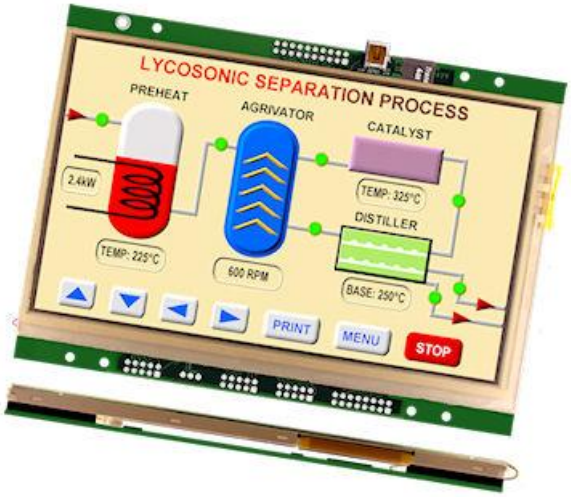
Ineltro AG
Riedthofstrasse 100
CH-8105 Regensdorf
Tel +41 (0)43 3437300
contact@ineltro.ch
www.ineltro.ch



Noritake itron

Product No	TU800x480C-XXX
Issue No	1v34
Document Ref	42782

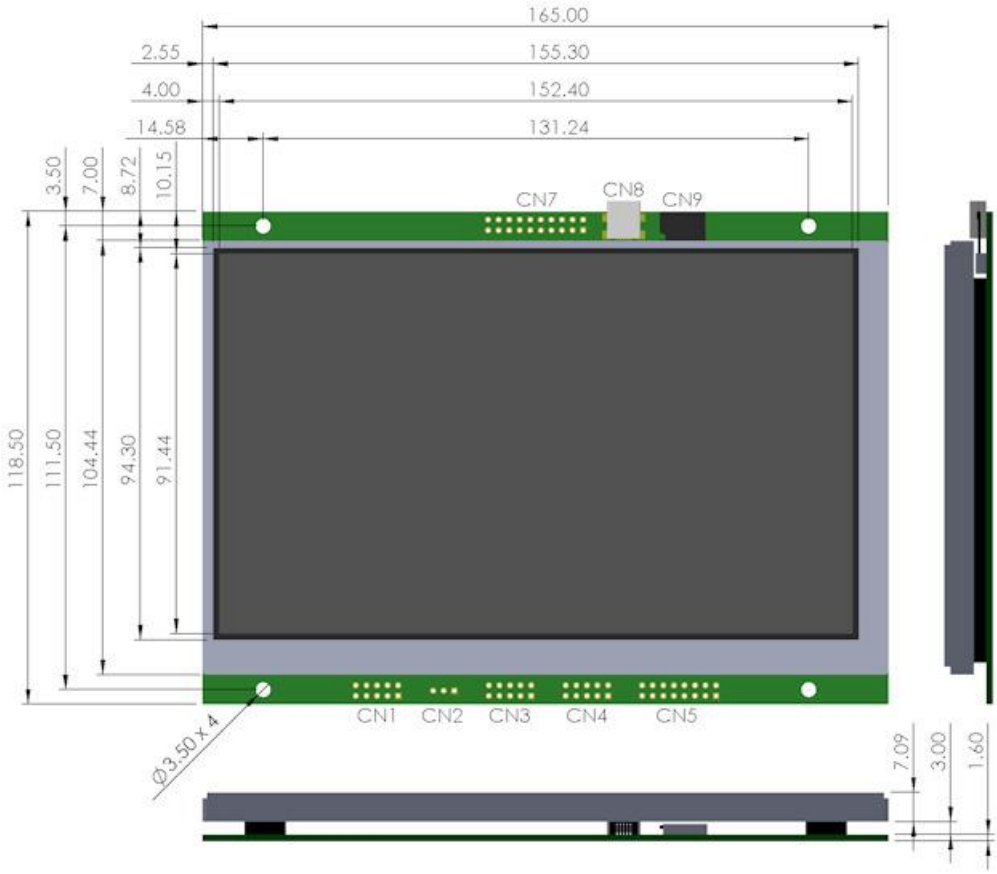
PRELIMINARY
Samples Q4/2010
7.0"iSMART TFT Module

800x480 pixels
262,144 Colours (18 bit)
40 Page Display RAM
128M Byte Flash
4G+ Micro SDHC Slot
LED Backlight Control
5V Supply 3.3V Logic
ASCII + UNICODE Fonts

Full RS232 Port
SPI - I2C Interfaces
Sync Serial Controller
USB 2.0 Interface
Analogue Touch Screen
Up to 12 x 12 Key Control
Up to 24 User Digital I/O
2 Analogue Inputs
2 PWM Outputs
Real Time Clock + Date

Run Animations
Auto Menu Control
Screen Rotation - 90, 180
Object Oriented Language
Graphic User Interface
Integrated Debugger

Downloads
Specification -



This product has been designed to simplify the implementation of TFT technology into your product. The high level text based object oriented command structure, entity library and 100 page screen memory allow most of the processing to be undertaken by the TFT module leaving the host CPU to concentrate on the core application processes. This allows proven firmware running on small 8 bit microcontrollers to be modified to drive this TFT module with a minimum of work and risk.

Module Part Number	Price**	RS232	RS485	Touch	USB CN8	Battery Holder	CANBUS Adaptor	Note
TU800X480C-K612A1		Yes	-	-	-	-	-	-
-K612A1T	€134	Yes	-	Yes	-	-	-	Standard
-K612A1TU*	€135	Yes	-	Yes	Yes	-	-	Can use as Dev Kit
-K612A1TUB	€136	Yes	-	Yes	Yes	Yes	-	Battery not included
-K612A1TUBC	€150	Yes	-	Yes	Yes	Yes	Yes	Battery not included
TU800X480C-K611A1		Yes	Yes	-	-	-	-	-
-K611A1T	€140	Yes	Yes	Yes	-	-	-	-
-K611A1TU	€141	Yes	Yes	Yes	Yes	-	-	-
-K611A1TUB	€142	Yes	Yes	Yes	Yes	Yes	-	Battery not included
-K611A1TUBC	€156	Yes	Yes	Yes	Yes	Yes	Yes	Battery not included

* Main distributor stock item, other versions supplied to order. Pre-fitted connector options are available.
** Unit price excludes freight and VAT. Option to pay in GBP £ , Orders will be delivered from December 20th

Optical & Environmental Parameters

Screen Type	800x480 pixels - RGB Stripe - Pixel Pitch 0.19x0.19mm
Display Area	152x91mm - 7.0" diagonal
RGB Colours	262,144 (18 bit)
Display Type	Transmissive
Contrast Ratio	250:1
View Angle (typ)	60 degrees
Response Time	25ms @ 25C
Default Viewing Angle	6 o'clock (12 o'clock-Invert the PCB and set 180 degrees orientation in software)
Operating Temperature	-20C to +70C
Storage Temperature	-30C to +80C
Humidity	20% to 70% RH
Vibration	10-55-10Hz, all amplitude 1mm, 30Min., X-Y-Z (Non operating)
Shock	392m/s^2 (40G) 9mS X-Y-Z, 3 times each direction (Non operating)

