Code by Robert E Bridges
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 600mS intervals, slow I know but fast enough for my current needs.
When, for example this timer notices that the numeric variable "SetTime" 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 SetTime > 0  NOTE: Variables declared in Nextion Programs.s
//================================
if(SetTime!=0)
{
  xx=SetTime
  xx=xx>>16
  rtc3=xx                          // Set the hour
  xx=SetTime
  xx=xx&0xFF00
  xx=xx>>8                         // Set the minutes
  rtc4=xx
  xx=SetTime&0xFF
  rtc5=xx                          // Set the seconds
  SetTime=0
}
```

Revision	Date	Author	Description
1.0	16/04/2022	Robert E Bridges	- Initial release
1.1	17/04/2022	Robert E Bridges	- Added printMoreTextToNextion and improved explanation.
1.15	18/04/2022	Robert E Bridges	- Changed to reflect Nextion use of Global Variables in Program.s
1.20	22/04/2022	Robert E Bridges	- Added nextionError. Set when Nextion returns an error or an invalid number of characters returned - Added errorCode. Set to the Nextion returned error or thr error code for invalid num chars. errorCode is cleared if getReply() is called and there are chars from Nextion. - All the following added: preserveTopTextLine writeToTopTextLine

```
                                                releaseTopTextLine
                                                clearTextScreen
                                                clearTopTextLine
                                                setBackLight
                                                getNumVarValue
                                                setNumVarValue
1.25            01/05/2022      Robert E Bridges    - Completed respondToReply. Now handles the return of Text from the Nextion.
                                                    - Added setTextBuffer. Adds a text buffer where text data is placed from Nextion.
1.30            08/05/2022      Robert E Bridges    - Added askSerialBufferClear      use   THIS
                                                      and isSerialBufferClear      before THIS
1.35            09/05/2022      Robert E Bridges    - Added setBkcmdLevel and handling of successful command completion when bkcmd = 1 or 3.
                                                      see setBkcmdLevel for explanation.
1.40            10/05/2022      Robert E Bridges    - Added lastComdCompletedOk as a complement to setBkcmdLevel above.
                                                    - Added timeout to getReply
*/
```

```
/*****************************************************************************************************
*              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 accommodate        *
*              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 rep7IntType {
       union {
              uint8_t        ans[4];
              uint16_t       num[2];
              int32_t        number32bitInt;
       };
       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;
		rep7IntType		reply7int;
		rep8Type		reply8;
		resetReplyType		resetReply;      //-- The largest Type
		uint8_t			data[sizeof(resetReplyType)]; // Just so that data can be analysed 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 explanation 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
                                                    //                         data: (122, 30) Pressed              in sleep mode
const uint8_t  touchCoordinateSleep         = 0X68; //     8             0x68 0x00 0x7A 0x00 0x1E 0x01 0xFF 0xFF 0xFF Returned when sendxy = 1 and
                                                    //                         data: (122, 30) Pressed (0 for NOT pressed)  exiting sleep
                                                    //
const uint8_t  stringDataEnclosed           = 0x70; //     0             means variable amount
                                                    //                   0x70 0x61 0x62 0x31 0x32 0x33 0xFF 0xFF 0xFF Returned when using get command
                                                    //                         data: ab123                          for string.
                                                    //                                                              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
                                                    //                    data: 67305985                        return a number
                                                    //                                                          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
                                                    //                           (see 1.16)                  receiving the specified quantity of data
```

```
/*****************************************************************************************************
*                                                                                                    *
*              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.
/*
* Error code generated by this library when incorrect number of characters returned by Nextion
*/
const uint8_t invalidNumCharsReturned   = 0x3F;
```

```
enum bkcmdStateType {
        noReturn,//    = 0,
        onSuccess,     // = 1,
        onFailure,     // = 2 Default
        always         // = 3
};

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
                bool           nextionError     = false;
                bool           comdExecOk       = false;                    // only used for bkcmd = 1 or 3
                bool           stringWaiting    = false;
                uint8_t        errorCode        = instructionSuccess;
                bkcmdStateType bkcmd            = onFailure;

                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);

