the loop

}

else if(nextChar == ':') { //End of current string

pos++;

cnt = 0;

}

else {

data[pos][cnt] = nextChar; //Store the incoming data one char at a time

cnt++;

data[pos][cnt] = '\0'; //Null terminate the data array;

}

} //End of While loop

if(started && finished) { //All data for the current command received.

UpdateHardware();//Activate switches/displays etc

}

}

void UpdateHardware() {

//Check if loco has changed, if so clear the display.

if(!strcmp(data[0],"locochanged")){

// Clear the lcd screen

lcd.clear();

//Turn all led's off

digitalWrite(AwsDRA\_Pin, LOW);

digitalWrite(Doors\_Pin, LOW);

digitalWrite(Dsd\_Pin, LOW);

digitalWrite(Emergency\_Pin, LOW);

digitalWrite(Engine\_Pin, LOW);

digitalWrite(Handbrake\_Pin, LOW);

digitalWrite(Sifa\_Pin, LOW);

strcpy(ButtonMap, data[1]);

strcpy(LocoName, data[2]);

return;

}

//Convert value to an int.

int value = atoi(data[2]);

//Convert value to a float.

float fvalue = atof(data[2]);

//Used to hold how many spaces required at end of a line for the lcd

byte padding = 0;

//Temporary variable used when formatting output string for the display

char tmp[30];

//Although we have both sunflower and DRA set to the same pin, because in the program

//I have them assigned to different serialports, only the command assigned to this

//Arduinos port will be activated, this allows the same code to be used on multiple Arduinos.

if( !strcmp(data[0],"Sunflower") || !strcmp(data[0], "DRA")) {

if(value == 1) {

digitalWrite(AwsDRA\_Pin, HIGH);

} else if(value == 0) {

digitalWrite(AwsDRA\_Pin, LOW);

}

}

//Here we check to see if the controlgroup is "Speed", if it is, display it

else if( !strcmp(data[0], "Speed")){

lcd.setCursor(0,0);

//The sprintf() function in the Arduino does not allow the use of float variables

// so we need to use the dtostrf() function to convert the value to a string.

char newVal[7];

dtostrf(fvalue, 4, 1, newVal);

sprintf(tmp,"%s = %s", data[0], newVal);

padding = lcdWidth - strlen(tmp);

lcd.print(tmp);

for(int i = 0; i < padding; i++) {

lcd.print(" ");

}

}

//Here we move the cursor to line 1 and display the controlgroup "Gears"

else if( !strcmp(data[0], "Gears")) {

lcd.setCursor(0,1);

sprintf(tmp, "%s = %d", data[0], value);

padding = lcdWidth - strlen(tmp);

lcd.print(tmp);

for(int i = 0; i < padding; i++) {

lcd.print(" ");

}

}

//Here we move the cursor to line 2 and display the controlgroup "Reverser"

//We use the float value and multiply it by 100 to get the % value. We then

//add the % sign to the end.

else if( !strcmp(data[0], "Reverser")) {

lcd.setCursor(0,2);

int i = fvalue \* 100;

sprintf(tmp, "%s = %d%%", data[0], i);

padding = lcdWidth - strlen(tmp);

lcd.print(tmp);

for(int i = 0; i < padding; i++) {

lcd.print(" ");

}

}

//Here we display text to show if the doors are open/closed

else if( !strcmp(data[0], "Doors")) {

lcd.setCursor(0,3);

if(value >= 1)

sprintf(tmp, "%s Open", data[0]);

else if(value == 0)

sprintf(tmp, "%s Closed", data[0]);

padding = lcdWidth - strlen(tmp);

lcd.print(tmp);

for(int i = 0; i < padding; i++) {

lcd.print(" ");

}

}

}

28/01/20

The program now outputs the name of the ButtonMap without the .xml extension and the LocoName, whenever the loco changes. To enable this I have added the code

char ButtonMap[80];

char LocoName[80];

in the main declaration section and then in the ‘if(!strcmp(data[0],"locochanged"))’ section of the UpdateHardware() function I added

strcpy(ButtonMap, data[1]);

strcpy(LocoName, data[2]);

When the program detects a new loco it sends”<locochanged:ButtonMap:LocoName> “so

Data[0] = “locochanged”, data[1] = name of the current ButtonMap, data[2] = name of current loco.

16/04/20

You can now output the same data to more than one serial port.