Noritake itron GU-7000 Code Library

Document Number:	E-M-0127-00
Issue Date:	08/20/2012

Noritake Co., Inc. **Electronics Division Headquarter** 2635 Clearbrook Drive Arlington Heights, IL 60005 Toll free: (800) 779 - 5846 Phone: (847) 439 - 9020 support.ele@noritake.com www.noritake-elec.com	**East Coast** **New Jersey Branch** 15-22 Fair Lawn Ave. Fair Lawn, NJ 07410 Toll free: (888) 296 - 3423 Phone: (201) 475 - 5200 Fax: (201) 796 - 2269	**Midwest, Canada, and Mexico** Chicago Branch 2635 Clearbrook Dr. Arlington Heights, IL 60005 Toll free: (800) 779 - 5846 Phone: (847) 439 - 9020 Fax: (847) 593 - 2285	**West Coast** **Los Angeles Branch** 21081 S. Western Ave. Ste 180 Torrance, CA 90501 Toll free: (888) 795 - 3423 Phone: (310) 320 - 1700 Fax: (310) 320 - 2900

Contents

Using the Library

About the Library

This library provides access to the base functionality of the GU-7000 series modules using 8-bit Atmel® AVR or Linux.

This library is intended for use with the following modules:

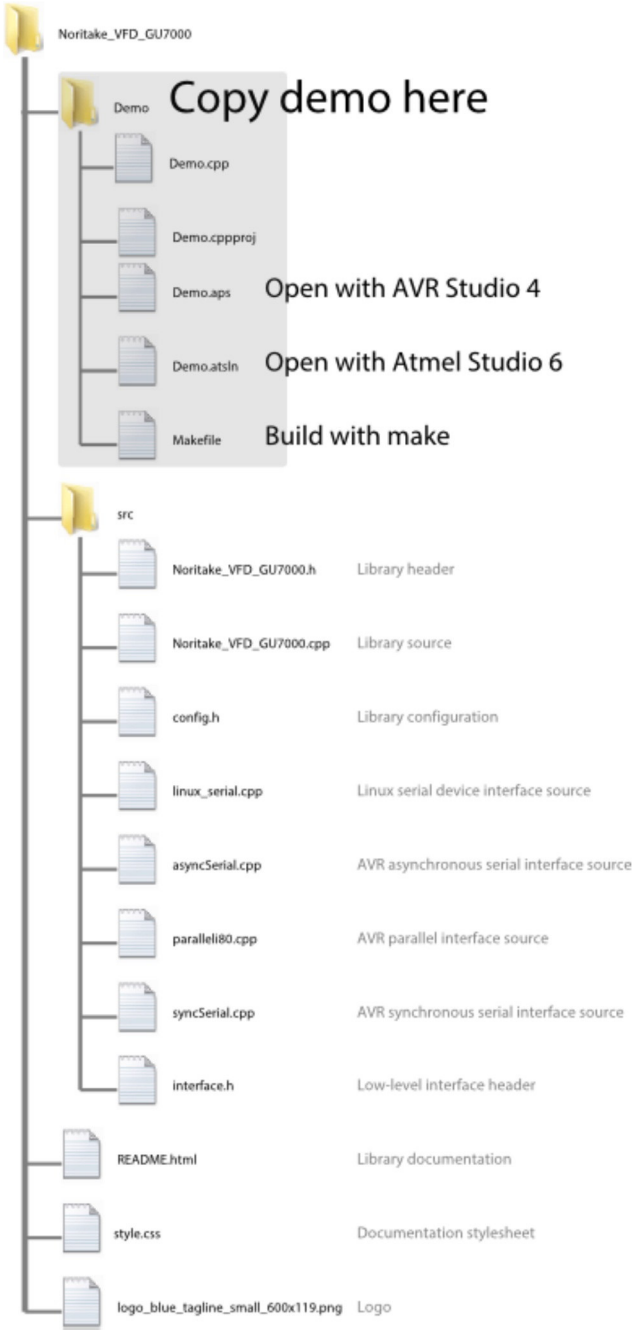
GU***X***-7***

Example:

- GU140X32F-7000
- GU140X16G-7003
- GU160X80E-7900B

Installation

1. Download the code library .
2. Unzip the library file to your work area.
3. Extract the demo folder into the folder for the code library.
4. Set the configuration options in config.h included with the code library. See the *Configuration* section.
5. Open the *Demo.aps* project file with AVR Studio 4 or *Demo.atsln* in Atmel Studio 6 or build *Makefile* on Linux.



Configuration

The library is configured by setting preprocessor values in the *config.h* file in the *src* directory of the library. The configuration options in the code library's *config.h* must match the host system's hardware setup.

The library is designed so that all of its source files are compiled even if they are not used. For example, when set up to use the parallel interface, the serial interface files will automatically disable themselves based on the options set in *config.h*.

