/\*

The MIT License (MIT)

Copyright (c) 2022 Robert E Bridges

Permission is hereby granted, free of charge, to any person obtaining a copy of

this software and associated documentation files (the "Software"), to deal in

the Software without restriction, including without limitation the rights to

use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of

the Software, and to permit persons to whom the Software is furnished to do so,

subject to the following conditions:

The above copyright notice and this permission notice shall be included in all

copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR

IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS

FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR

COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER

IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN

CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

On a personal note, if you develop an application or product using this library

and make millions of dollars, I'm happy for you!

\*/

/\*

Code by Robert E Bridges bob@bricomp-uk.com

This library is intended to be used to create your own Nextion Library. Most of it is done for you.

The function that you will mostly alter is the "respondToReply()" function.

I developed this library to control the valves in my Home Heating system, so there are functions

that pertain to the opening/closing of valves. This can be used as an example as to how to use/develop

the Library.

I mostly communicate with the nextion through the passing of data into/from numeric variables.

I have a TimerEvent which runs at 1 second intervals, slow I know but fast enough for my current needs.

When, for example this timer notices that the numeric variable "SetTime.val" is not zero it takes the value

from this variable and sets the time. The format of the data in this variable is (in HEX) "HHMMSS".

After having set the time the variable is set back to 0 again.

Other variables are interrogated and responded to in a similar way by the code for this Timer Event.

An example is to give an impressionof a flashing led, turning on or off a radio button with a different

colour for on and off.

Below is the Nextion code snippet to set the RTC time.

//=================================

//Set RTC time if SetTimeVar >0

//=================================

if(page0.SetTime.val!=0)

{

xx.val=page0.SetTime.val

xx.val=xx.val>>16

rtc3=xx.val // Set the hour

xx.val=page0.SetTime.val

xx.val=xx.val&0xFF00

xx.val=xx.val>>8 // Set the minutes

rtc4=xx.val

xx.val=page0.SetTime.val&0xFF

rtc5=xx.val // Set the seconds

page0.SetTime.val=0

}

Revision Date Author Description

1.0 16/04/2022 Bridges Initial release

\*/

#ifndef NextionLib\_h

#define NextionLib\_h

#include "Arduino.h"

#pragma pack(push,1)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* These are all the data types used to communicate with the Nextion. More correctly \*

\* they are the data types for data returned FROM the Nextion display. \*

\* Some data returns only need 4 bytes, the Id and the Nextion terminating string, \*

\* \0xFF\0xFF\0xFF, whilst others require much more right up to the reset function \*

\* which returns two data sets in one go i.e. startUp message and ready message \*

\* |----- Start up message ----| |- Ready Message -| \*

\* 0x00 0x00 0x00 0xFF 0xFF 0xFF 0x88 0xFF 0xFF 0xFF \*

\* \*

\* All the comms are put int the variable nextionEvent which is of nextionEventType. \*

\* this consists of the Id of the message which is returned in nextionEvent.Id. The \*

\* remaining bytes are put into nextionEvent.reply3, or nextionEvent.reply4 etc. \*

\* The relevant reply type is examined to interpret the data. \*

\* Infact when data is returned from the Nextion it is placed in \*

\* nextionEvent.resetReply because this is the lartgest structure and can accomodate \*

\* all types of reply. \*

\* Note that there is sometimes the need to convert from little endian to big endian \*

\* due to Teensy and Nextion using different endians. \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

struct rep3Type {

uint32\_t nextTerm; // = 0xFFFFFF swap little endian to big endian = 0xFFFFFF00

};

struct rep4Type {

uint8\_t pageNum;

uint32\_t nextTerm;

};

struct rep5Type {

uint8\_t ans[2];

uint32\_t nextTerm;

};

struct rep6Type {

uint8\_t pageNum;

uint8\_t component;

uint8\_t pressed;

uint32\_t nextTerm;

};

struct rep7Type {

union {

uint8\_t ans[4];

uint16\_t num[2];

uint32\_t number32bit;

};

uint32\_t nextTerm;

};

struct rep8Type {

union {

uint8\_t x[2];

uint16\_t xPos;