Electrical Parameters

Parameter		Sym	Min	Typ	Max	Unit	Condition	Note
Supply Voltage		VCC	4.5	5	5.5	VDC	VSS=0V	Absolute Max 6.0VDC
Logic Supply Output		VDD	3.2	3.3	3.4	VDC	VCC=5V	Max50mA
Logic Input Voltage	**"H"**	VIH	-0.5	-	3.4 (1)	VDC	VCC=5V	/RES, K0-K24, SCK, /SS, HB, SIN, SCL,SDA
	"L"	VIL	VSS	-	VSS+0.5	VDC	VSS=0V	
Logic Output Voltage	**"H"**	VOH	3.0	-	3.4	VDC	IOH=2mA VCC=5v	K0-K24, SDA, SCL, SOUT, MB
	"L"	VOL	0	-	0.7	VDC	IOL=-2mA VCC=5V	
"H" Level Logic Input Current		IIH	-	-	1.0	uADC	VCC=5.5V	/RES, K0-K24, SCK, /SS, SIN, SCL, SDA
"L" Level Logic Input Current		IIL	-	-	1.0	uADC	VCC=5.5V	
RS232 Input Voltage	**"H"**	VIH	2	-	15	VDC	VCC=5V	RXD, CTS, DSR
	"L"	VIL	-15	-	VSS+0.5	VDC	VCC=5V	
RS232 Output Voltage	**"H"**	VOH	4	7	-	VDC	3kΩ to GND VCC=5V	TXD, DTR, RTS
	"L"	VOL	-	-7	-4	VDC	3kΩ to GND VCC=5V	
Power Supply Current 1		ICC1	580	620	650	mADC	VCC=5V	All dots on
Power Supply Current 2		ICC2	250	270	320	mADC	VCC=5V	LED Backlight Off
Power Supply Current 3		ICC3	50	60	70	mADC	VCC=5V	Reset LOW

Note (1) The voltage applied to logic signals must not exceed the rising VCC at power on as this could affect module initialisation

Connector Pin Assignment

CON	Function	1	2	3	4	5	6	7	8	9	10	Note
CN1	Full RS232 Port (6 signals)	NC	DTR	TXD	CTS	RXD	RTS	DSR	NC	GND	NC	Fits IDC 9 way D type
CN2	5V Power In / Piezo to GND	5V	/PZ	0V	-	-	-	-	-	-	-	Connect piezo negative
CN3	I2C Serial Mode	5V	SCL	-	SDA	0V	-	-	/RES	MB	HB	3v3 Logic (5v in option)
	Asynchronous Serial Mode	5V	-	SI	-	0V	-	SO	/RES	MB	HB	3v3 Logic (5v in option)
	Clock Serial / SPI Mode	5V	-	RD	RCK	0V	/RSS	TD	-	/TSS	TCK	SPI join 4 + 10 and 6+9
CN4	Analogue In, PWN Audio	AN1	AN2	0V	5V	PW1	PW2	ATX	ARX	ACK	AFS	AC97 Audio Pins 7-10
	User I/O	K16	K17	0V	5V	K18	K19	K20	K21	K22	K23	Additional I/O

CON	Function	1/2	3/4	5/6	7/8	9/10	11/12	13/14	15/16	17/18	19/20	Note
CN5	USB/ SD Card Extension	DA2	CDA	CK	DA0	0V	0V	DM	CNX	-	-	SD Card Pins 1-10
		DA3	3V3	0V	DA1	CD	5V	DP	0V	-	-	USB Pins 11-16
CN7	8x8 Keyboard Matrix and user I/O Ports	5V	3V3	K0	K2	K4	K6	K8	K10	K12	K14	3V3 output max 50mA
		0V	0V	K1	K3	K5	K7	K9	K11	K13	K15	

CN8	USB Connector	Standard Mini B can be omitted on user request. 5V power is then provided from the PC
CN9	SD Card Slot	Micro SD Card holder allows permanent installation for large storage

5V pins are common un-fused input /outputs. 3V3 pins are outputs only with a total 50mA capacity. Do not connect pins '-' or NC

Accessories

Noritake- Itron offers a range of accessories to get you up and running quickly.

USB Cable
IFCKUSBminiB2M

RS232 Cable
IFCK232-610A

CAN Bus Interface
EMBCK33A
Maximum speed 1MHz

More Details...

AC97 Audio Module
MCBK-AC97P1
Bi-directional stereo codec and amplifier

TBA

Battery Holder
CONFSCR1216
Uses a CR1216 battery
Solders to rear of TFT

USB-SD Expander
CBK-USBSD1
Supplied with bezel for panel mounting

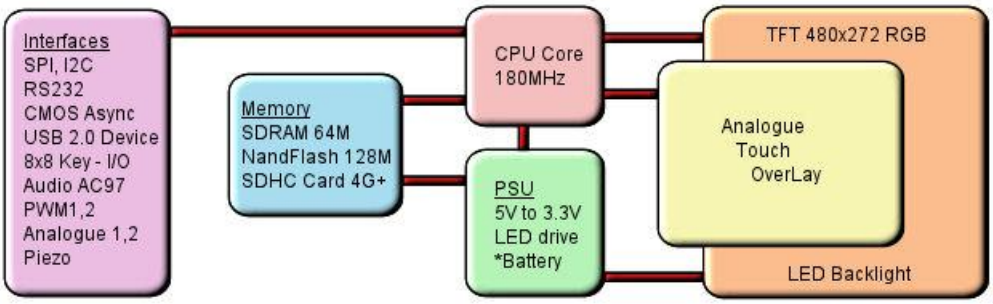
TBA

IDC Interface Cable
IFCK10DC10-200A
10 way 200mm length

iSMART TFT Module Overview

Product Overview
This product has been designed to simplify the implementation of TFT technology into your product.
The high level text based object oriented command structure, entity library and multi page screen memory allow most of the processing to be undertaken by the TFT module leaving the host CPU to concentrate on the core application processes. This allows proven firmware running on small 8 bit microcontrollers to be modified to drive this TFT with a minimum of work and risk.

Hardware for 4.3"

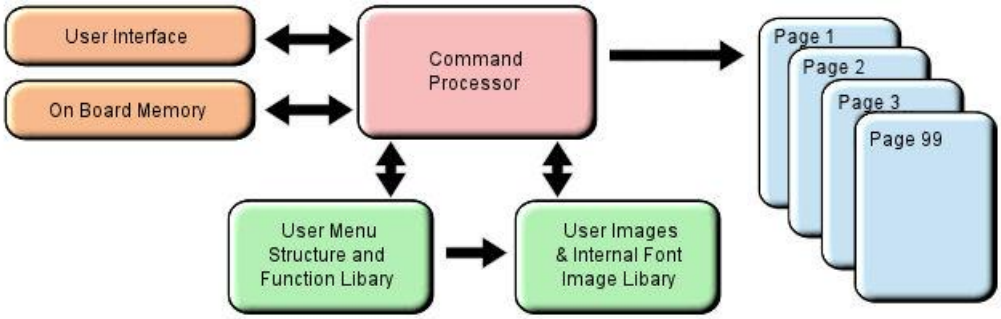


*option

Software Overview
Several customers have asked why we developed our own object oriented programming language rather than provide a product with Linux or an operating system supporting compiled 'C'. If we look back at the original requirements we can see some of the reasons.