Mandatory Options

There are a few mandatory options that must be set regardless of the hardware setup.

Name	Description	Example
F_CPU	CPU frequency in Hertz. This is only used for Atmel® AVR.	16000000UL for 16MHz
NORITAKE_VFD_RESET_DELAY	The delay time in milliseconds before beginning communication with the module. This allows the VFD module to start up as well as avoids program restarts due to in-circuit debuggers. This value will vary depend on the power supply and hardware setup. 500ms is generally sufficient	500 for 500ms
NORITAKE_VFD_HEIGHT	Height of the display in characters. This number is given after the “CU” in the model number: CU__***-Y***	24 for CU24043-Y1A
NORITAKE_VFD_WIDTH	Width of the display in characters. This number is given two digits after the first two digits after the “CU” in the model number: CU**__*-Y***	04 for CU24043-Y1A
NORITAKE_VFD_MODEL_CLASS	Indicates the capabilities of the module. The model class number is the last 4-digit number in the model number: GU***X****-____	7040 for GU140X16G-7040A; 7903 for GU140X16G-7903
NORITAKE_VFD_GENERATION	Indicates the generation of the module. If the last letter in the model number is B, the generation is 'B'. Otherwise, 0.	0 for GU140X16G-7003; 0 for GU140X16G-7040A; 'B' for GU140X16G-7003B;

Asynchronous Serial Interface for Atmel® AVR

Name	Description	Example
NORITAKE_VFD_INTERFACE	0 for serial interface	
NORITAKE_VFD_SERIAL_SYNC	0 for asynchronous serial	
NORITAKE_VFD_RS232	Determines the interface protocol.	0 for models that end in 7003; 1 for models that do not end in 7003
NORITAKE_VFD_BAUD	The baud rate of the device.	38400 for 38400bps

Name	Description	Example
OUT_PORT & OUT_PIN	The serial data SIN port and pin.	OUT_PORT = PORTA and OUT_PIN = 1 for PA1
BUSY_PORT & BUSY_PIN	The serial SBUSY port and pin.	BUSY_PORT = PORTA and BUSY_PIN = 1 for PA1
RESET_PORT & RESET_PIN	The serial RESET port and pin.	RESET_PORT = PORTA and RESET_PIN = 1 for PA1

Synchronous Serial Interface for Atmel® AVR

Name	Description	Example
NORITAKE_VFD_INTERFACE	0 for serial interface	
NORITAKE_VFD_SERIAL_SYNC	1 for synchronous serial	
SCK_PORT & SCK_PIN	The serial data SCK port and pin.	SCK_PORT = PORTA and SCK_PIN = 1 for PA1
OUT_PORT & OUT_PIN	The serial data SIN port and pin.	OUT_PORT = PORTA and OUT_PIN = 1 for PA1
BUSY_PORT & BUSY_PIN	The serial SBUSY port and pin.	BUSY_PORT = PORTA and BUSY_PIN = 1 for PA1
RESET_PORT & RESET_PIN	The serial RESET port and pin.	RESET_PORT = PORTA and RESET_PIN = 1 for PA1

Parallel Interface for Atmel® AVR

Name	Description	Example
NORITAKE_VFD_INTERFACE	1 for parallel interface	
NORITAKE_VFD_BUSY_CONNECTED	Indicates that PBUSY is connected on parallel pin 3. This cannot be set at the same time as NORITAKE_VFD_RESET_CONNECTED. Check the module specification for jumper settings or this option.	0 if PBUSY is shared with D7 1 if PBUSY is connected to parallel pin 3
NORITAKE_VFD_RESET_CONNECTED	Indicates that /RESET is connected on parallel pin 3. This cannot be set at the same time as NORITAKE_VFD_BUSY_CONNECTED. Check the module specification for jumper settings or this option.	0 if /RESET is not connected 1 if /RESET is connected to parallel pin 3

Name	Description	Example
WR_PORT & WR_PIN	The WR signal port and pin.	WR_PORT = PORTA and WR_PIN = 1 for PA1
BUSY_PORT & BUSY_PIN	The PBUSY signal port and pin.	BUSY_PORT = PORTA and BUSY_PIN = 1 for PA1
D0_PORT & D0_PIN	The D0 signal port and pin.	D0_PORT = PORTA and D0_PIN = 1 for PA1
D1_PORT & D1_PIN	The D1 signal port and pin.	D1_PORT = PORTA and D1_PIN = 1 for PA1
D2_PORT & D2_PIN	The D2 signal port and pin.	D2_PORT = PORTA and D2_PIN = 1 for PA1
D3_PORT & D3_PIN	The D3 signal port and pin.	D3_PORT = PORTA and D3_PIN = 1 for PA1
D4_PORT & D4_PIN	The D4 signal port and pin.	D4_PORT = PORTA and D4_PIN = 1 for PA1
D5_PORT & D5_PIN	The D5 signal port and pin.	D5_PORT = PORTA and D5_PIN = 1 for PA1
D6_PORT & D6_PIN	The D6 signal port and pin.	D6_PORT = PORTA and D6_PIN = 1 for PA1
D7_PORT & D7_PIN	The D7 signal port and pin.	D7_PORT = PORTA and D7_PIN = 1 for PA1

Linux Serial Device Interface

The Linux serial device interface uses Linux character device files to interface with the module.