};

union {

uint8\_t y[2];

uint16\_t yPos;

};

uint8\_t pressed;

uint32\_t nextTerm;

};

// After Reset Nextion Returns 00 00 00 FF FF followed by 88 FF FF FF

struct resetReplyType { // first 00 in nextionEvent char Id

uint32\_t startup4Bytes; // 00 00 FF FF swap little endian to big endian = 0x0FFFF0000

uint8\_t startupByte; // FF

uint32\_t readyReply; // 88 FF FF FF swap little endian to big endian = 0x0FFFFFF88

uint32\_t overflow; // Just to allow a 4 byte buffer if extra erroneous bytes are

}; // sent during "reset" (Have Seen It in error conditions)

struct nextionEventType {

char id;

union {

rep3Type reply3;

rep4Type reply4;

rep5Type reply5;

rep6Type reply6;

rep7Type reply7;

rep8Type reply8;

resetReplyType resetReply; //-- The largest Type

uint8\_t data[sizeof(resetReplyType)]; // Just so that data can be annalysed for debug purposes

};

}; // nextionEvent;

#pragma pack(pop)

enum onOffFlashingType {

off = 0,

on, // = 1,

flashing // = 2

};

enum topMidBottmType {

top = 0,

mid, // = 1,

bottom // = 2,

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \*

\* This is an explanation of the data returned from the Nextion. \*

\* I think it's self explanatory, but then I wrote it!! \*

\* There is the Id returned by the Nextion, followed by the number of following bytes, \*

\* followed by an expanation of those bytes. It is only because we have this \*

\* information that this library was able to be written. All is based upon this info. \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* /-------- Id Codes Returned by Nextion

/

/ /----------- Number of Char/Bytes returned after Id Char/Byte

/ /

/ / /-------- Char/Bytes returned after Id Char/Byte

/ / /

/ / |-------------------------------------| \*/

const uint8\_t nextionStartUp = 0x00; // 5 0x00 0x00 0x00 0xFF 0xFF 0xFF Returned when Nextion has started or reset

const uint8\_t instructionSuccess = 0x01; // 3 0x01 0xFF 0xFF 0xFF (ONLY SENT WHEN bkcmd = 1 or 3)

const uint8\_t touchEvent = 0x65; // 6 0x65 0x00 0x01 0x01 0xFF 0xFF 0xFF Returned when Touch occurs

// data: Page 0, Component 1, Pressed Returns page, component and pressed or not,

// 0 or 1

const uint8\_t currentPageNumber = 0x66; // 4 0x66 0x01 0xFF 0xFF 0xFF Returned when the sendme command is used.

// data : page 1

const uint8\_t touchCoordinateAwake = 0X67; // 8 0x67 0x00 0x7A 0x00 0x1E 0x01 0xFF 0xFF 0xFF Returned when sendxy = 1 and not in sleep

// data: (122, 30) Pressed mode

const uint8\_t touchCoordinateSleep = 0X68; // 8 0x68 0x00 0x7A 0x00 0x1E 0x01 0xFF 0xFF 0xFF Returned when sendxy = 1 and exiting sleep

// data: (122, 30) Pressed (0 for NOT pressed)

const uint8\_t stringDataEnclosed = 0x70; // 0 means variable amount

// 0x70 0x61 0x62 0x31 0x32 0x33 0xFF 0xFF 0xFF Returned when using get command for string.

// data: ab123 Each byte is converted to char.

const uint8\_t numericDataEnclosed = 0x71; // 7 0x71 0x01 0x02 0x03 0x04 0xFF 0xFF 0xFF Returned when get command to return a number

// data: 67305985 4 byte 32 bit value in little endian Order.

const uint8\_t autoEnteredSleepMode = 0x86; // 3 0x86 0xFF 0xFF 0xFF Returned when Nextion enters sleep

// automatically.

// Using sleep = 1 will not return an 0x86

const uint8\_t autoAwakeFromSleepMode = 0x87; // 3 0x87 0xFF 0xFF 0xFF Returned when Nextion leaves sleep

// automatically.