1/ The customer's end user or distributor could write code and insert images to add in their own functionality with a text editor.
2/ The program code could be updated or expanded by the host system using ASCII text over a serial link.
3/ The product should be license free and use simple development tools.
4/ The customer can create his own large images and control them like fonts.
5/ The SD card should be able to stream video and audio with the minimum of user programming.
6/ Existing host software need only have limited changes to upgrade a display from 4X20 LCD to a full colour TFT.
7/ The module has the intelligence to operate as a host and the compact command language to act as a high speed slave.
8/ The number of commands should be minimized by using 'overloading' and provide a higher level of functionality than C functions.
9/ The parameters for interfaces and screen entities should be held in styles similar to HTML.
10/ The application development time should take days rather than weeks.

These reasons may not be key to your application, but we believe it does offer new product opportunities.



High Level Object Oriented Commands
The module has an integrated compiler and debugger so that users can write the high level object oriented language commands in a text file or send via an interface to develop their application. Although pictures and fonts can be loaded via an interface, it is best to store these on an SD card or transfer via USB from on a PC. The multi faceted commands are divided into 4 groups as shown below.
You may be thinking how can 25 commands operate a host system, so lets take a look at the **LOAD** command. It can perform the equivalent language functions of strcpy, strcat, format, inp, outp and a page collation function. Please study our application example code for an understanding of this compact language.

library & system	page & visibility	draw on page	functions
FPROG Load Menu/Img to Flash	**PAGE** Create a page of entities	**POSN** Position cursor on page	**FUNC** Create a function
LIB Load Image/Font to RAM	**STYLE** Set parameters	**TEXT** draw text on page	**VAR** Create a variable
INC Include a sub file	**SHOW** Show a page or entity	**DRAW** draw box, circle, line, pixel	**IF ? :** Conditional test
RUN Call function or user code	**HIDE** Hide a page or entity	**IMG** draw image on page	**LOOP** Repeat commands
RESET Reset system, library, time	**DEL** Delete entity from Library	**KEY** create touch or external key	**CALC** Calculation and string edit
;; Refresh current page	**LOAD** Copy and format pages,		**WAIT** Set delay period
; Terminate command	strings, interface and data		**INT** Set an interrupt

Styles make your Application Consistent
All entities and buffers use parameters stored in a Style similar to HTML web pages. These are extensive and define colours, entity types, buffer size and interface parameters like baud rate, clock edges and data format. Styles can be embedded in parent styles to reduce repetition and simplify changes.

Screen Page Creation and Control
Pages can be smaller than the screen for pop up help menus, status information and lists. Buttons can be varying size, with radio, rectangle or check box style with special types for navigation actions. The cursor position command allows relative or absolute positioning for reduced instructions during page layout. Entities can be updated by incoming host commands and their associated functions can run all the time or only when the entity or it's page is visible. When a text is numeric, it can be compared, incremented or decremented or form part of an equation using the CALC command. Buffers or variables can be created for interfaces, on-board memory, the SD Card, timers, counters and text. Hex code can be included in text variables when prefixed by \\.
When creating your page structures and functions in a file, // prefixes user comments.

Uploading your Menu Structure, Functions and Images
Data received from interfaces or flash memory is processed and stored in RAM libraries for high speed access to create or refresh pages and entities. Every entity has a text name for easy reference by future update commands.
In a similar way to a PC, your software could be permanently retained on an SD card and auto loaded at Power On or saved in internal flash by transferring it from an SD card or uploading it via an interface port. SD cards of 1G size and SDHC cards of 4G, 8G, 16G and 32G size are supported. 2G SD cards are not supported.

If an SD Card is used, the module will look for a file called 'TU480A.MNU' which will reference all other menu or image files. This may be your only menu file with all functions included. It would have a header similar to the example below to copy other files on the SD card to the internal flash memory. See the 'example projects' section

```
RESET(LIBRARY); FPROG;
LIB(BACKIMAGE,"SDHC/backmain.bmp");        //load background picture into the onboard flash library
LIB(STARTIMAGE,"SDHC/startbut.bmp");       //load start button into the onboard flash library
…….. FEND;
```

From Q1,2011, entities can be changed via the user interfaces by direct reference to there name or style
Examples:

```
homepage.back="BLUE";          change the background colour of the page called homepage to blue
rs2.set="96e";                 change the rs232 baud rate to 9600 baud with even parity
StatusText="Visual Error";;    change the text area called StausText to show Visual Error
GenText.font="40X56Kata"       change font size of all text using style GenText
```

We hope you find the 'getting started' and online examples suitable for understanding the functional techniques and rapid implementation in your application. Please do not hesitate to contact our tech team by email for assistance. tech@noritake-itron.com

System Hardware Setup Parameters and Development Status
This product has been released to a limited market in Europe with 35 customers evaluating product prior to full release on 16th Sep 2010.
This page identifies the current and expected operating status of interfaces with release dates which are subject to revision.
The introduction of interface protocols (Modbus RTU) will take place in late November 2010.
The parameters for an interface are defined using the command setup(Name) {....}.

Parameters	Description	Status	View
system	set up main display system		SYS
bled = 0 - 100;	set LED backlight 0=OFF, 100=full ON or 1-99	OK - 100 levels on v4+ PCB only	
wdog = 0, 100, 500 or 1000;	set watchdog time to OFF, 100ms, 500ms or 1 second	OK	
encode = s , w, m;	s= single byte ASCII, w=2 byte UNI, m= UTF8	OK	
test=show/hideTouchAreas	show or hide outline view of touch areas on screen.	OK	
rotate= 0 or 180;	set screen orientation with respect to PCB.	OK	

asynchronous interfaces	set up rs2, rs4, as1, as2, dbg		RS2
set="96NC"	quick set up combination	OK	ASY
baud = num;	num = 110 to 115200.	OK	
data = num;	num = 5, 6, 7, 8	OK	
stop = num;	num = 1, 15, 2 15 is 1.5 bits	OK	
parity = ch;	parity = Odd, Even, None, Mark, Space	OK	
rxi= Y or C or N;	set receive buffer interface active	OK	
proc = ";" or \\0D or other	process on receive string terminator	OK	
procDel = Y or N	delete or keep termination character.	OK	
rxb= num;	set size of receive buffer in bytes.	OK	
txi= Y or E or N;	set transmit buffer interface	OK	
txb= num;	set size of transmit buffer in bytes.	OK	
encode = s , w, m;	s= single byte ASCII, w=2 byte UNI, m= UTF8	OK	
flow = N , H, S;	flow control - none, hardware, software XON XOFF	OK except XON/OFF plan 2nd Dec	