Make sure logged in user has read and write access to the device file before using the library.

You can give non-root users access to the device file with:

```
chmod o+rw /dev/ttyUSB0
```

You must be root or running with elevated privilige through `sudo` or `su` to run this command.

Name	Description	Example

Name	Description	Example
Name	**Description**	**Example**
NORITAKE_VFD_INTERFACE	1 for Linux serial device serial interface	
NORITAKE_VFD_FILE	The path to the device file connected to the module.	/dev/ttyUSB0
NORITAKE_VFD_BAUD	The baud rate of the device.	38400 for 38400bps

Type and Method Reference

Index

    - void GU7000_screenSaver(ScreenSaver mode);
- Fonts
    - void GU7000_setFontStyle(bool proportional, bool evenSpacing);
    - void GU7000_setFontSize(uint8_t x, uint8_t y, bool tall);
    - void GU7000_useCustomChars(bool enable);
    - void GU7000_defineCustomChar(uint8_t code, FontFormat format, const uint8_t *data);
    - void GU7000_deleteCustomChar(uint8_t code, FontFormat format);
- Encoding
    - void GU7000_useMultibyteChars(bool enable);
    - void GU7000_setMultibyteCharset(uint8_t code);
    - void GU7000_setAsciiVariant(AsciiVariant code);
    - void GU7000_setCharset(Charset code);
- Screen Effects
    - void GU7000_invertOn();
    - void GU7000_invertOff();
    - void GU7000_setCompositionMode(CompositionMode mode);
    - void GU7000_setScrollMode(ScrollMode mode);
    - void GU7000_setHorizScrollSpeed(uint8_t speed);
    - void GU7000_scrollScreen(unsigned x, unsigned y, unsigned count, uint8_t speed);
    - void GU7000_blinkScreen();
    - void GU7000_blinkScreen(bool enable, bool reverse, uint8_t on, uint8_t off, uint8_t times);
- Drawing
    - void GU7000_drawImage(unsigned width, uint8_t height, const uint8_t *data);
    - void GU7000_drawImage_p(unsigned width, uint8_t height, const uint8_t *data);
    - void GU7000_drawFROMImage(unsigned long address, uint8_t srcHeight, unsigned width, uint8_t height);
    - void GU7000_drawImage(unsigned x, uint8_t y, ImageMemoryArea area, unsigned long address, uint8_t srcHeight, unsigned width, uint8_t height, unsigned offsetx, unsigned offsety);
    - void GU7000_drawImage(unsigned x, uint8_t y, ImageMemoryArea area, unsigned long address, unsigned width, uint8_t height);
    - void GU7000_drawImage_p(unsigned x, uint8_t y, unsigned width, uint8_t height, const uint8_t *data);
    - void GU7000_drawImage(unsigned x, uint8_t y, unsigned width, uint8_t height, const uint8_t *data);
    - void GU7000_fillRect(unsigned x0, unsigned y0, unsigned x1, unsigned y1, bool on);
- Window and Screen
    - void GU7000_selectWindow(uint8_t window);
    - void GU7000_defineWindow(uint8_t window, unsigned x, unsigned y, unsigned width, unsigned height);
    - void GU7000_deleteWindow(uint8_t window);
    - void GU7000_joinScreens();
    - void GU7000_separateScreens();
- LED Backlight Control
    - void GU7000_setBacklightColor(uint8_t r, uint8_t g, uint8_t b);
    - void GU7000_setBacklightColor(unsigned rgb);

Types

ImageMemoryArea

Identifies the memory area to use in an operation.

Users

- GU7000_drawImage(ImageMemoryArea area, unsigned long address, uint8_t srcHeight, unsigned width, uint8_t height)
- GU7000_drawImage(unsigned x, uint8_t y, ImageMemoryArea area, unsigned long address, uint8_t srcHeight, unsigned width, uint8_t height, unsigned offsetx, unsigned offsety)
- GU7000_drawImage(unsigned x, uint8_t y, ImageMemoryArea area, unsigned long address, unsigned width, uint8_t height)
- GU7000_defineImage(ImageMemoryArea area, unsigned addr, unsigned width, uint8_t height, const uint8_t *data)
- GU7000_defineImage_p(ImageMemoryArea area, unsigned addr, unsigned width, uint8_t height, const uint8_t *data)

- GU7000_scrollImage(ImageMemoryArea area, unsigned long address, uint8_t srcHeight, unsigned width, uint8_t height, uint8_t speed)

Parameters

FlashImageArea	1: FlashROM on the module; contents until it is overwritten
ScreenImageArea	2: reflects the contents of the screen

Only GU-79** modules have the Flash ROM image memory area.

