

intLib

Generated by Doxygen 1.8.6

Thu Apr 10 2014 20:58:11



# Contents

<b>1</b>	<b>uIntPLib</b>	<b>1</b>
<b>2</b>	<b>uIntPLib</b>	<b>3</b>
<b>3</b>	<b>Todo List</b>	<b>5</b>
<b>4</b>	<b>Module Documentation</b>	<b>7</b>
4.1	Lcd_h	7
4.1.1	Detailed Description	9
4.1.2	Function Documentation	10
4.1.2.1	arrayToNum	10
4.1.2.2	LCDClear	11
4.1.2.3	LCDDisplayOn	11
4.1.2.4	LCDHome	11
4.1.2.5	LCDInit	12
4.1.2.6	LCDPosition	12
4.1.2.7	LCDPositionNoDelay	12
4.1.2.8	LCDRegisterSpecial	13
4.1.2.9	LCDSend	13
4.1.2.10	LCDSendChar	14
4.1.2.11	LCDSendCmd	14
4.1.2.12	LCDSendHex	14
4.1.2.13	LCDSendNum	15
4.1.2.14	LCDSendNumArray	15
4.1.2.15	LCDSendNumStrict	16
4.1.2.16	LCDSendString	16
4.1.2.17	LCDSendVU	17
4.1.2.18	LCDShift	17
4.1.2.19	numToArray	18
4.1.3	Variable Documentation	19
4.1.3.1	LCD0Status	19
4.1.3.2	LCD_CmdInit_Vector	19

---

4.1.3.3	LCD_InitDelay_Vector . . . . .	19
<b>5</b>	<b>Data Structure Documentation</b>	<b>21</b>
5.1	CommandInstance Struct Reference . . . . .	21
5.1.1	Detailed Description . . . . .	21
5.2	IRInstance Struct Reference . . . . .	21
5.2.1	Detailed Description . . . . .	21
5.3	LCDStatus Struct Reference . . . . .	22
5.3.1	Detailed Description . . . . .	22
5.4	UARTInstance Struct Reference . . . . .	22
5.4.1	Detailed Description . . . . .	22
<b>Index</b>		<b>23</b>

# Chapter 1

## uIntPLib

Universal Integrated Peripheral Library

This is a library made with functions masks to medium level programming. Intended to make code more portable, while maintaining its performance.

Doxygen generated documentation is located at [latex/refman.pdf](#) Complete documentation is under construction.



## Chapter 2

# uIntPLib

Universal Integrated Peripheral Library

This is a library made with functions masks to medium level programming. Intended to make code more portable, while maintaining its performance.

Doxygen generated documentation is located at [latex/refman.pdf](#) Complete documentation is under construction.





## Chapter 3

### Todo List

**Global `LCD0Status`**

create masks for LCD commands.

**Global `LCDSend (uint8_t send)`**

make function to send parallel data.



# Chapter 4

## Module Documentation

### 4.1 Lcd\_h

#### Data Structures

- struct [LCDStatus](#)

#### Macros

- #define **true** 1
- #define **false** 0
- #define **trueDefinedLCD**
- #define [lcd\\_vector\\_index](#) 7  
*Initialization command sequence length.*
- #define **maxLengthOut** 16

#### Functions

- void [LCDInit](#) (void)  
*Initializes the LCD Module.*
- void [LCDSendCmd](#) (uint8\_t cmd)  
*Send a Command to the LCD.*
- void [LCDSendChar](#) (uint8\_t txt)  
*send single character to LCD.*
- \_\_inline void [LCDSend](#) (uint8\_t send)  
*Send data to LCD, no RS control.*
- void [LCDPosition](#) (uint8\_t row, uint8\_t col)  
*Set LCD write position.*
- void [LCDPositionNoDelay](#) (uint8\_t row, uint8\_t col)  
*Set LCD write position, no delay in function.*
- void [LCDSendString](#) (uint8\_t \*string, uint8\_t breakLine)  
*Send string to LCD Writes a string of characteres on display Processes according to the ASCII code 0 - NULL.*
- void [LCDSendNumStrict](#) (int64\_t num, uint8\_t length, uint8\_t isSigned, uint8\_t showZeros)  
*Send decimal number to LCD with a strict length.*
- void [LCDSendNum](#) (int64\_t num, uint8\_t length, uint8\_t isSigned, uint8\_t showZeros)  
*Send decimal number to LCD with a variable length.*
- void [LCDSendNumArray](#) (uint8\_t \*index)