/*****************************************************************************************************
*              sendCommand(const char* command); - Sends command to Nextion.                            *
*              sendCommand(const char* command, uint32_t num); - Sends command & num to Nextion.         *
*              sendCommand(const char* command, uint32_t txt, encloseText); - Sends command & txt        *
*-----------------------------------------------------------------------------------------------------*
*              In the 3rd form above, if encloseTxt is true then txt is enclosed between                 *
*              quotation marks ".                                                                        *
*              So sendCommand( "page0.CommentBox.txt=","Hello There",true); results in                   *
*              page0.CommentBox.txt="Hello There"\xFF\xFF\xFF being sent to the Nextion.                  *
*-----------------------------------------------------------------------------------------------------*
*              Sends the command to Nextion. If bkcmd level has been set to 1 or 3 the code is           *
*              setup to look for a response from the Nextion.                                            *
*              if bkcmd set to 1 or 3, use the command lastComdCompletedOk(uint32_t timeout)             *
*              below after a command or before the next command to determine that the (last)             *
*              command completed ok.                                                                     *
*****************************************************************************************************/
              void sendCommand(const char* command);
              void sendCommand(const char* command, uint32_t num);
              void sendCommand(const char* command, const char* txt, bool encloseText);

/*****************************************************************************************************
*              setBkCmdLevel(bkcmdStateType level) - Sets Nextion bkcmd value                            *
*-----------------------------------------------------------------------------------------------------*
*              The default value is onFailure (2)                                                        *
*              When set to 1 or 3, use the command bool lastComdCompletedOk(uint32_t timeout)            *
*              below after a command or before the next command to determine that the (last)             *
*              command completed ok.                                                                     *
*              level is ONLY allowed to be 1 or 3 if compiled with #define bkcmd1or3allowed in           *
*              Nextiopn.cpp.                                                                             *
*****************************************************************************************************/
              void setBkCmdLevel(bkcmdStateType level);
```

```
/**************************************************************************************************
*              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 some time 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);

/**************************************************************************************************
*              lastComdCompletedOk(uint32_t timeout) - ret true/false if last comd completed ok    *
*---------------------------------------------------------------------------------------------------*
*              This command is to be used if bkcmd level is set to 1 or 3 and ONLY where a         *
*              command is used to set a state on the Nextion.                                      *
*              Where a request for information      is sent to nextion, as in "get varName", the   *
*              returned value is the handshake.                                                    *
*              If other values are used (0 or 2) it is transparent and will return true.           *
*              This is not an indication that the command completed ok as handshaking is off.      *
**************************************************************************************************/
              bool lastComdCompletedOk(uint32_t timeout);


/**************************************************************************************************
*              Set the Text Area to be used for the Return of Text data from Nextion               *
*              If text is sent from the Nextion (following the 0x70 identifier) it will be         *
*              sent to SerialUsb if this function has not been used to specify a variable          *
*              to hold the text data. The parameter must be the size of the textBuffer             *
*              variable. If more text is returned than there is space for in textBuffer            *
*              it will be sent to the SerialUsb.                                                   *
*---------------------------------------------------------------------------------------------------*
*             Usage:   setTextBuffer( textBuffer, sizeof( textBuffer ));                           *
**************************************************************************************************/
              void setTextBuffer(const char* textBuffer, uint8_t textBufferSize);

/**************************************************************************************************
*              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.         *
*              Sets bkcmd to onFailure (Default)                                                      *
*-----------------------------------------------------------------------------------------------------*
*              Usage:                                                                                 *
*                 reset() - If no baud rate is passed then the baudRate defaults to the reset 9600    *
*                 reset(1) - Sets the Baud Rate to that in use at the entry to the Reset function.    *
*                 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();

/*****************************************************************************************************
*              Check if char(s) returned from Nextion. If not do something else and come back         *
*              later to check again. Wait for timeout. Default is 0..don't wait.                      *
*-----------------------------------------------------------------------------------------------------*
*              If there is a reply from Nextion then the Reply Char is received and the required      *
*              number of following char/bytes dependent upon the value of the Id.                     *
*              The Id char is placed in nextionEvent.id.                                              *
*              The remaining 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.                                            *
*              If the first char is received within timeout a further timeout of 1 second             *
*              is allowed for remaining characters.                                                   *
*              This proc does NOT get any strings returned from Nextion,Use respondToReply()          *
*              for that.                                                                              *
*****************************************************************************************************/
              bool getReply(uint32_t timeout = 0);
```

```
/*****************************************************************************************************
*              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);

/*****************************************************************************************************
*              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        *
*              recoveryBaudRate can be tried first by the recoverNextionComms() function,             *
*              thus saving some time 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);

/*****************************************************************************************************
*              setBackLight(uint32_t backLight) - Sets the display BackLight(0..100).                 *
*-----------------------------------------------------------------------------------------------------*
*              Any value greater than 100 will default to 100.                                        *
*              0 is off 100 is MAX brightness.                                                        *
*****************************************************************************************************/
              void setBackLight(uint32_t backLight);
```