ScrollMode

Identifies scrolling and wrapping behavior.

Users

- GU7000_setScrollMode(ScrollMode mode)

Parameters

WrappingMode	1: When the cursor reaches the right end of the screen, the cursor moves to the left end of the next line. If on the last line, the cursor moves to the home position (0, 0).
VertScrollMode	2: When the cursor reaches the right end of the screen, the cursor moves to the left end of the next line. If on the last line, the screen is moved up by 8 dots. The bottom 8 dots are cleared. The top 8 dots are lost.
HorizScrollMode	3: When the cursor reaches the right end of the screen, the current line is shifted left by the size of one character. The space to the right is cleared. The cursor is not moved.

GU7000_lineFeed() is ignored in this mode.

CompositionMode

Identifies the way that new images and text are combined with the contents that are already on the screen.

Users

- GU7000_setCompositionMode(CompositionMode mode)

Parameters

NormalCompositionMode	0: screen image is cleared and the source appears exactly
OrCompositionMode	1: dots are lit if either the source or screen image specify it lit
AndCompositionMode	2: dots are lit only if both the source and screen image specify it lit
XorCompositionMode	2: dots are lit only if either the source or screen image specify it lit but not both

FontFormat

Identifies the size and data format

Users

- GU7000_setFontSize(FontFormat format, uint8_t x, uint8_t y)
- GU7000_defineCustomChar(uint8_t code, FontFormat format, const uint8_t *data)

- GU7000_deleteCustomChar(uint8_t code, FontFormat format)

Parameters

GU70005x7Format	1: 5×7 font
GU70007x8Format	1: 5×7 font
CUUFormat	0x81: 5×8 font. This format is only for defining 6×8 custom characters in a different format; it does not exist on the module, itself

AsciiVariant

Identifies national ASCII variants.

Users

- GU7000_setAsciiVariant(AsciiVariant code)

Parameters

AmericaAscii	0
FranceAscii	1
GermanyAscii	2
EnglandAscii	3
Denmark1Ascii	4
SweedenAscii	5
ItalyAscii	6
Spain1Ascii	7
JapanAscii	8
NorwayAscii	9
Denmark2Ascii	10
Spain2Ascii	11
LatinAmericaAscii	12
KoreaAscii	13

Charset

Identifies code page.

Users

- GU7000_setCharset(Charset code)

Parameters

CP437	0
EuroStdCharset	0: CP437
Katakana	1
CP850	2
MultilingualCharset	2: CP850

CP860	3
PortugeseCharset	3: CP860
CP863	4
CanadianFrenchCharset	4: CP863
CP865	5
NordicCharset	5: CP865
CP1252	0x10
CP866	0x11
Cyrillic2Charset	0x11: CP866
CP852	0x12
Latin2Charset	0x12: CP852
CP858	0x13

MultibyteCharset

Identifies a multibyte character set.

Users

- GU7000_setMultibyteCharset(uint8_t code)

Parameters

ShiftJIS	0: JIS (X0208 Shift-JIS)
JapaneseMBCS	0: ShiftJIS
KSC5601	1: KSC5601-87
KoreanMBCS	1: KSC5601
GB2312	2: GB2312-80
SimplifiedChineseMBCS	2: GB2312
Big5	3: Big5
TraditionalChineseMBCS	3: Big5

ScreenSaver

Screen saver.

Users

- GU7000_screenSaver()

Parameters

AllDotsOffSaver	turns all dots off
AllDotsOnSaver	turns all dots on
InvertSaver	

Printing Methods

void print(char c);

Print a character.

Parameters

c	character to print

See GU7000_setScrollMode(ScrollMode mode) for information on scrolling and wrapping.

void print(const char *str);

Print a string.

Each character is printed with print(char c).

Parameters

str	string to print

void print(const char *buffer, size_t size);

Print a buffer of the given size.

Each character is printed with print(char c).

Parameters

buffer	characters to print
size	number of characters to print

void print(int number, uint8_t base);

Print a number in the given base.

Each character is printed with print(char c).

No leading space or zeros are used.

Parameters

number	number to print
base	base to print in: 2 ≤ base ≤ 36

Arithmetic with a long, unsigned, or constant larger than INT_MAX will cause the long, unsigned, or unsigned long overload of this function to be used.

void print(unsigned number, uint8_t base);

Print a number in the given base.

No leading space or zeros are used.

Each character is printed with print(char c).

Parameters

number	number to print
base	base to print in: 2 ≤ base ≤ 36

Arithmetic with a long or constant larger than UINT_MAX will cause the long or unsigned long overload of this function to be used.

void print(long number, uint8_t base);

Print a number in the given base.