- Sends a number in a array to the LCD.*
  - void `LCDClear` (void)
- Clears the LCD.*
  - void `LCDDisplayOn` (uint8\_t onOff)
- Configures LCD appearence option.*
  - void `LCDSendHex` (uint8\_t \*array)
- Prints a decimal number in hexadecimal format.*
  - void `numToArray` (int32\_t num, uint8\_t \*array, uint8\_t length, uint16\_t base)
- Converts a variable to an array of numbers.*
  - void `LCDRegisterSpecial` (uint8\_t number, uint8\_t \*character)
- Registers special characters in the LCD.*
  - void `LCDShift` (uint8\_t shift)
- Enables the data shift option in LCD.*
  - void `LCDHome` (void)
- Sends the LCD cursor to home position.*
  - void `arrayToNum` (uint8\_t \*array, uint32\_t \*num, uint8\_t base)
- Converts a number array to a variable.*
  - void `LCDSendVU` (uint32\_t num, uint32\_t base)
- Send LCD VU level.*

## Variables

- const char `LCD_CmdInit_Vector` [`lcd_vector_index`] = {0x38, 0x38, 0x38, 0x01, LCD\_DISPLAY\_CONFIG, LCD\_DISPLAY\_INCREMENT, 0x01}
  - const unsigned int `LCD_InitDelay_Vector` [`lcd_vector_index`] = {8000, 200, 200, 16000, 600, 200, 15000}
  - `LCDStatus LCD0Status`
- LCD struct for current position.*

## LCD\_Splashscreen

LCD splashscreen text configuration.

- #define `LCD_splashscreen_row1` `__DATE__`  
*compile date, used as program version*
- #define `LCD_splashscreen_row2` `__TIME__`  
*compile time, used as program version*
- #define `LCD_splashscreen2_row1` `PROJECT_NAME`  
*geneartion of project name in LCD*
- #define `LCD_splashscreen2_row2` ("rnm sys undvdp")  
*creator's watermark*

## LCD\_Function\_Masks

Function masks used in the lib.

- #define `LCDDelay`(x) `SysDelayUs`(x)  
*delay loop mask.*
- #define `LCDPinSet`(x, y) `PinAddrSet`(x, y)  
*pin set function mask.*
- #define `LCDPinClear`(x, y) `PinAddrClear`(x,y)

- pin clear function mask.*
- #define **LCD\_DTA\_Send**(text) ShiftSerialSend(LCD\_DTA\_Port, LCD\_DTA\_Pin, LCD\_CLK\_Port, LCD\_CLK\_Pin, text)
- lcd send function mask.*
- #define **LCD\_RS\_High LCDPinSet**(LCD\_RS\_Port, LCD\_RS\_Pin)
- RS pin set mask.*
- #define **LCD\_RS\_Low LCDPinClear**(LCD\_RS\_Port, LCD\_RS\_Pin)
- RS pin clear mask.*
- #define **LCD\_EN\_High LCDPinSet**(LCD\_EN\_Port, LCD\_EN\_Pin)
- EN pin set mask.*
- #define **LCD\_EN\_Low LCDPinClear**(LCD\_EN\_Port, LCD\_EN\_Pin)
- EN pin clear mask.*

## LCD\_Option\_Flags

LCD options flags.

- #define **LCD\_DISPLAY\_ON** 0x0C
- #define **LCD\_DISPLAY\_OFF** 0x08
- #define **LCD\_CURSOR\_ON** 0x0A
- #define **LCD\_CURSOR\_OFF** 0x08
- #define **LCD\_BLINK\_ON** 0x09
- #define **LCD\_BLINK\_OFF** 0x08
- #define **LCD\_SHIFT** 0x10
- #define **LCD\_SHIFT\_DISPLAY** 0x08
- #define **LCD\_SHIFT\_CURSOR** 0x02
- #define **LCD\_SHIFT\_RIGHT** 0x04
- #define **LCD\_SHIFT\_LEFT** 0x00
- #define **LCD\_SET\_CGRAM** 0x40
- #define **LCD\_INCREMENT** 0x04
- #define **LCD\_INCREMENT\_NO\_SHIFT** 0x00
- #define **LCD\_INCREMENT\_SHIFT** 0x01
- #define **LCD\_INCREMENT\_POSITIVE** 0x02
- #define **LCD\_INCREMENT\_NEGATIVE** 0x00

## LCD\_Options\_Select

Defitions of initial LCD options configuration.