// Using sleep = 0 will not return an 0x87

const uint8\_t nextionReady = 0x88; // 3 0x88 0xFF 0xFF 0xFF Returned when Nextion has powered up and is

// now initialized successfully

const uint8\_t powerOnMicroSDCardDet = 0x89; // 3 0x89 0xFF 0xFF 0xFF Returned when power on detects inserted

// microSD and begins Upgrade by microSD

// process.

const uint8\_t transparentDataFin = 0xFD; // 3 0xFD 0xFF 0xFF 0xFF Returned when all requested bytes of

// Transparent Data mode have been received,

// and is now leaving transparent data mode

// (see 1.16)

const uint8\_t transparentDataReady = 0xFE; // 3 0xFE 0xFF 0xFF 0xFF Returned when requesting Transparent Data

// mode, and device is now ready to begin

// receiving the specified quantity of data

// (see 1.16)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \*

\* Below are the error codes returned by the Nextion \*

\* Whether they are returned or not depends upon the value by the Nextion bkcmd. \*

\* This can be set to Level 0 ... to Level 3. Below are shown the bkcmd level at which \*

\* the error/state message is returned. The default is Level 2. \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Error/event codes (ONLY 0x01 is an event code)

/------------ Error/Event Code

/

/ /------- Error/Event Code returned when bkcmd equals value shown

/ /

/ |--------| \*/

const uint8\_t invalidInstruction = 0x00; // bkcmd 2,3 0x00 0xFF 0xFF 0xFF Returned when instruction sent by user has failed

//const uint8\_t instructionSuccess = 0x01; // bkcmd 1,3 0x01 0xFF 0xFF 0xFF (ONLY SENT WHEN bkcmd = 1 or 3 )

const uint8\_t invalidComponentId = 0x02; // bkcmd 2,3 0x02 0xFF 0xFF 0xFF Returned when invalid Component ID or name was used

const uint8\_t invalidPageId = 0x03; // bkcmd 2,3 0x03 0xFF 0xFF 0xFF Returned when invalid Page ID or name was used

const uint8\_t invalidPictureId = 0x04; // bkcmd 2,3 0x04 0xFF 0xFF 0xFF Returned when invalid Picture ID was used

const uint8\_t invalidFontId = 0x05; // bkcmd 2,3 0x05 0xFF 0xFF 0xFF Returned when invalid Font ID was used

const uint8\_t invalidFileOperation = 0x06; // bkcmd 2,3 0x06 0xFF 0xFF 0xFF Returned when File operation fails

const uint8\_t invalidCrc = 0x09; // bkcmd 2,3 0x09 0xFF 0xFF 0xFF Returned when Instructions with CRC validation fails their

// CRC check

const uint8\_t invalidBaudRateSetting = 0x11; // bkcmd 2,3 0x11 0xFF 0xFF 0xFF Returned when invalid Baud rate was used

const uint8\_t invalidWaveformIdChan = 0x12; // bkcmd 2,3 0x12 0xFF 0xFF 0xFF Returned when invalid Waveform ID or Channel # was used

const uint8\_t invalidVarNameAttrib = 0x1A; // bkcmd 2,3 0x1A 0xFF 0xFF 0xFF Returned when invalid Variable name or invalid attribute was

// used

const uint8\_t invalidVarOperation = 0x1B; // bkcmd 2,3 0x1B 0xFF 0xFF 0xFF Returned when Operation of Variable is invalid.

// ie: Text assignment t0.txt = abc or t0.txt = 23,

// or Numeric assignment j0.val = ”50? or j0.val = abc

const uint8\_t assignmentFailed = 0x1C; // bkcmd 2,3 0x1C 0xFF 0xFF 0xFF Returned when attribute assignment failed to assign

const uint8\_t EEPROMOperationFailed = 0x1D; // bkcmd 2,3 0x1D 0xFF 0xFF 0xFF Returned when an EEPROM Operation has failed

const uint8\_t invalidQtyParams = 0x1E; // bkcmd 2,3 0x1E 0xFF 0xFF 0xFF Returned when the number of instruction parameters is