No leading space or zeros are used.

Each character is printed with print(char c).

Arithmetic with an unsigned long or constant larger than LONG_MAX will cause the unsigned long overload of this function to be used.

Parameters

number	number to print
base	base to print in: 2 ≤ base ≤ 36

void print(unsigned long number, uint8_t base);

Print a number in the given base.

No leading space or zeros are used.

Each character is printed with print(char c).

Parameters

number	number to print
base	base to print in: 2 ≤ base ≤ 36

void print(unsigned x, uint8_t y, char c);

Print a character from (x, y).

The cursor is not moved.

Characters can be inverted for the duration of the command by sending character '\x11' and disabled by '\x10'. Inverting from GU7000_invertOn() affects the characters printed with this command and can be disabled with '\x10'.

Normal commands cannot be issued inside this command. This command is only available on Generation B.

Parameters

x	x coordinate of top-left corner; -1 continues from the last unaligned print
y	y coordinate of top-left corner
c	character to print 0x10 disables inverted characters 0x11 enables inverted characters

void print(unsigned x, uint8_t y, const char *str);

Print a string from (x, y).

No more than 255 characters can be printed at once. This command is only available on Generation B. See print(unsigned x, uint8_t y, char c) for additional information.

Parameters

x	x coordinate of top-left corner; -1 continues from the last unaligned print
y	y coordinate of top-left corner
str	string to print

void print(unsigned x, uint8_t y, const char *buffer, uint8_t len);

Print a buffer from (x, y).

No more than 255 characters can be printed at once.
This command is only available on Generation B.
See print(unsigned x, uint8_t y, char c) for additional information.

Parameters

x	x coordinate of top-left corner; -1 continues from the last unaligned print
y	y coordinate of top-left corner
buffer	characters to print
len	number of characters to print

void print(unsigned x, uint8_t y, int number, uint8_t base);

Print a number from (x, y).

Note that there are no long int overloads.
This command is only available on Generation B.

Parameters

x	x coordinate of top-left corner; -1 continues from the last unaligned print
y	y coordinate of top-left corner
number	number to print
base	base to print in: 2 ≤ base ≤ 36

void print(unsigned x, uint8_t y, unsigned number, uint8_t base);

Print a number from (x, y).

Note that there are no long int overloads.
This command is only available on Generation B.

Parameters

x	x coordinate of top-left corner; -1 continues from the last unaligned print
y	y coordinate of top-left corner
number	number to print
base	base to print in: 2 ≤ base ≤ 36

void print_p(const char *str);

Print a string from host's ROM.

No more than 255 characters can be printed at once.

Parameters

str	string to print

void print_p(unsigned x, uint8_t y, const char *str);

Print a string from (x, y) from the host's ROM.

No more than 255 characters can be printed at once.
This command is only available on Generation B.

Parameters

x	x coordinate of top-left corner; -1 continues from the last unaligned print
y	y coordinate of top-left corner

str	string to print

void print_p(unsigned x, uint8_t y, const char *buffer, uint8_t len);

Print a buffer from (x, y) from the host's ROM.

This command is only available on Generation B.

Parameters

x	x coordinate of top-left corner; -1 continues from the last unaligned print
y	y coordinate of top-left corner
str	characters to print
len	number of characters to print

void println(char c);

Print a character and go to the next line as if GU7000_carriageReturn() and GU7000_lineFeed() were called.

Parameters

c	character to print

void println(const char *str);

Print a string and go to the next line as if GU7000_carriageReturn() and GU7000_lineFeed() were called.

Parameters

str	string to print

void println(const char *buffer, size_t size);

Print a buffer of the given size and go to the next line as if GU7000_carriageReturn() and GU7000_lineFeed() were called.

Parameters

buffer	characters to print
size	number of characters to print

void println(int number, uint8_t base);

Print a number in the given base and go to the next line as if GU7000_carriageReturn() and GU7000_lineFeed() were called.

Arithmetic with a long, unsigned, or constant larger than INT_MAX will cause the long, unsigned, or unsigned long overload of this function to be used.

No leading space or zeros are used.

Each character is printed with print(char c).

Parameters

number	number to print

base	base to print in: 2 ≤ base ≤ 36

void println(unsigned number, uint8_t base);

Print a number in the given base.

No leading space or zeros are used and go to the next line as if GU7000_carriageReturn() and GU7000_lineFeed() were called.

Each character is printed with print(char c).

Arithmetic with a long or constant larger than UINT_MAX will cause the long or unsigned long overload of this function to be used.

Parameters

number	number to print
base	base to print in: 2 ≤ base ≤ 36

void println(long number, uint8_t base);

Print a number in the given base and go to the next line as if GU7000_carriageReturn() and GU7000_lineFeed() were called.

No leading space or zeros are used.

Each character is printed with print(char c).