- #define **LCD\_DISPLAY\_CONFIG** (LCD\_DISPLAY\_ON|LCD\_CURSOR\_OFF|LCD\_BLINK\_OFF)
- #define **LCD\_DISPLAY\_INCREMENT** (LCD\_INCREMENT|LCD\_INCREMENT\_NO\_SHIFT)

### 4.1.1 Detailed Description

This file contains the functions to properly (hopefully) drive the LCD peripheral. Current implementation uses a serial shift register to drive the LCD in 8 bit mode.

Further modifications to this library include:

- Parallel data transfer (make function to).

For the complete execution of the library, the following macros have to be created specifically for this library:

- LCD\_SPLASHSCREEN1  
(1 or 0)
- LCD\_SPLASHSCREEN2  
(1 or 0)
- LCD\_SPLASHSCREEN\_CLEAR  
(1 or 0)

The library uses theses global value macros:

- PROJECT\_NAME  
("string\_of\_name\_here")

The library uses these global function macros:

- SysDelayUs(time)  
CPU delay loop, usually.
- PinAddrSet(port, pin)  
Direct address write.
- PinAddrClear(port, pin)  
Direct address write.
- ShiftSerialSend(DTA\_PORT, DTA\_PIN, CLK\_PORT, CLK\_PIN, text)  
Used by [LCD\\_DTA\\_Send\(text\)](#) function, determines how data is transfered.

The following external connection macros have to be set:

- LCD\_RS\_PORT  
(register address)
- LCD\_RS\_PIN  
(bit position)
- LCD\_EN\_PORT  
(register address)
- LCD\_EN\_PIN  
(bit position)
- LCD\_DTA\_PORT  
(register address)
- LCD\_DTA\_PIN  
(bit position)
- LCD\_CLK\_PIN  
(bit position)

## 4.1.2 Function Documentation

### 4.1.2.1 void arrayToNum ( uint8\_t \* array, uint32\_t \* num, uint8\_t base )

**Parameters**

<i>*array</i>	uint8_t source array.
<i>*num</i>	uint32_t target variable.
<i>base</i>	uint8_t base of digits in array.

Definition at line 477 of file [lcd.c](#).

```
00478 {  
00479     while(*array<33)  
00480     {  
00481         *num += *array * base;  
00482         array++;  
00483     }  
00484 }
```

**4.1.2.2 void LCDClear ( void )**

Sends a clear all command to the LCD.

**Returns**

None.

Definition at line 304 of file [lcd.c](#).

```
00305 {  
00306     LCDSendCmd(0x01);  
00307     LCD0Status.row=1;  
00308     LCD0Status.col=1;  
00309     LCDDelay(800);  
00310 }
```

**4.1.2.3 void LCDDisplayOn ( uint8\_t onOff )**

Sends the logical combination of the following flags:

- LCD\_DISPLAY\_ON/OFF
- LCD\_CURSOR\_ON/OFF
- LCD\_BLINK\_ON/OFF

**Returns**

None.

Definition at line 324 of file [lcd.c](#).

```
00325 {  
00326     LCD0Status.display = onOff;  
00327     LCDSendCmd(onOff);  
00328 }
```

**4.1.2.4 void LCDHome ( void )****Returns**

None.

Definition at line 463 of file [lcd.c](#).

```
00464 {  
00465     LCDSendCmd(0x02);  
00466     LCDDelay(1500);  
00467 }
```

#### 4.1.2.5 void LCDInit ( void )

Called once at startup. Takes no parameters.

##### Returns

None.