// invalid

const uint8\_t ioOperationFailed = 0x1F; // bkcmd 2,3 0x1F 0xFF 0xFF 0xFF Returned when an IO operation has failed

const uint8\_t invalidEscapeChar = 0x20; // bkcmd 2,3 0x20 0xFF 0xFF 0xFF Returned when an unsupported escape uint8\_tacter is used

const uint8\_t variableNameToLong = 0x23; // bkcmd 2,3 0x23 0xFF 0xFF 0xFF Returned when variable name is too long.Max length is

// 29 characters: 14 for page + “.” + 14 for component.

const uint8\_t serialBufferOverflow = 0x24; // always 0x24 0xFF 0xFF 0xFF Returned when a Serial Buffer overflow occurs

// Buffer will continue to receive the current instruction, all

// previous instructions are lost.

const uint8\_t boilerButton = 5;

const uint8\_t hwButton = 6;

class Stream;

class Nextion {

public:

typedef void (\*setNextionBaudCallbackFunc) (uint32\_t); // create function pointer type

typedef void (\*nextionTurnValveOnOffCallbackFunc) (uint32\_t, bool); // create function pointer type

uint32\_t baudRate = 9600;

const uint32\_t resetNextionBaud = baudRate;

uint32\_t recoveryBaudRate = baudRate; // used for recovery when changing baud rate does not work

nextionEventType nextionEvent;

Nextion(Stream\* s); // s is the serial stream to use e.g. Serial1

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* begin(uint32\_t br, setNextionBaudCallbackFunc func = nullptr) - passes the Nextion \*

\* baud rate to the library. This is put into the variable baudRate. No changes to the \*

\* baudRate are made by this Function. Also, if passed, sets the call back function \*

\* so that this library can have control over the Teensy baudrate. \*

\* Turns on automatic control of Teensy baudrate if passed. \*

\*-------------------------------------------------------------------------------------------------\*

\* Usage: \*

\* begin( baudRate ) - autoSetting of Teensy baud rate set off. \*

\* begin( baudRate, setNextionBaud ) - passes the baud rate and function to change \*

\* Teensy baudRate. \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void begin(uint32\_t br, setNextionBaudCallbackFunc func = nullptr);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* setValveCallBack(nextionTurnValveOnOffCallbackFunc func) - passes the Nextion the \*

\* call back function tu turn a valve on or off \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void setValveCallBack(nextionTurnValveOnOffCallbackFunc func);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Check if char(s) returned from Nextion. If not do something else and come back \*

\* later to check again. \*

\*-------------------------------------------------------------------------------------------------\*

\* If there is a reply from Nextion then the Reply Char is received and the required \*

\* number of following char/bytes dependant upon the value of the Id. \*

\* The Id char is placed in nextionEvent.id. \*

\* The remainig chars are placed in nextionEvent.reply8 ready to be decoded. \*

\* true is returned if there is an Id char and the required number of chars \*

\* are returned. Otherwise false is returned. Timeout is set to 1 Second \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool getReply();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* setLedState - Sets the state of the leds in top, middle or bottom Row. \*

\* which = led (0..7) and state is on (1), off (0) or flashing (2). \*

\*-------------------------------------------------------------------------------------------------\*

\* Just sets the state in variable holding leds row state. There is no change \*

\* to the leds display until setNextionLeds( row ) is used. \*

\*-------------------------------------------------------------------------------------------------\*

\* Usage: setLedState( mid, 4, flashing ); \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void setLedState(topMidBottmType whichLed, uint8\_t which/\*0..7\*/, onOffFlashingType state);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* setNextionLeds actually sends command to Nextion to change the state of \*

\* which leds ( top, middle or bottom row ). \*

\*-------------------------------------------------------------------------------------------------\*

\* Usage: setNextionLeds( top ); \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void setNextionLeds(topMidBottmType which);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* clearLeds sets the leds state variable to all (top, middle and bottom) off. \*

\* Uses setNextionLeds to send command to update all rows on Nextion. \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void clearLeds();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* printTextToNextion - Sends Text to Nextion to be placed in variable \*