```
/**************************************************************************************************
*               getNumVarValue(const char* varName) - Gets the value of Nextion Variable.         *
*---------------------------------------------------------------------------------------------------*
*               Waits for up to 100ms for a reply. If no reply returns 0xFFFF.                    *
*               In reality this command should only be sent when the Nextion Serial buffer is     *
*               otherwise, any reply may be from previously stacked up Nextion commands and       *
*               therefore be erroneous.                                                           *
*               The varName MUST exist.                                                           *
**************************************************************************************************/
               int32_t getNumVarValue(const char* varName);

/**************************************************************************************************
*               getStringVarValue(const char* varName) - Gets the text from Nextion Variable.     *
*---------------------------------------------------------------------------------------------------*
*               Waits for up to 100ms for a reply. If no reply returns 0xFFFF.                    *
*               In reality this command should only be sent when the Nextion Serial buffer is empty *
*               otherwise, any reply may be from previously stacked up Nextion commands and therefore *
*               be erroneous.                                                                     *
*               The varName MUST exist.                                                           *
*               The result is placed in the string setup with the setTextBuffer function.         *
*               If no screen has been setup it will simply be echoed to the screen (Serial).      *
*               Returns true if string returned successfully. stringWaiting is set to true.       *
**************************************************************************************************/
               bool getStringVarValue(const char* varName);

/**************************************************************************************************
*               setNumVarValue(const char* varName, int32_t var ) - Sets Nextion Variable to var. *
*---------------------------------------------------------------------------------------------------*
*               The varName MUST exist.                                                           *
**************************************************************************************************/
               bool setNumVarValue(const char* varName, int32_t var);

/**************************************************************************************************
*               askSerialBufferClear() - Ask Nextion if Serial Buffer Clear (Empty)               *
*---------------------------------------------------------------------------------------------------*
*               Sends "get clrBufr" to Nextion. Nextion will reply with 0xFDFD when it gets to    *
*               this request in the SerialBuffer, indicating it has executed this last command   *
*               in the Serial Buffer. If other commands are sent after this one the Serial        *
*               Buffer WILL NOT BE CLEAR.                                                         *
*               Use the command isSerialBufferClear(), below to confirm Serial Buffer Clear.      *
*               Requires this line "int clrBufr=65021" in Nextion Program.s                       *
**************************************************************************************************/
               void askSerialBufferClear();

/**************************************************************************************************
*               isSerialBufferClear() - Query answer from askSerialBufferClear() above            *
*---------------------------------------------------------------------------------------------------*
*               NOTE that if other commands are stacked up which will give a reply from Nextion,  *
*               then they will be handled by the calls to getReply and respondToReply used by     *
*               this function. They may return a reply, but if it is NOT a Numeric reply with     *
*               0xFDFD they will NOT return true.                                                 *
**************************************************************************************************/
               bool isSerialBufferClear();
```

```
/*****************************************************************************************************
*              bool askSerialBufferClear(uint32_t timeout) - As above but waits for a reply             *
*------------------------------------------------------------------------------------------------------*
*              Combines askSerialBufferClear() and isSerialBufferClear() with a timeout to              *
*              determine if the Nextion input Serial Buffer is Clear.                                   *
*****************************************************************************************************/
              bool askSerialBufferClear(uint32_t timeout);

/*****************************************************************************************************
*              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 hot water. 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 "SetTime=HHMMSS0xFF0xFF0xFF"              *
*              When the Nextion sees that SetTime is not zero it sets the Nextion time.                 *
*              The SetTime variable is then set to 0.                                                   *
*------------------------------------------------------------------------------------------------------*
*              Usage:                                                                                   *
*                              uint32_t time = Hours * 0x10000 + Minutes * 0x100 + Seconds              *
*                              display.setTime(time)                                                    *
*****************************************************************************************************/
              void setTime(uint32_t time);

/*****************************************************************************************************
*              turnDebugOn(bool on) - Turn Nextion debug variable on or off                             *
*------------------------------------------------------------------------------------------------------*
*              Usage:                                                                                   *
*                 turnDebugOn( true )  - Turn debug on                                                  *
*                 turnDebugOn( false ) - Turn debug off                                                 *
*****************************************************************************************************/
              bool turnDebugOn(bool on);
```

```
/*****************************************************************************************************
*              turnScreenDimOn(bool on) - Turn Nextion dimAllowed variable on or off                 *
*-----------------------------------------------------------------------------------------------------*
*              Usage:                                                                                 *
*                 turnScreenDimOn( true )  - Turn Dim on                                              *
*                 turnScreenDimOn( false ) - Turn Dim off                                             *
*****************************************************************************************************/
              bool turnScreenDimOn(bool on);