Definition at line 35 of file [lcd.c](#).

```

00036 {
00037     uint8_t Vector_Scan = 0;
00038     LCDDelay(15000);
00039     for(Vector_Scan=0; Vector_Scan < lcd_vector_index; Vector_Scan++)
00040     {
00041         LCDSendCmd(LCD_CmdInit_Vector[Vector_Scan]);
00042         LCDDelay(LCD_InitDelay_Vector[Vector_Scan]);
00043     }
00044     //splash screen
00045     #if LCD_SPLASHSCREEN1 == 1
00046         LCDPosition(1,1);
00047         LCDSendString(LCD_splashscreen_row1, 0);
00048         LCDPosition(2,1);
00049         LCDSendString(LCD_splashscreen_row2, 0);
00050         LCDDelay(2*1000*1000);
00051     #endif
00052     #if LCD_SPLASHSCREEN2 == 1
00053         LCDPosition(1,1);
00054         LCDSendString(LCD_splashscreen2_row1, false);
00055         LCDPosition(2,1);
00056         LCDSendString(LCD_splashscreen2_row2, false);
00057         LCDDelay(2*1000*1000);
00058     #endif
00059     #if LCD_SPLASHSCREEN_CLEAR == 1
00060         LCDClear();
00061     #endif
00062 }
```

#### 4.1.2.6 void LCDPosition ( uint8\_t row, uint8\_t col )

##### Parameters

<i>row</i>	uint8_t row.
<i>col</i>	uint8_t column.

##### Returns

None.

Definition at line 116 of file [lcd.c](#).

```

00117 {
00118     LCD0Status.row = row;
00119     LCD0Status.col = col;
00120     col--;
00121     if(row==1)
00122         row = 0x80;
00123     if(row==2)
00124         row = 0xC0;
00125     LCDSendCmd(row+col);
00126     LCDDelay(20);
00127 }
```

#### 4.1.2.7 void LCDPositionNoDelay ( uint8\_t row, uint8\_t col )

Sends the LCD position command but does not implement a delay after.



## Parameters

<i>row</i>	uint8_t row.
<i>col</i>	uint8_t column.

## Returns

None.

Definition at line 140 of file [lcd.c](#).

```
00141 {  
00142     LCD0Status.row = row;  
00143     LCD0Status.col = col;  
00144     col--;  
00145     if(row==1)  
00146         row = 0x80;  
00147     if(row==2)  
00148         row = 0xC0;  
00149     LCDSendCmd(row+col);  
00150 }
```

4.1.2.8 void LCDRegisterSpecial ( uint8\_t *number*, uint8\_t\* *character* )

## Parameters

<i>number</i>	uint8_t number, from 1 to 8, of special character to be transferred.
* <i>character</i>	uint8_t array containing the bits of the character to be set.

## Returns

None.

Definition at line 421 of file [lcd.c](#).

```
00422 {  
00423     uint8_t scan=0, data=0;  
00424     LCDSendCmd(0x40+(number<<3));  
00425     do  
00426     {  
00427         data = *(character+scan);  
00428         LCDDelay(640);  
00429         LCDSendChar(data&0x1F);  
00430         scan++;  
00431     }  
00432     while(scan<8);  
00433     LCDDelay(320);  
00434 }
```

4.1.2.9 void LCDSend ( uint8\_t *send* )

## Parameters

<i>send</i>	uint8_t data to be sent.
-------------	--------------------------

## Returns

None.

**Todo** make function to send parallel data.

Definition at line 97 of file [lcd.c](#).

```

00098 {
00099     LCD_EN_Low;
00100     LCD_DTA_Send(send);
00101     LCDDelay(4);
00102     LCD_EN_High;
00103     LCDDelay(4);
00104     LCD_EN_Low;
00106 }

```

#### 4.1.2.10 void LCDSendChar ( uint8\_t txt )

##### Parameters

<i>txt</i>	uint8_t type data to be sent, 8 bits.
------------	---------------------------------------

##### Returns

None.

Definition at line 82 of file [lcd.c](#).

```

00083 {
00084     LCD_RS_High;
00085     LCDSend(txt);
00086     LCD0Status.col ++;
00087     LCD_RS_Low;
00088 }

```

#### 4.1.2.11 void LCDSendCmd ( uint8\_t cmd )

##### Returns

None.