\* page0.msg.txt. If transmit is set to true the text is terminated with a " \*

\* character and m0 is clicked to cause the screen on page1 to be updated using \*

\* the finishNextionTextTransmittion() command (see below). \*

\*-------------------------------------------------------------------------------------------------\*

\* Usage: printTextToNextion( "This is a load of text for page1", true ); \*

\*-------------------------------------------------------------------------------------------------\*

\* A string representing the Nextion time in the format " HH:MM:SS " is inserted after \*

\* the first character. This is carried out by the Nextion display. \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void printTextToNextion(const char\* p, bool transmit);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* printNumericText - Sends number to Nextion. This command MUST have been preceded \*

\* by the printTextToNextion command shown above. If transmit is set to true the text \*

\* is terminated with a "character and m0 is clicked to cause the screen on page1 to \*

\* be updated using the finishNextionTextTransmittion() command (see below). \*

\*-------------------------------------------------------------------------------------------------\*

\* Usage: printNumericText( n, true ); // where n is a uint32\_t \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void printNumericText(uint32\_t num, bool transmit);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* finishNextionTextTransmittion() - Terminate the text transmitted to Nextion with a \*

\* " character and terminate the command correctly. Also issue the click m0 command \*

\* to cause the screen on page1 to be updated. \*

\*-------------------------------------------------------------------------------------------------\*

\* Usage: finishNextionTextTransmittion() \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void finishNextionTextTransmittion();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* I like to keep a monitor of what has happened in the system. This display is on \*

\* page1 of the Nextion display. I use the first character position to indicate the \*

\* type of message/source of message. e.g. C for command, E for error message. After \*

\* this character I inser the Time in " HH:MM:SS " format. This is done by the \*

\* Nextion Display. \*

\*-------------------------------------------------------------------------------------------------\*

\* printCommandOrErrorTextMessage - sends the commandOrError charater followed by the \*

\* textMessage to the Nextion using the printTextToNextion command above. \*

\* If transmit is set to true the text is terminated with a "character and m0 is \*

\* clicked to cause the screen on page1 to be updated using the \*

\* finishNextionTextTransmittion() command (see above). \*

\*-------------------------------------------------------------------------------------------------\*

\* Usage: finishNextionTextTransmittion("E", "Error setting valve state", true ) \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void printCommandOrErrorTextMessage(const char\* commandOrError, const char\* textMessage, bool transmit);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* clearBuffer() - Clears the Teensy (Nextion) serial input. \*

\* Use where things have perhaps gone wrong and you need to clear out erroneous \*

\* replies. \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void clearBuffer();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* commsOk() - Checks that valid communications exist with the Nextion Display. \*

\* It sends the command "sendme\xFF\xFF\xFF" and looks for a reply. It does not look \*

\* for the page number for a reply, because comms may have been lost due to using \*

\* the wrong baud rate, in which case a reply might be 0x23FFFFFF - variable name \*

\* too long or some other error reply. Instead it looks for any valid reply. \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool commsOk();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* reset(baudRate) - Resets the Nextion Display and sets the baud rate to "baudRate" \*

\*-------------------------------------------------------------------------------------------------\*

\* Sends a reset command to the Nextion. Sets the Teensy baud rate to 9600 if that \*

\* baud rate NOT already in use. ( upon reset the Nextion defaults to this baud rate ) \*

\* and waits for a valid reply. The Teensy baud rate is set using the callBack \*

\* function registered using the display.begin function. \*

\* When a valid reply has been seen the Nextion AND Teensy have the buadRate changed \*

\* to the baud rate passed in the function call. \*

\* The function returns true if valid comms with the Nextion can be established. \*

\*-------------------------------------------------------------------------------------------------\*

\* Usage: \*

\* reset() - If no baud rate is passed then the baudRate defaults to the reset 9600 \*

\* reset(115200) - Will do a reset and set the baudRate to 115200. \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool reset(uint32\_t br = 0);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* recoverNextionComms() - attempts to recover Nextion Comms once they have been lost \*

\*-------------------------------------------------------------------------------------------------\*

\* First sets the Teensy baud rate to the recoverBaudRate (see setNextionBaudRate \*