/*****************************************************************************************************
*              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);

/*****************************************************************************************************
*              setValveCallBack(nextionTurnValveOnOffCallbackFunc func) - passes the Nextion the      *
*              call back function    tu turn a valve on or off                                        *
*****************************************************************************************************/
              void setValveCallBack(nextionTurnValveOnOffCallbackFunc func);

/*****************************************************************************************************
*              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 ) set with setLedState function above.          *
*-----------------------------------------------------------------------------------------------------*
*              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();
```

```
/*************************************************************************************************
*              printTimeEmbeddedTextToNextion - 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 m0,1 is clicked to cause the screen on page1 to be updated using           *
*              the finishNextionTextTransmittion() command (see below).                                 *
*              The procedure sends page0.msg.txt=" to the Nextion followed by the text.                  *
*-------------------------------------------------------------------------------------------------*
*       Usage: printTimeEmbeddedTextToNextion( "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 printTimeEmbeddedTextToNextion(const char* p, bool transmit);

/*************************************************************************************************
*              printTextToNextion - Sends Text to Nextion to be placed in variable                      *
*              page1.va0.txt. If transmit is set to true the text is terminated with a "                *
*              character and m0,0 is clicked to cause the screen on page1 to be updated using           *
*              the finishNextionTextTransmittion() command (see below).                                 *
*              The procedure sends page1.va0.txt=" to the Nextion followed by the text.                  *
*-------------------------------------------------------------------------------------------------*
*             Usage:   printTextToNextion( "This is a load of text for page1", true );                  *
*************************************************************************************************/
              void printTextToNextion(const char* p, bool transmit);


/*************************************************************************************************
*              printMoreTextToNextion - It is the same as the printTextToNextion function except        *
*              that the page0.msg.txt=" is NOT sent.                                                    *
*-------------------------------------------------------------------------------------------------*
*             Usage:   printMoreTextToNextion( "This is a load of text for page1", true );              *
*              NOTE: DO NOT use this without first using printTextToNextion( "text", false );           *
*************************************************************************************************/
              void printMoreTextToNextion(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                           *
*              NOTE: DO NOT use this without first using printTextToNextion( "text", false );           *
*************************************************************************************************/
              void printNumericText(uint32_t num, bool transmit);
```

```
/*****************************************************************************************************
*              finishNextionTextTransmittion() - Terminate the text transmitted to Nextion with a       *
*              " character and terminate the command correctly. Also issues the relevant                *
*              click m0 command dependant upon which printText command was used to cause the            *
*              screen on page1 to be updated.                                                           *
*              ( Uses "click m0,1" or "click m0,0" as appropriate )                                     *
*-----------------------------------------------------------------------------------------------------*
*             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).                                     *
*****************************************************************************************************/
              void printCommandOrErrorTextMessage(const char* commandOrError, const char* textMessage, bool transmit);


/*****************************************************************************************************
*              preserveTopTextLine() - Top text line writing inhibited.                                 *
*-----------------------------------------------------------------------------------------------------*
*              All general text commands do not use top line if this command actuated.                  *
*****************************************************************************************************/
              void preserveTopTextLine();

/*****************************************************************************************************
*              writeToTopTextLine(const char* textMessage)                                              *
*****************************************************************************************************/
              void writeToTopTextLine(const char* textMessage);

/*****************************************************************************************************
*              releaseTopTextLine() - Allows writing to the Top Text Line                               *
*-----------------------------------------------------------------------------------------------------*
*              All general text commands can use top line again (Default Setting).                      *
*****************************************************************************************************/
              void releaseTopTextLine();

/*****************************************************************************************************
*              clearTextScreen()- Clears the Nextion Text Screen (page1)                                *
*-------------------------------- -----------------------------------------------------------------*
*              If the Top Line is preserved that is not cleared, use clearTopTextLine instead.          *
*****************************************************************************************************/
              void clearTextScreen();
```

```
/*****************************************************************************************************
*              clearTopTextLine() - Clears the Nextion Text Screen Top Text Line                     *
*****************************************************************************************************/
              void clearTopTextLine();


/*****************************************************************************************************
*              setDaylightSavingOn( on) - Turn Nextion daylight saving variable on or off            *
*-----------------------------------------------------------------------------------------------------*
*              Usage:                                                                                 *
*                 setDaylightSavingOn( true )  - Turn on                                              *
*                 setDaylightSavingOn( false ) - Turn off                                             *
*****************************************************************************************************/
              bool setDaylightSavingOn( bool on);

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

};
```