Definition at line 68 of file [lcd.c](#).

```

00069 {
00070     LCD_RS_Low;
00071     LCDSend(cmd);
00072 }

```

#### 4.1.2.12 void LCDSendHex ( uint8\_t \* array )

##### Parameters

<i>*array</i>	uint8_t number array to be written.
---------------	-------------------------------------

##### Returns

None.

Definition at line 338 of file [lcd.c](#).

```

00339 {
00340     uint8_t offset, temp;
00341     LCDSendChar('0');
00342     LCDSendChar('x');
00343     array += 2;
00344     while(*array<=32)
00345     {
00346         temp = *array;
00347         if(temp>9)
00348         {

```

```

00349         temp -= 10;
00350         offset = 'A';
00351     }
00352     else
00353         offset = '0';
00354     LCDSendChar(temp+offset);
00355     array++;
00356 }
00357 }

```

#### 4.1.2.13 void LCDSendNum ( int64\_t num, uint8\_t length, uint8\_t isSigned, uint8\_t showZeros )

Writes a decimal number with a variable length in the LCD.

##### Parameters

<i>num</i>	int64_t number to be written.
<i>length</i>	uint8_t length, in decimal digits, of the number.
<i>isSigned</i>	uint8_t flag to determine if the number is to be treated as a negative number.
<i>showZeros</i>	uint8_t flag to determine if leading zeros will be shown.

##### Returns

None.

Definition at line 246 of file [lcd.c](#).

```

00247 {
00248     uint8_t index =0;
00249     uint8_t out = ' ';
00250     uint64_t multiple = 1;
00251     limitCeilValue(length,10);
00252     if(num<0 && isSigned==true)
00253     {
00254         out = '-';
00255         num *= -1;
00256     }
00257     LCDSendChar(out);
00258     index = length;
00259     while(length>1)
00260     {
00261         multiple *= 10;
00262         length--;
00263     }
00264     while(index >= 1)
00265     {
00266         out = (uint32_t) (num/multiple);
00267         num -= out*(multiple);
00268         if(out!=0)
00269             showZeros = true;
00270         if(out==0 && showZeros==false)
00271             out -= 16;
00272         LCDSendChar(out+48);
00273         multiple /= 10;
00274         index--;
00275     }
00276 }

```

#### 4.1.2.14 void LCDSendNumArray ( uint8\_t \* index )

Sends a number arranged in an array to the LCD. Each cell corresponds to a digit in the LCD.

##### Parameters

<i>*index</i>	uint8_t base address of the array to be written.
---------------	--

**Returns**

None.

Definition at line 288 of file [lcd.c](#).

```
00289 {
00290     while(*index<33)
00291     {
00292         LCDSendChar(*index+'0');
00293         index++;
00294     }
00295 }
```

**4.1.2.15 void LCDSendNumStrict ( int64\_t num, uint8\_t length, uint8\_t isSigned, uint8\_t showZeros )**

Writes a decimal number with a strict length in the LCD.

**Parameters**

<i>num</i>	int64_t number to be written.
<i>length</i>	uint8_t length, in decimal digits, of the number.
<i>isSigned</i>	uint8_t flag to determine if the number is to be treted as a negative number.
<i>showZeros</i>	uint8_t flag to determine if leading zeros will be shown.

**Returns**

None.

Definition at line 200 of file [lcd.c](#).

```
00201 {
00202     uint8_t index =0;
00203     uint8_t out;
00204     uint64_t multiple = 1;
00205     limitCeilValue(length,10);
00206     if(num<0 && isSigned==true)
00207     {
00208         LCDSendChar('-');
00209         num *= -1;
00210         length--;
00211     }
00212     index = length;
00213     while(length>1)
00214     {
00215         multiple *= 10;
00216         length--;
00217     }
00218     while(index >= 1)
00219     {
00220         out = (uint32_t) (num/multiple);
00221         num -= out*(multiple);
00222         if(out!=0)
00223             showZeros = true;
00224         if(out==0 && showZeros==false)
00225             out -= 16;
00226         LCDSendChar(out+48);
00227         multiple /= 10;
00228         index--;
00229     }
00230 }
```

**4.1.2.16 void LCDSendString ( uint8\_t \* string, uint8\_t breakLine )**

## Parameters

<i>*string</i>	uint8_t string to be sent.
<i>breakLine</i>	uint8_t break line at the end of LCD length.