spi interface	set up spi , tsync, rsync	Receive OK, Transmit 26th Nov	SPI
set = "MR100";	quick set up combination	OK for receive	
active= M or S or N;	set as Master, Slave or None	Slave Only	
edge= R or F;	uses Rising or Falling clock edge	OK	
speed = 100;	set transmit speed in master mode		
rxi= Y or C or N;	set receive buffer interface as active	OK	
proc=";" or \\0D or other	process on receive string terminator	OK	
procDel = Y or N	delete or keep termination character.	OK	
encode = s , w, m;	s= single byte ASCII, w=2 byte UNI, m= UTF8	OK	
rxb= num;	set size of receive buffer in bytes	OK	
rxo= M or L;	set receive data order	OK	
rxf = N , H;	use none or hardware MB		
rxs = N , Y;	use select input \RSS.	OK	
txi= Y or E or N;	set transmit buffer interface as active		
end= "nn"	byte returned when no data left in buffer		
txb= num;	set size of transmit buffer in bytes.		
txo= M or L;	set transmit data order		
txf = N , H;	none or hardware HB in Master mode		
txs = N , Y;	use select output \TSS in master mode		

i2c interface	set up i2c		I2C
set = "S7E";	quick set up of I2C - Slave and Address	OK	
addr= "nn"	address pair where nn write, nn+1 read	OK	
end= "nn"	byte returned when no data left in buffer	OK	
active= M or S or N;	set as Master Slave or None	OK	
speed = 100;	set transmit speed value in master mode	OK	
rxi= Y or C or N;	set receive buffer interface as active with command	OK	
proc = ";" or \\0D or other	process on receive string terminator	OK	
procDel = Y or N	delete or keep termination character.	OK	
encode = s , w, m;	s= single byte ASCII, w=2 byte UNI, m= UTF8	OK	
rxb= num;	set size of receive buffer in bytes	OK	
txi= Y or E or N;	set transmit buffer interface as active with echo	OK	
txb= num;	set size of transmit buffer in bytes.	OK	

key i/o interfaces	K23 is the highest order bit and K0 the lowest		KEY
active	ihigh is active "\\000000" >"\\FFFFFF"	OK	
inp	high is input, low output"\\000000" >"\\FFFFFF"	OK	
trig	high is trigger interrupt	OK	
edge	high is rising edge, low is falling edge	OK	
keyb	high is scanned keyboard connectiony	OK	

pwm controller	pwm1, pwm2 - 160Hz to 1MHz	OK	PWM
active=N,1,2,12;	set pwm activity None, pwem1, pwm2, pwm1 and 2		
polln=H or L;	poll1, poll2 is High or Low on first phase		
cyclen=hhh;	value in microseconds for cycle1, cycle2		
dutyn=hh;	value as a percentage of High period		
delay=nnn;	delay in microseconds between pwm1 and pwm2		

analogue converters	adc1, adc2 are processed at 1000 samples per second	OK	ADC
active=N,1,2,12;	set none, ADC1, ADC2 or both		
calib1=function name;	set user function to use for calibrate/scale ADC1		
calib2=function name;	set user function to use for calibrate/scale ADC2		
avg1= 1-16;	number of samples taken and averaged for ADC1		
avg2= 1-16;	number of samples taken and averaged for ADC2		

buzz = buzzer output	Use LOAD(BUZZ,var) var=ON,OFF, or time in millisecs	OK	BUZ

Real Time Clock and Date	Specify real time clock	OK	RTC
active	enable = Y or disable = N		
format	various characters specify the date and time format		
RTCSECS	numeric variable containing seconds (0-59)		
RTCMINS	numeric variable containing minutes (0-59)		
RTCHOURS	numeric variable containing hours (0-23)		
RTCDAYS	numeric variable containing days (1-31)		
RTCMONTHS	numeric variable containing month (1-12)		
RTCYEARS	numeric variable containing year (1900-2099)		
	use LOAD(var,RTC) and LOAD(RTC,varY,":",varM,":",etc		

RUN TIME COUNTER	Predefined variables which can be set and tested. The runtime counter is continually counting. It is independent of the real time clock.	OK	RUN
	Reset(RUNTIME); resets counter to zero LOAD(CNTSECS,23); set value of seconds		
CNTMILLI	increments every millisecond 0-999		
CNTSECS	increments every second 0-59		
CNTMINS	increments every minute 0-59		
CNTHOURS	increments every hour 0-23		
CNTDAYS	increments every day 0-n		

other interface references			
internal eeprom	parameter storage using extended variables VarE	OK	VAR
sdhc = SD Card (1G or 4G+)	FAT32 - 8 character file names, no directory. Not 2G	Read OK. Write 26th Nov	SD
internal NAND flash	Proprietary structure	active v33 for firmware	
usbcom = usb com port		26th Nov	COM
usbmsd = mass storage		26th Nov	MSD
CAN adaptor - 1MHz	adaptor connects to CN3	OK	CAN
ac97= audio buffer	adaptor connects to CN4	TBD	I2S

System, RTC and Counter Setup

System

Set up the system. These parameters can be set at initialisation or at any time during operation by specifying the parameter to be changed. Example: setup(system){ bled=50; }

bled = 0 - 100

set backlight to OFF=0 or ON=100 (1-99 brightness levels available v4 PCB, v32 firmware)

wdog = 0, 100, 500, 1000

set the watchdog time out period in milliseconds.

rotate = 0, 180

set the rotation of the screen with respect to PCB

test=hide/showTouchAreas

hide or show touch areas during product development

calibrate = y or n

initialise the internal touch screen calibration screen. This automatically returns to the previous page on completion. If it is necessary to abort then send setup(system) {calibrate=n};

encode = s, w, m

menu text strings can contain single byte ASCII (s), 2 bytes for UNIcode (w) or multibyte for UTF8 (m)

Example system set up

```
setup(system)
  {
  bled=100;
  wdog=100;
  rotate=0;
  test=showTouchAreas;
  encode=s;  //ASCII handling with extended unicode/utf8 in occasional strings
  }
```

operational

Real Time Clock RTC

The real time clock requires a battery to be fitted to the rear of the module or a 3VDC supply applied via a connector fitted to the rear of the PCB. The default format is 14 Sep 2010 09:50:06 which can be modified to suit the application which is achieved by loading the RTC into a variable having the required format. Another method is to use predefined variables of individual RTC values.

SET RTC

The RTC is set using 24 hour time with LOAD(RTC, "YYYY:MM:DD:hh:mm:ss");
with fixed format where:
- YYYY is year 1900-2099
- MM is month 01-12
- DD is day of month 01-31
- hh is hours 00-23
- mm is minutes 00-59
- ss is seconds 00-59

Use vars to setup the time in a user page

```
VAR(years,2010,U16);
VAR(months,11,U8);
VAR(days,2,U8);
VAR(hours,10,U8);
VAR(mins,30,U8);
```

User changes the vars via buttons then a SAVE button would load the RTC

```
LOAD(RTC,years,":",months,":",days,":",hours,":",mins,":00");
```

READ RTC

You can LOAD the RTC into a variable where the format is specified in a style as follows:

```
STYLE( myRtcStyle, Data )
  {
  type = text;          // Setup a text variable
  length = 64;          // with max length of 64 chars
  format = "jS F Y g:ia"; // RTC format string
  }


VAR( RtcVar, "", myRtcStyle );  // Create a var to store formatted string
LOAD( RtcVar, RTC );       // Grab the formatted RTC time and date
TEXT( Txt1, RtcVar );;     // Show the formatted time on display in Txt1 and refresh screen
LOAD( RS2, RtcVar );      // Send formatted time on RS232 port
```

The RTC date/time can be displayed as a formatted string using special characters