Arithmetic with an unsigned long or constant larger than LONG_MAX will cause the unsigned long overload of this function to be used.

Parameters

number	number to print
base	base to print in: 2 ≤ base ≤ 36

void println(unsigned long number, uint8_t base);

Print a number in the given base and go to the next line as if GU7000_carriageReturn() and GU7000_lineFeed() were called.

No leading space or zeros are used.

Each character is printed with print(char c).

Parameters

number	number to print
base	base to print in: 2 ≤ base ≤ 36

Character-Based

void GU7000_back();

Move the cursor back one character position in the current font.

If the cursor is within one character position of the left of the screen, the cursor moves to the last character position of the previous line. If on the first line when this happens, the cursor does not move.

void GU7000_forward();

Move the cursor forward one character position in the current font.

If the cursor is within one character position of the right of the screen, the cursor moves to the first character position of the next line. If on the last line when this happens, the cursor moves to the home position (0, 0).

void GU7000_lineFeed();

Move the cursor down one line.

This command is ignored in horizontal scrolling modes.

If the cursor is on the last line, then the cursor is moved to the home position (0, 0).

void GU7000_home();

Move the cursor to the home position (0, 0).

void GU7000_carriageReturn();

Move the cursor to the beginning of the current line.

void GU7000_setCursor(unsigned x, unsigned y);

Set cursor to the given postion.

If the position is invalid, the command is ignored.

This command may be used to set the cursor into the hidden memory area.

Parameters

x	target x coordinate
y	target y coordinate

void GU7000_cursorOn();

Turn the cursor on.

void GU7000_cursorOff();

Turn the cursor off.

Module Control

void GU7000_clearScreen();

Clear the screen and move the cursor to the home position (0, 0) of the current screen.

If in separate screen mode, only one of the two screens is cleared. If in the hidden memory area, the cursor returns to (0, 0) of the hidden memory area not of the visible screen.

void GU7000_init();

Initialize the module.

void GU7000_reset();

Reset the module.

void GU7000_setScreenBrightness(unsigned level);

Set screen brightness.

Parameters

level	percent of capacity (rounded to next 12.5%): 0 ≤ level ≤ 100

void GU7000_wait(uint8_t time);

Stop processing commands for the given time period.

Parameters

time	time × .5s

void GU7000_displayOn();

Turn the display on. The screen may fade on.

void GU7000_displayOff();

Turn the display off. The screen may fade off.

Turning the display off is saves more power than simply turning all dots off or setting the brightness to 0% with GU7000_setScreenBrightness(unsigned level).

void GU7000_screenSaver(ScreenSaver mode);

Set screen saver.

Screen savers are disabled if a command is received.

Parameters

mode	screen saver mode; See ScreenSaver

Fonts

void GU7000_setFontStyle(bool proportional, bool evenSpacing);

Set the font style.

Parameters

proportional	true uses a proportional font false uses a fixed-width font
evenSpacing	true inserts an extra dot on the right

void GU7000_setFontSize(uint8_t x, uint8_t y, bool tall);

Set the magnification of the font.

Characters already on the screen are not affected.

Parameters

x	width magnification: 1 ≤ x ≤ 4
y	height magnification: 1 ≤ y ≤ 4
tall	uses the 8×16 font; this is required to enable multibyte character sets; See GU7000_useMultibyteChars(bool enable)

void GU7000_useCustomChars(bool enable);

Enables or disables the use of custom characters.

Custom characters are not deleted if this is called with false; they may be used again when custom characters are re-enabled.

Parameters

enable	true enables custom characters; false disables them

void GU7000_defineCustomChar(uint8_t code, FontFormat format, const uint8_t *data);

Redefine the appearance of the given character code. Redefining a character does not change the appearance of characters already on screen. Custom characters may be redefined multiple times without deleting them.

Custom characters must be enabled with GU7000_useCustomChars (bool enable) to see changes.

Only 16 characters may be defined per format.

Custom characters are destroyed when the module is reset or the initialize command is used. Delete characters with GU7000_deleteCustomChar(uint8_t code, FontFormat format).

Parameters

code	character code 0x20 ≤ code ≤ 0xff (except 16×16 and 32×32) 0xec40 ≤ code ≤ 0xec4f for 16×16 and 32×32 Japanese 0xfea1 ≤ code ≤ 0xfeb0 for 16×16 and 32×32 except Japanese
format	size and format of data; see FontFormat
data	display data; see the module specification for format

void GU7000_deleteCustomChar(uint8_t code, FontFormat format);

Delete a custom character.

The command is ignored if a character was not defined.

Parameters

code	character code 0x20 ≤ code ≤ 0xff (except 16×16 and 32×32) 0xec40 ≤ code ≤ 0xec4f for 16×16 and 32×32 Japanese 0xfea1 ≤ code ≤ 0xfeb0 for 16×16 and 32×32 except Japanese
format	size and format of data; see FontFormat

Encoding

void GU7000_useMultibyteChars(bool enable);

Enable or disable the use of multibyte character sets.