## Returns

None.

Definition at line 164 of file [lcd.c](#).

```

00165 {
00166     while(*string)
00167     {
00168         LCDSendChar(*string);
00169         string++;
00170         if(LCD0Status.col==LCD_col_num && breakLine==true)
00171         {
00172             if(LCD0Status.row<=LCD_row_num)
00173                 LCDPosition(LCD0Status.row+1, 1);
00174             else
00175                 LCDPosition(0, 1);
00176         }
00177     }
00178 }
```

## 4.1.2.17 void LCDSendVU ( uint32\_t num, uint32\_t base )

This function uses special characters filled in horizontal increasing steps to make a Visual Units Display from the LCD.

## Parameters

<i>num</i>	uint32_t number to be written.
<i>base</i>	uint32_t max number value.

## Returns

None.

Definition at line 498 of file [lcd.c](#).

```

00499 {
00500     uint8_t index, pass=1;
00501     num = (unsigned int) num*(LCD_col_num*LCD_char_width)/base;
00502     while(num>0)
00503     {
00504         index = LCD_char_width;
00505         while(num<LCD_char_width)
00506         {
00507             index--;
00508             num++;
00509         }
00510         LCDSendChar(index);
00511         num -= LCD_char_width;
00512         pass++;
00513     }
00514     while(pass<=LCD_col_num)
00515     {
00516         pass++;
00517         LCDSendChar(0);
00518     }
00519 }
```

## 4.1.2.18 void LCDShift ( uint8\_t shift )

Sets the configuration according to the following flags:

- LCD\_SHIFT
- LCD\_SHIFT\_DISPLAY
- LCD\_SHIFT\_CURSOR
- LCD\_SHIFT\_RIGHT
- LCD\_SHIFT\_LEFT

#### Parameters

<i>shift</i>	uint8_t option flag to be set.
--------------	--------------------------------

#### Returns

None.

Definition at line 452 of file [lcd.c](#).

```
00453 {
00454     LCDSendCmd(shift|LCD_SHIFT);
00455 }
```

#### 4.1.2.19 void numToArray ( int32\_t num, uint8\_t \* array, uint8\_t length, uint16\_t base )

The termination of array is done by the number 33. The highest selectable number base is 32.

#### Parameters

<i>num</i>	int32_t number to be converted.
<i>*array</i>	uint8_t destination array.
<i>length</i>	uint8_t number length, in decimal digits.
<i>base</i>	uint8_t base of output array.

#### Returns

None.

Definition at line 373 of file [lcd.c](#).

```
00374 {
00375     uint16_t index =1;
00376     uint8_t out;
00377     uint64_t multiple = 1;
00378
00379     limitCeilValue(length, (unsigned char) 1<<64/base);
00380     limitCeilValue(length, maxLengthOut);
00381
00382
00383     //create multiple number
00384     while(index<length)
00385     {
00386         multiple *= base;
00387         index++;
00388     }
00389     //sort multiples
00390     while(index >= 1)
00391     {
00392         //determines the multiple
00393         out = (uint8_t) (num/multiple);
00394         //takes out multiple
00395         num -= out*(multiple);
00396
00397         //escreve no vetor, desloca indice
00398         *array = out;
00399         array++;
00400         multiple /= base;
00401         //change multiple position
00402         index--;
00403     }
00404     *array = 33;
00405 }
```

### 4.1.3 Variable Documentation

#### 4.1.3.1 LCDStatus LCD0Status

**Todo** create masks for LCD commands.

4.1.3.2 `const char LCD_CmdInit_Vector[lcd_vector_index] = {0x38, 0x38, 0x38, 0x01, LCD_DISPLAY_CONFIG, LCD_DISPLAY_INCREMENT, 0x01}`

Initialization Commands Sequence:

Definition at line 20 of file [lcd.c](#).

4.1.3.3 `const unsigned int LCD_InitDelay_Vector[lcd_vector_index] = {8000, 200, 200, 16000, 600, 200, 15000}`

LCD Init command delay vector, in uS.

Definition at line 25 of file [lcd.c](#).