\* below). Uses the commsOK function to determine that comms have been re-established. \*

\* If that does not work then all the baud rates that the Nextion might use are cycled \*

\* through until a valid baud rate can be found. \*

\* Returns the value of the baud rate found. \*

\* If NO valid baud rate can be found then returns 0. \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uint32\_t recoverNextionComms();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* respondToReply() - returns true if something needs responding to. \*

\*-------------------------------------------------------------------------------------------------\*

\* This is where you need to put your code. Use getReply() to get any info from the \*

\* Nextion (see above) and this function to decode the reply and respond to it. \*

\* It returns true if further response is needed. \*

\*-------------------------------------------------------------------------------------------------\*

\* I like to have requests from the Nextion Display embedded into numbers. Within this \*

\* code I want to turn valves on or off. The number returned by the Nextion contains \*

\* the valve to be moved and whether it should be opened or closed (0 or 1) \*

\* If you have handled the Nextion response fully then set needsResponse to false. \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool respondToReply();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* printAnyReturnCharacters(uint32\_t nextionTime, uint8\_t id). \*

\* This function is intended to be used in debugging your code. It prints out to the \*

\* SerialUsb the value "nextionTime" and "Id", both values that might be useful in \*

\* tracking down where your error occurred, followed by any values that are in the \*

\* Serial input stream from the Nextion. \*

\* It might be that you have used "respondToReply", with your code in it, but still \*

\* there is something being returned that needs to be responded to. Use this function \*

\* to see what unexpected data is being sent from the Nextion Display. \*

\* ALL data is output in HEX. \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void printAnyReturnCharacters(uint32\_t nextionTime, uint8\_t id);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* turnNextionButton(uint8\_t which, bool on) \*

\* I have Nextion buttons named Sw0..Sw6. I use this function to set the relevant \*

\* button on (1) or off (0) \*

\* I have ghosted this function with the phrase "turnNextionValve" since some of the \*

\* buttons are controlling valves and it makes more sense in the code to refer to \*

\* them as valves. \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define turnNextionValve turnNextionButton

void turnNextionButton(uint8\_t which, bool on);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* setHotWaterOnForMins(uint8\_t howLong) \*

\*-------------------------------------------------------------------------------------------------\*

\* This is somewhat clever. Teensy sets the hot water on and sends a command to the \*

\* Nextion to turn off the hot water in "howLong" minutes. \*

\* When the Nextion receives this command (via a numeric value in a Number Variable) \*

\* it turns the display for the valve open "on" and when the timeout occurs it sends \*

\* a command to the Teensy to turn off the hotwater. This is done via the callback \*

\* setup via the setValveCallBack(nextionTurnValveOnOffCallbackFunc func)function. \*

\* Thus some timing control is offloaded to the Nextion. \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void setHotWaterOnForMins(uint8\_t howLong);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* setTime(uint32\_t time) - Sets the time on the Nextion. \*

\*-------------------------------------------------------------------------------------------------\*

\* The time is sent as HEX HHMMSS in the variable "page0.SetTime.val=time0xFF0xFF0xFF" \*

\* When the Nextion sees that SetTime.val is not zero it sets the Nextion time. The \*

\* SetTime.val variable is then set to 0. \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void setTime(uint32\_t time);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* setNextionBaudRate(uint32\_t br) - Sets the baud rate on Nextion and Teensy. \*

\*-------------------------------------------------------------------------------------------------\*

\* This routine saves the current baud rate in a variable recoveryBaudRate so that \*

\* this recoveryBaudRate can be tried first by the recoverNextionComms() function \*

\* thus saving sometime in the recovery, in order for this function to work correctly \*

\* it requires that the setNextionBaudCallbackFunc was passed to the Library with the \*

\* Nextion.display.begin function. If not it will be the responsibility of the calling \*

\* program to set the Teensy BaudRate accordingly. \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void setNextionBaudRate(uint32\_t br);

elapsedMillis nextionTime; // Just used for internal counting purposes.

private:

// variable for the serial stream

Stream\* \_s;

uint16\_t nextionLeds[3] = { 0,0,0 };

};

#endif