Use GU7000_setMultibyteCharset(uint8_t code) to select the character set.

Multibyte charactersets are only available on GU-79** modules.

Multi-byte characters may only be printed when the 8×16 font is selected. 12×16 may be used for Japanese.

Parameters

enable	true enables multibyte character sets; false disables them

void GU7000_setMultibyteCharset(uint8_t code);

Set the multibyte character set.

Use GU7000_useMultibyteChars(bool enable) to enable multicharacter sets.

Multibyte charactersets are only available on GU-79** modules.

Multi-byte characters may only be printed when the 8×16 font is selected. 12×16 may be used for Japanese.

Parameters

code	number code for the characterset; see MultibyteCharset

void GU7000_setAsciiVariant(AsciiVariant code);

Select the national ASCII variant.

The appearance of these characters changes:

0x23	0x24	0x40	0x5b	0x5c	0x5d	0x5e	0x60	0x7c	0x7d	0x7e
'#'	'$'	'@'	'['	'\'	']'	'^'	'`'	'{'	'|'	'}'

Characters already on the screen are not affected.

Parameters

code	ASCII variant code; see AsciiVariant

void GU7000_setCharset(Charset code);

Select the character set (code page).

The appearance of characters 0x80 - 0xff changes according to the table for the selected character set.

Characters already on the screen are not affected.

This only affects 8-bit character sets. Use GU7000_setMultibyteCharset(uint8_t code) to change the multibyte character set.

Parameters

code	Character set code; see Charset

Screen Effects

void GU7000_invertOn();

Turn inversion on. Dots that would have been been lit will be drawn unlit and vice versa.

Text and images already on screen are not affected.

void GU7000_invertOff();

Turns inversion off.

Text and images already on screen are not affected.

void GU7000_setCompositionMode(CompositionMode mode);

Determine the way that new images and text are combined with the contents that are already on the screen.

Parameters

mode	composition mode; see CompositionMode

void GU7000_setScrollMode(ScrollMode mode);

Select the behavior when text reaches the end of the screen.

No commands are processed while the module is scrolling.

Parameters

mode	scroll mode; see ScrollMode

void GU7000_setHorizScrollSpeed(uint8_t speed);

Set the delay between scrolling animation steps.

No commands are processed while the module is scrolling. Timing differs per module: $13 \le T \le 15$

Parameters

speed	0 prevents any delay between steps 1 *T* ms / 2 dots (n - 1) × *T* ms / dot

void GU7000_scrollScreen(unsigned x, unsigned y, unsigned count, uint8_t speed);

Scroll the contents of the screen by (x, y) leftwards/upwards.

This command only moves the viewing area. The visible screen area is moved left/up and the right / bottom edge now appears in the hidden memory area.

No commands are processed while the module is scrolling.

The cursor remains in place. It does not scroll with the cursor.

Scrolling can be used for page flipping if the module has enough display memory for two full screens. Set the cursor into the hidden memory area and draw he next frame. When rendering is complete, scroll the width of a screen to exchange the hidden memory area with the previous visible memory area. The cursor will remain in the hidden memory area.

Parameters

Timing differs per module: $13 \le T \le 15$

x	number of dots to move left
y	number of dots to move up rounded down to the next 8 dots
count	number of steps to break animation into
speed	delay between each step; speed × T

void GU7000_blinkScreen();

Stop the screen from blinking. Blink the screen with GU7000_blinkScreen(bool enable, bool reverse, uint8_t on, uint8_t off, uint8_t times).

void GU7000_blinkScreen(bool enable, bool reverse, uint8_t on, uint8_t off, uint8_t times);

Enable or disable screen blinking.

Parameters

enable	true enables screen blinking; false disables it
reverse	true reverses the screen on 'off' cycles; false clears the screen
on	time to display the normal screen image; on × T
off	time to display the inverted or blank screen image; on × T
times	number of times to blink the screen off; 0 blinks until vfd.GU7000_blinkScreen (false, false, 1, 1, 1);

Timing differs per module: $13 \le T \le 15$

Drawing

void GU7000_drawImage(unsigned width, uint8_t height, const uint8_t *data);

Draw image at the cursor from host RAM.

If height is larger than the screen, then the call will be ignored.

The cursor is not moved.

Parameters

width	width of image; may be less than the width of the image in memory
height	height of image; must be the height of the image in memory rounded down to the next 8 dots
data	image data; see Image Format

void GU7000_drawImage_p(unsigned width, uint8_t height, const uint8_t *data);

Draw image at the cursor from host ROM.

The cursor is not moved.

If height is larger than the screen, then the call will be ignored.