> Day:

d	Day of month with leading zeros	01-31
j	Day of month without leading zeros	1-31
S	Ordinal suffix for day of month	st, nd, rd, th

> Month:

F	Full textual representation of month	January-December
m	Numeric representation of month with leading zeros	01-12
M	Short textual representation of month, three letters	Jan-Dec
n	Numeric representation of month without leading zeros	1-12

> Year:

Y	Full numeric representation of year, 4 digits	1900-2099
y	Two digit representation of year	00-99

> Time:

a	Lowercase Ante meridiem and Post meridiem	am, pm
A	Uppercase Ante meridiem and Post meridiem	AM, PM
g	12-hour format of hour without leading zeros	1-12
G	24-hour format of hour without leading zeros	0-23
h	12-hour format of hour with leading zeros	01-12
H	24-hour format of hour with leading zeros	00-23
i	Minutes with leading zeros	00-59
s	Seconds with leading zeros	00-59

> other characters not in list will be shown as is

Format examples:
"d M Y H:i:s" will display as: 14 Sep 2010 09:50:06 (default format)
"d/m/y" will display as: 14/09/10
"jS F Y g:ia" will display as: 14th September 2010 9:50am

Predefined variables below can be read, but not set.

RTCSECS	numeric variable containing seconds (0-59) which can be tested or loaded into a text.
RTCMINS	numeric variable containing minutes (0-59) which can be tested or loaded into a text.
RTCHOURS	numeric variable containing hours (0-23) which can be tested or loaded into a text.
RTCDAYS	numeric variable containing days (1-31) which can be tested or loaded into a text.
RTCMONTHS	numeric variable containing month (1-12) which can be tested or loaded into a text.
RTCYEARS	numeric variable containing year (1900-2099) which can be tested or loaded into a text.

operational

Runtime Counter

The RUNTIME counter uses pre-define variables which can be set and tested for values
The command Reset(RUNTIME) sets all vales to zero and starts the timer.
This runtime counter is independent of the real time clock and runs continually so no setup is required.

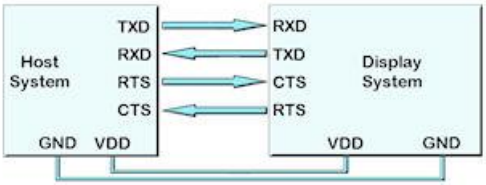
CNTMILLI	Increments every millisecond 0-999
CNTSECS	Increments every second 0-59
CNTMINS	Increments every minute 0-59
CNTHOURS	Increments every hour 0-23
CNTDAYS	Increments every 24 hours

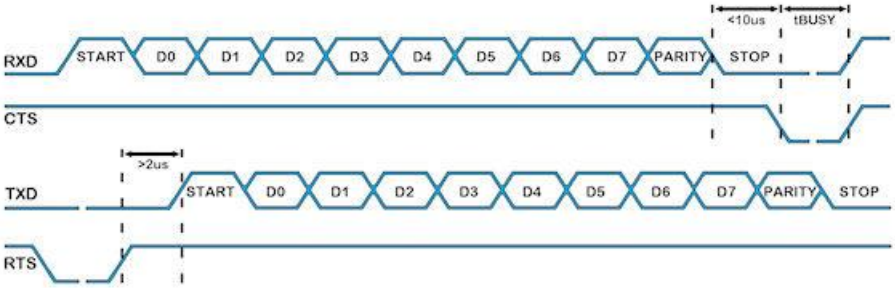
operational

RS232 Interface - RS2

The asynchronous communication speed and parity can be set with the setup command. The hardware lines RTS-CTS and DTR-DSR enable communication between host and module and are selected by jumpers on the back of the module. Only one pair can be selected at any one time. (RTS-CTS or DTR-DSR).

If RS485 is available on the module (suffix -K611xxx) then only RTS-CTS can be used.





rs232 set up parameters

set="96NC"	quick set up combination "48, 96, 192, 384, 768, 1150 with parity N, O, E and Command option".
or	
baud = num;	num = 110 to 115200. Any value can be set to allow trimming for deviating clocks i.e. 38450
data = num;	num = 5, 6, 7, 8
stop = num;	num = 1, 15, 2 - note 15 is 1.5 bits
parity = ch;	h = first letter of Odd, Even, None, Mark, Space
rxi= Y or C or N;	set receive interface as active (Y), a command processing source (C) or disable (N). Default = N
proc = ";" or other	process on receive terminator. See below
procDel = Y or N	remove or keep the termination character(s) before processing
rxb= num;	set size of receive buffer in bytes. Default = 8192 bytes
txi= Y or E or N;	set transmit interface as active (Y), to echo command processing (E) or disable (N)
txb= num;	set size of transmit buffer in bytes. Default = 8192 bytes
encode = s,w,m (r)	set s=ASCII, w=UNICODE, m=UTF8 or use sr, wr and mr specifying raw data bytes.
flow = N , H, S;	none, hardware RTS/CTS or DTR/DSR, software XON XOFF

Serial Port Interrupt Characters
Serial Port termination characters can now be specified to generate an interrupt.
Changes applied to RS2, RS4, AS1, AS2, DBG, I2C, SPI ports
The proc parameter is used in the port setup to define the termination character(s).

```
proc = all;          <- trigger on all received characters
proc = CRLF;         <- trigger on a CR followed by LF (0Dh 0A)
proc = CR;           <- trigger on CR (0Dh) ...in command mode rxi=C this is fixed
proc = LF;           <- trigger on LF (0Ah)
proc = NUL;          <- trigger on NUL (00h)
proc = \\xx;         <- trigger on xxh
proc = "ABCD";       <- string in format defined by SYSTEM encode param
proc = "\\xx\\xx";   <- string in format defined by SYSTEM encode param
```

When sending commands (rxi=C) to the module, processing only occurs when \\0D or 0D hex is received.
Example: TEXT(MyText,"Hello World");;\\0D

Example set up

```
setup(rs2)
  {
  baud=38400;
  data=8;
  parity=N;
  rxi=C;
  txi=Y;
  encode=s;
  }
```

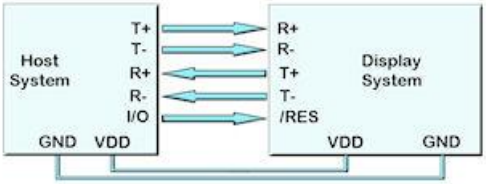
Example usage:

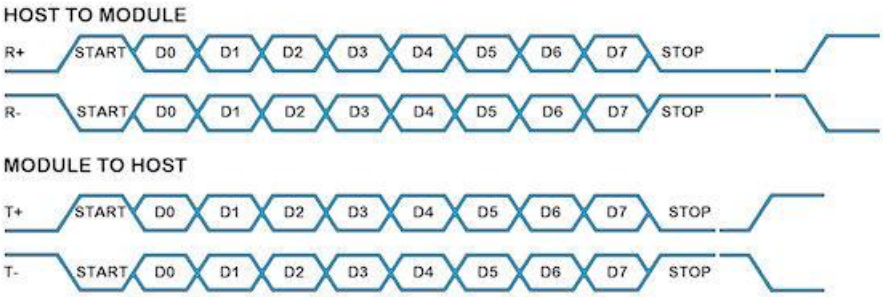
```
PAGE( PageName, PageStyle)
  {
  ....
  ....
  INT( SerRxInt, RS2RXC, SerRxEvent ); //Used when rxi=Y
  }

 FUNC( SerRxEvent )
  {
  LOAD( Var, RS2 ); // Must read RS2 to clear interrupt
  LOAD( RS4, Var);  //send out of RS485 interface.
  TEXT ( RecvTxt, Var);; //show received ASCII data on screen
                         //  and refresh
  }
```