## Chapter 5

# Data Structure Documentation

### 5.1 CommandInstance Struct Reference

#### Data Fields

- `uint8_t charIn`
- `uint8_t cmdBuffer` [MAX\_BUFFER\_SIZE]
- `uint16_t charOut` [MAX\_BUFFER\_SIZE]
- `uint8_t charOutPtr`

#### 5.1.1 Detailed Description

Definition at line 15 of file [cmd\\_sort.h](#).

The documentation for this struct was generated from the following file:

- `my_lib/cmd_sort.h`

### 5.2 IRInstance Struct Reference

#### Data Fields

- `uint16_t Mode`
- `uint8_t CarrierFrequency`
- `uint16_t CarrierPeriod`
- `uint32_t TxPin`
- `uint32_t TxPort`
- `uint32_t RxPin`
- `uint32_t RxPort`
- `uint16_t ReceiveAddress`
- `uint16_t ReceiveBuffer`
- `uint16_t Pulses`
- `uint8_t LastData`

#### 5.2.1 Detailed Description

Definition at line 83 of file [ir.h](#).

The documentation for this struct was generated from the following file:

- `my_lib/ir.h`

## 5.3 LCDStatus Struct Reference

### Data Fields

- `uint8_t row`
- `uint8_t col`
- `uint8_t display`
- `uint8_t shift`
- `uint8_t cgramAdress`
- `uint8_t specialChar [8]`

### 5.3.1 Detailed Description

Definition at line 135 of file `lcd.h`.

The documentation for this struct was generated from the following file:

- `my_lib/lcd.h`

## 5.4 UARTInstance Struct Reference

### Data Fields

- `uint8_t RxBuffer [UART_BUFFER_SIZE]`
- `uint8_t RxBufferPtr`
- `uint8_t TxBuffer [UART_BUFFER_SIZE]`
- `uint8_t TxBufferPtr`
- `uint16_t Mode`
- `uint8_t TxLastSent [UART_BUFFER_SIZE]`
- `uint8_t TxLastSentPtr`

### 5.4.1 Detailed Description

Definition at line 23 of file `myUart.h`.

The documentation for this struct was generated from the following file:

- `my_lib/myUart.h`

# Index

arrayToNum  
    Lcd\_h, 10

CommandInstance, 21

IRInstance, 21

LCD0Status  
    Lcd\_h, 19

LCD\_CmdInit\_Vector  
    Lcd\_h, 19

LCD\_InitDelay\_Vector  
    Lcd\_h, 19

LCDClear  
    Lcd\_h, 11

LCDDisplayOn  
    Lcd\_h, 11

LCDHome  
    Lcd\_h, 11

LCDInit  
    Lcd\_h, 11

LCDPosition  
    Lcd\_h, 12

LCDPositionNoDelay  
    Lcd\_h, 12

LCDRegisterSpecial  
    Lcd\_h, 13

LCDSend  
    Lcd\_h, 13

LCDSendChar  
    Lcd\_h, 14

LCDSendCmd  
    Lcd\_h, 14

LCDSendHex  
    Lcd\_h, 14

LCDSendNum  
    Lcd\_h, 15

LCDSendNumArray  
    Lcd\_h, 15

LCDSendNumStrict  
    Lcd\_h, 16

LCDSendString  
    Lcd\_h, 16

LCDSendVU  
    Lcd\_h, 17

LCDShift  
    Lcd\_h, 17

LCDStatus, 22

Lcd\_h, 7  
    arrayToNum, 10

LCD0Status, 19

LCD\_CmdInit\_Vector, 19

LCD\_InitDelay\_Vector, 19

LCDClear, 11

LCDDisplayOn, 11

LCDHome, 11

LCDInit, 11

LCDPosition, 12

LCDPositionNoDelay, 12

LCDRegisterSpecial, 13

LCDSend, 13

LCDSendChar, 14

LCDSendCmd, 14

LCDSendHex, 14

LCDSendNum, 15

LCDSendNumArray, 15

LCDSendNumStrict, 16

LCDSendString, 16

LCDSendVU, 17

LCDShift, 17

numToArray, 18

numToArray  
    Lcd\_h, 18

UARTInstance, 22