Parameters

width	width of image; may be less than the width of the image in memory
height	height of image; must be the height of the image in memory rounded down to the next 8 dots
data	image data; see Image Format

void GU7000_drawFROMImage(unsigned long address, uint8_t srcHeight, unsigned width, uint8_t height);

Draw image at the cursor from the module's Flash image memory area (FROM).

The source image starts from (offsetx, offsety) with the origin at the byte specified by address.

Only GU-79** modules have the Flash ROM image memory area.

If height or srcHeight is larger than the screen, then the call will be ignored.

Parameters

address	address in the memory area
srcHeight	height of image in memory memory rounded down to the next 8 dots
width	width of image; may be less than the width of the image in memory
height	height of image; may be less than the height of the image in memory

void GU7000_drawImage(unsigned x, uint8_t y, ImageMemoryArea area, unsigned long address, uint8_t srcHeight, unsigned width, uint8_t height, unsigned offsetx, unsigned offsety);

Draw image at (x, y) from the module's image memory areas.

The source image starts from (offsetx, offsety) with the origin at the byte specified by address.

If height or srcHeight is larger than the screen, then the call will be ignored.
This command is only available on Generation B.

Parameters

x	x coordinate of top-left corner
y	y coordinate of top-left corner
area	area in which the bitmap is stored; see ImageMemoryArea
address	address in the memory area
srcHeight	height of image in memory memory rounded down to the next 8 dots
width	width of image; may be less than the width of the image in memory
height	height of image; may be less than the height of the image in memory
offsetx	x coordiante offset from source address
offsety	y coordiante offset from source address

void GU7000_drawImage(unsigned x, uint8_t y, ImageMemoryArea area, unsigned long address, unsigned width, uint8_t height);

Draw image at (x, y) from the module's image memory areas.

The source image starts from the byte specified by address.

If height is larger than the screen, then the call will be ignored. Unlike other functions, this rounds height up since images with heights not divisible by 8 may be drawn. This command is only available on Generation B.

Parameters

x	x coordinate of top-left corner
y	y coordinate of top-left corner
area	area in which the bitmap is stored; see ImageMemoryArea
address	address in the memory area
width	width of image; may be less than the width of the image in memory
height	height of image rounded **up** to the next 8 dots

void GU7000_drawImage_p(unsigned x, uint8_t y, unsigned width, uint8_t height, const uint8_t *data);

Draw image at (x, y) from host RAM.

If height is larger than the screen, then the call will be ignored. This command is only available on Generation B.

Parameters

width	width of image; may be less than the width of the image in memory
height	height of image; must be the height of the image in memory rounded down to the next 8 dots
data	image data; see Image Format

void GU7000_drawImage(unsigned x, uint8_t y, unsigned width, uint8_t height, const uint8_t *data);

Draw image at (x, y) from host ROM.

If height is larger than the screen, then the call will be ignored. This command is only available on Generation B.

Parameters

width	width of image; may be less than the width of the image in memory
height	height of image; must be the height of the image in memory rounded down to the next 8 dots
data	image data; see Image Format

void GU7000_fillRect(unsigned x0, unsigned y0, unsigned x1, unsigned y1, bool on);

Draw a filled rectangle from (x0,y0)-(x1,y1).

Parameters

x0	x coordinate of the top-left corner
y0	y coordinate of the top-left corner
x1	x coordinate of the bottom-right corner

y1	y coordinate of the bottom-right corner
on	true lights the rectangle; false turns the dots of the rectangle off

Window and Screen

void GU7000_selectWindow(uint8_t window);

Select the base window or one of the user-defined windows.

Parameters

window	0 selects the base window 1 ≤ window ≤ user-defined window

void GU7000_defineWindow(uint8_t window, unsigned x, unsigned y, unsigned width, unsigned height);

Define or redefine a window from (x,y) to (x+width, y+height).

Parameters

window	1 ≤ window ≤ user-defined window
x	x coordinate of upper-left corner
y	x coordinate of upper-left corner
width	width of the window; may not be past the edge of the visible screen
height	height of the window; may not be past the edge of the visisble screen

void GU7000_deleteWindow(uint8_t window);

Delete a user-defined window.

The command is ignored if the user-defined window is not defined.

Parameters

window	1 ≤ window ≤ user-defined window

void GU7000_joinScreens();

Treat the display memory as one screen.

Text scrolls and wraps when the cursor reaches the right end of the hidden memory area.

void GU7000_separateScreens();

Treat the display memory as two separate screens: visible screen area and hidden memory area.

LED Backlight Control

void GU7000_setBacklightColor(uint8_t r, uint8_t g, uint8_t b);

Set the color of the LED backlight.

Parameters

r	red component (16 shades); 0 ≤ r ≤ 255
g	green component (16 shades); 0 ≤ g ≤ 255
b	blue component (16 shades); 0 ≤ b ≤ 255

void GU7000_setBacklightColor(unsigned rgb);

Set the color of the LED backlight.

Parameters

rgb	R8G8B8 color; each component has 16 shades