Active v22 except XON/OFF flow control Oct 26th

RS485 Interface - RS4

RS485 is available on the module (suffix -K611xxx)
The asynchronous communication speed and parity can be set with the setup command.





rs485 set up parameters

set="96NC"	quick set up combination "48, 96, 192, 384, 768, 1150 with parity N, O, E and Command option".
or	
baud = num;	num = 110 to 115200. Any value can be set to allow trimming for deviating clocks i.e. 38450
data = num;	num = 5, 6, 7, 8
stop = num;	num = 1, 15, 2 - note 15 is 1.5 bits
parity = ch;	h = first letter of Odd, Even, None, Mark, Space
rxi= Y or C or N;	set receive interface as active (Y), a command processing source (C) or disable (N). Default = N
proc = ";" or other	process on receive termination character(s). See below
procDel = Y or N	remove or keep the termination character(s) before processing
rxb= num;	set size of receive buffer in bytes. Default = 8192 bytes
txi= Y or E or N;	set transmit interface as active (Y), to echo command processing (E) or disable (N)
txb= num;	set size of transmit buffer in bytes. Default = 8192 bytes
encode = s,w,m (r)	set s=ASCII, w=UNICODE, m=UTF8 or use sr, wr and mr specifying raw data bytes.
flow = n,s	set n=none, s=software XON,XOFF

Serial Port Interrupt Characters
Serial Port termination characters can now be specified to generate an interrupt.
Changes applied to RS2, RS4, AS1, AS2, DBG, I2C, SPI ports
The proc parameter is used in the port setup to define the termination character(s).

```
proc = all;          <- trigger on all received characters
proc = CRLF;         <- trigger on a CR followed by LF (0Dh 0A)
proc = CR;           <- trigger on CR (0Dh)  ...in command mode rxi=C this is fixed
proc = LF;           <- trigger on LF (0Ah)
proc = NUL;          <- trigger on NUL (00h)
proc = \\xx;         <- trigger on xxh
proc = "ABCD";       <- string in format defined by SYSTEM encode param
proc = "\\xx\\xx";   <- string in format defined by SYSTEM encode param
```

When sending commands (rxi=C) to the module, processing only occurs when \\0D or 0D hex is received.
Example: TEXT(MyText,"Hello World");;\\0D

Example set up

```
setup(rs4)
 {
 baud=38400;
 data=8;
 parity=N;
 rxi=C;
 txi=Y;
 encode=s;
 }
```

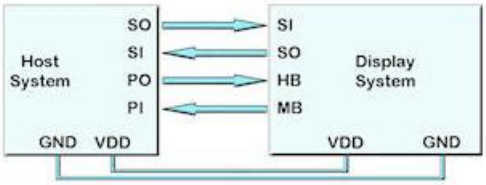
Example usage:

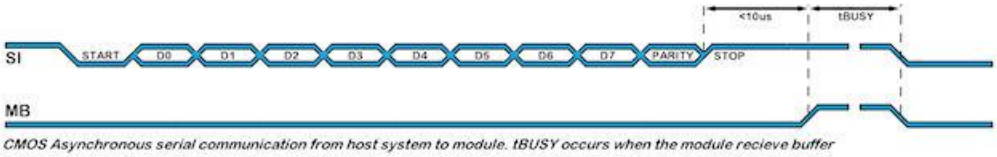
```
PAGE( PageName, PageStyle)
 {
 ....
 ....
 INT( SerRxInt, RS4RXC, SerRxEvent ); //Used when rxi=Y
 }

FUNC( SerRxEvent )
 {
 LOAD( Var, RS4 ); // Must read RS4 to clear interrupt
 LOAD( RS2, Var);  //send out of RS232 interface.
 TEXT ( RecvTxt, Var);; //show received ASCII data on screen
                        //  and refresh
 }
```

Active v22 except flow control 28th Oct

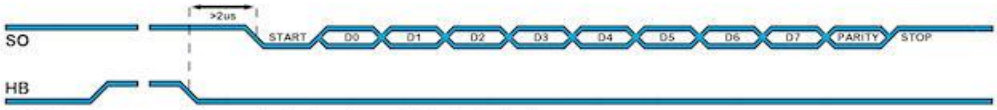
CMOS Asynchronous Interfaces - AS1, AS2, DBG (3v3 level)

The asynchronous communication speed and parity can be set with the setup commands. The host busy line (HB) stops the module from sending data to the host. The use of the HB and MB busy lines are optional, and can be connected together if not required.





CMOS Asynchronous serial communication from host system to module. tBUSY occurs when the module recieve buffer



CMOS Asynchronous serial communication from the module to the host system.

as1, as2, dbg set up parameters

set="96NC"	quick set up combination "48, 96, 192, 384, 768, 1150 with parity N, O, E and Command option".
baud = num;	num = 110 to 115200. Any value can be set to allow trimming for deviating clocks i.e. 38450
data = num;	num = 5, 6, 7, 8
stop = num;	num = 1, 15, 2 - note 15 is 1.5 bits
parity = ch;	h = first letter of Odd, Even, None, Mark, Space
rxi= Y or C or N;	set receive buffer interface as active (Y), a command processing source (C) or disable (N). Default = N
proc = ";" or other	process on receive termination character(s). See below
procDel = Y or N	remove or keep the termination character(s) before processing
rxb= num;	set size of receive buffer in bytes. Default = 8192 bytes
txi= Y or E or N;	set transmit buffer interface as active (Y), to echo command processing (E) or disable (N)
txb= num;	set size of transmit buffer in bytes. Default = 8192 bytes
encode = s,w,m (r)	set s=ASCII, w=UNICODE, m=UTF8 or use sr, wr and mr specifying raw data bytes.
flow = N , H, S;	none, hardware RTS/CTS or DTR/DSR, software XON XOFF

Serial Port Interrupt Characters
Serial Port termination characters can now be specified to generate an interrupt.
Changes applied to RS2, RS4, AS1, AS2, DBG, I2C, SPI ports
The proc parameter is used in the port setup to define the termination character(s).

```
proc = all;          <- trigger on all received characters
proc = CRLF;         <- trigger on a CR followed by LF (0Dh 0A)
proc = CR;           <- trigger on CR (0Dh) ...in command mode rxi=C this is fixed
proc = LF;           <- trigger on LF (0Ah)
proc = NUL;          <- trigger on NUL (00h)
proc = \\xx;         <- trigger on xxh
proc = "ABCD";       <- string in format defined by SYSTEM encode param
proc = "\\xx\\xx";   <- string in format defined by SYSTEM encode param
```

When sending commands (rxi=C) to the module, processing only occurs when \\0D or 0D hex is received.
Example: TEXT(MyText,"Hello World");;\\0D

Example set up

Example

```
setup(as1)
 {
 baud=38400;
 data=8;
 parity=N;
 rxi=C;
 txi=Y;
 encode=s;
 }

PAGE( PageName, PageStyle)
 {
 ....
 ....
 INT( ASerRxInt, AS1RXC, SerRxEvent ); //Used when rxi=Y
 }

FUNC( SerRxEvent )
 {
 LOAD( Var, AS1 ); // Must read AS1 to clear interrupt
 LOAD( RS4, Var);  //send out of RS485 interface.
 TEXT ( RecvTxt, Var);; //show received ASCII data on screen
                        //  and refresh
 }
```

CANBUS Adaptor

When attaching a CANBUS adaptor type EMCBK33 to CN3 using a 10 way IDC cable, the connector is fitted to the backside of the module and the following set up is required to match the default settings in the adaptor.

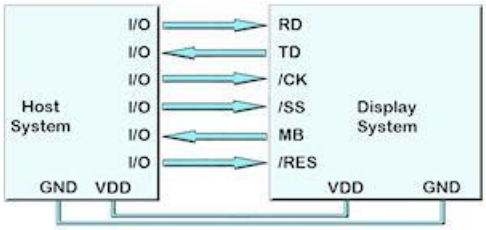
```
setup(as1)
 {
  baud=38400;
  data=8;
  stop=1;
  parity=none;
  rxi=C;
  encode=sr;
  flow=H;
 }
```

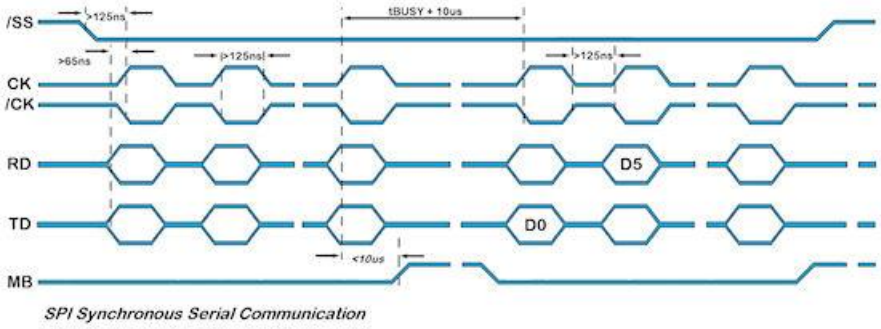
The default receive address for the adaptor is ID=155h with 11bit or 29bitID packets accepted (2.0a or 2.0b spec)
All bytes are received on AS1 with 1 to 8 bytes of data.
The transmit ID is also 155H. with data sent via AS1 with data length of 1.

Active v0.22 except XON/OFF flow control plan TBD

SPI and I2C Interfacing

SPI Interface - SPI (3v3 level)
With synchronous communications enabled, data can be clocked into the TFT module using the rising or falling edge of SCK. This is selectable by the setup command which also sets other parameters. By default, data is clocked in on the rising edge with the most significant bit sent first.
The /SS pin can be used as an enable pin if other devices are connected to the serial line and also allows byte synchronization. If MB is set high, the input buffer is full or disabled. A dummy/end byte for reading and buffer status can be set by the user.





SPI Synchronous Serial Communication

spi - set up parameters

set = "MR100"; or "SRC";	quick set up as Master/Slave, edge R/F, Command and speed 20-1000
or	
active= M or S or N;	set as Master, Slave or None for both transmit and receive. Default = N
edge= R or F;	uses Rising or Falling clock edge. Default = R
speed = 100;	set transmit speed value in kilobits/sec from 20 to 1000 for master mode. Default = 100
rxi= Y or C or N;	set receive buffer interface as active (Y), a command processing source (C) or disable (N). Default = N
proc = ";" or other	process on receive termination character(s). See below.
procDel = Y or N	remove or keep the termination character(s) before processing
encode = s,w,m (r)	set s=ASCII, w=UNICODE, m=UTF8 or use sr, wr and mr specifying raw data bytes.
rxb= num;	set size of receive buffer in bytes. Default = 8192 bytes
rxo= M or L;	set receive data order as most significant bit (M) or least significant bit (L). Default = M
rxf = N , H;	use none or hardware MB to signify receive buffer full. Default = N
rxs = N , Y;	use select input \RSS. Default = N
txi= Y or E or N;	set transmit buffer interface as active (Y), to echo command processing (E) or disable (N)
end= "nn"	byte returned when no data left in display's spi transmit buffer and as a dummy byte to send if required.
txb= num;	set size of transmit buffer in bytes. Default = 8192 bytes
txo= M or L;	set transmit data order as most significant bit (M) or least significant bit (L). Default = M
txf = N , H;	none or hardware HB used to signify halt transmit in master mode. Default = N
txs = N , Y;	use select output \TSS in master mode. Default = N

Serial Port Interrupt Characters
Serial Port termination characters can now be specified to generate an interrupt.
Changes applied to RS2, RS4, AS1, AS2, DBG, I2C, SPI ports
The proc parameter is used in the port setup to define the termination character(s).

```
proc = all;          <- trigger on all received characters
proc = CRLF;         <- trigger on a CR followed by LF (0Dh 0A)
proc = CR;           <- trigger on CR (0Dh)  ...in command mode rxi=C this is fixed
proc = LF;           <- trigger on LF (0Ah)
proc = NUL;          <- trigger on NUL (00h)
proc = \\xx;         <- trigger on xxh
proc = "ABCD";       <- string in format defined by SYSTEM encode param
proc = "\\xx\\xx";   <- string in format defined by SYSTEM encode param
```

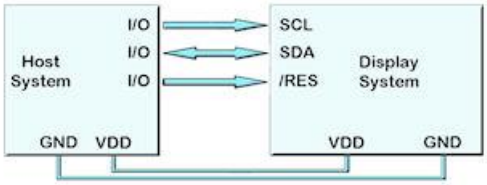
When sending commands (rxi=C) to the module, processing only occurs when \\0D or 0D hex is received.
Example: TEXT(MyText,"Hello World");;\\0D

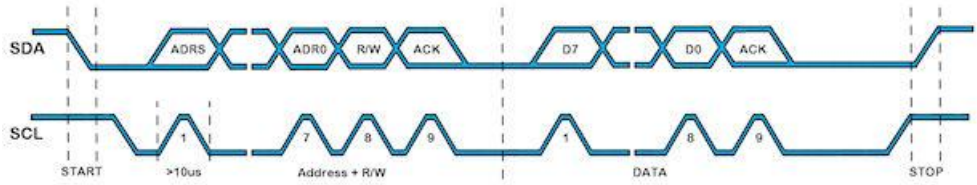
Example spi set up

```
setup(spi)
  {
  active=S;
  edge=R;
  rxi=C;
  txi=Y;
  encode=s;
  }
```

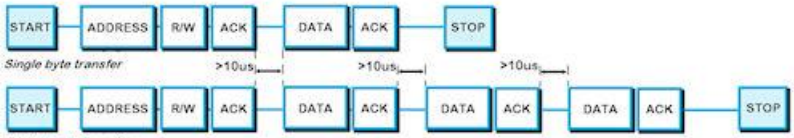
SPI receive active v32. Transmit plan 26th Oct

TWI / I2C Interface - I2C (3v3 level)
The I2C interface operates as a slave either in 'slave receive' or 'slave transmit' mode with a user defined address set in the I2C setup. Receive (i2c.rxb) and transmit (i2c.txb) buffers of 8192 bytes are created which can be cleared and set by the command processor. An end byte indicating empty buffer can be set.





Typical I2C transmission



An overview of how TWI / I2C communicates
A START condition is signalled by driving SDA low while SCL is high. A STOP condition is signalled by driving SDA high while SCL is high. After a START condition is detected followed by address + R/W bit, the command / data bytes are stored in a 8192 byte buffer. The module will pull SDA low during the 9thclock cycle of a data transfer to acknowledge the receipt of a byte. Additional data may be sent providing the host receives an Ack. If the host has not detected an Ack the data transfer must be started again by providing a STOP and START condition and address + R/W bit low. When reading an I2C packet must be sent with address+1 read the data bytes from the I2C transmit buffer.

twi / i2c set up parameters

set = "C7E";	quick set up of I2C - Slave with Command and Address
or	
addr= "nn"	address pair where nn for write and nn+1 for read with range 02 to FE.
end= "nn"	byte returned when no data left in display's i2c transmit buffer
active= M or S or N	set as Master (M) or Slave (S) or disabled (N). Default = N
speed = 100;	set transmit speed value in kilobits/sec from 20 to 400 for master mode. Default = 100
rxi= Y or C or N;	set receive buffer interface as active (Y), a command processing source (C) or disable (N). Default = N
proc = ";" or other	process on receive termination character(s)
procDel = Y or N	remove or keep the termination character(s) before processing
encode= s,w,m	s= ASCII single byte, w=UNICODE 2 byte, m=UTF8 multibyte
rxb= num;	set size of receive buffer in bytes. Default = 8192 bytes
txi= Y or E or N;	set transmit buffer interface as active (Y), to echo command processing (E) or disable (N)
txb= num;	set size of transmit buffer in bytes. Default = 8192 bytes

Serial Port Interrupt Characters
Serial Port termination characters can now be specified to generate an interrupt.
Changes applied to RS2, RS4, AS1, AS2, DBG, I2C, SPI ports
The proc parameter is used in the port setup to define the termination character(s).

```
proc = all;          <- trigger on all received characters
proc = CRLF;         <- trigger on a CR followed by LF (0Dh 0A)
proc = CR;           <- trigger on CR (0Dh) .. when sending commands (rxi=C), this is fixed
proc = LF;           <- trigger on LF (0Ah)
proc = NUL;          <- trigger on NUL (00h)
proc = \\xx;         <- trigger on xxh
proc = "ABCD";       <- string in format defined by SYSTEM encode param
proc = "\\xx\\xx";   <- string in format defined by SYSTEM encode param
```

When sending commands (rxi=C) to the module, processing only occurs when \\0D or 0D hex is received.
Example: TEXT(MyText,"Hello World");\\0D

Example set up for TWI/I2C

```
setup(i2c)
  {
  addr=3E;
  end="\\00";
  active=S;
  rxi=C;
  encode=s;
  }
```

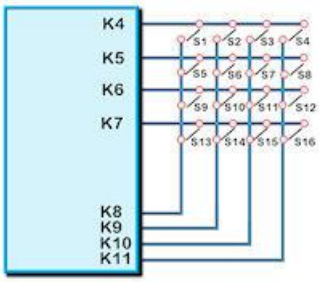
Active v34

Keyboard and I/O Interfacing

Keyboard Control
24 I/O lines (K0-K23) can be configured to scan a key matrix with up to 144 keys configured using the setup commands for I/O control. When a key is pressed, a function can be initiated using a key command.

Dual key presses are supported to enable SHIFT functionality.

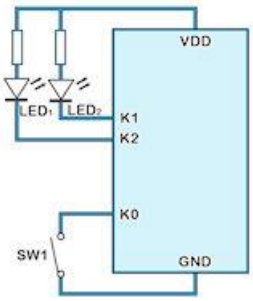
No diodes are required in the key matrix for dual key operation making it ideal for low cost membrane keyboards.



I/O Control
The module contains simple Input and Output functions for the 24 I/O lines (K0-K23).All inputs include an optional pull-up resistor ~50K-120K in value. The outputs can source ~1mA and sink ~3mA.
Certain I/O have expanded functions for customization.

NOTE: The ports K0 to K15 connect directly to the CPU without ESD protection. K16 to K23 have series 100R resistors and 10pF capacitors to GND.

K23 is the highest order bit and K0 the lowest.



keyio — **24 bits of user i/o and keyboard -K23 to K0** **operatonal**

active = hex value;	high is active "\\000000" >"\\FFFFFF", default is inactive
inp = hex value;	high is input, low is output "\\000000" >"\\FFFFFF"
trig = hex value;	high is trigger interrupt "\\000000" >"\\FFFFFF" as defined by edge
edge = hex value;	high is rising edge, low is falling edge "\\000000" >"\\FFFFFF"
keyb = hex value;	high is scanned keyboard connection "\\000000">"\\FFFFFF"

Single bit variables can be set and tested K00, K01, K02...K23 once enabled
8 bit variables can be set and tested KA, KB, KC, KD, KE once enabled
KA = K07,K06,K05,K04,K03,K02,K01,K00
KB = K15,K14,K13,K12,K11,K10,K09,K08
KC = K14,K12,K10,K08,K06,K04,K02,K00
KD = K15,K13,K11,K09,K07,K05,K03,K01
KE = K23,K22,K21,K20,K19,K18,K17,K16

example setup

```
setup(keyio)
  {
  active=\\000FFF;                  //K00-K11 as active
  inp=\\00000C;                     //K00,K01 as output and K2,K3 as input
  keyb=\\000FF0;                    //K04-K11 as keyboard
  }
```

example usage to set

LOAD(K01,1); set K1 to high
LOAD(K02,0); set K2 to low
LOAD(KA,\\02); set K0,K2-K7 low and K1 high

LOAD(myVar,K01) load port into user variable
LOAD(myVar,KA) load 8bit port into user variable

example usage with interrupt

```
SETUP(keyio)
  {
  active=\\000001;  // set K00 to be active
  inp=\\000001;     // set K00 as input
  trig=\\000001;    // enable trigger interrupt on K00
  edge=\\000000;  // set to trigger in falling edge
  }

PAGE(mypage,pagestyle)
  {
  //set up entities or keys on page
  INT(myInt,K00,myEvent);  // setup interrupt to call 'myEvent' on every K00 event
  // rest of page
  }

FUNC(myEvent)  // This function is called each time a falling edge is detected on K00
  {
    // some actions
  }
```

The current firmware requires the K parameter to be 3 characters in length

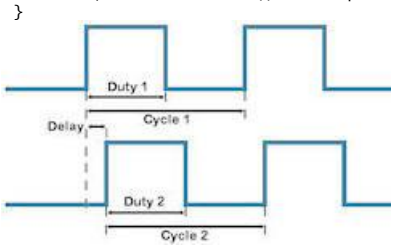
NOTE v4 PCB cannot use K2,K3,K4 when RS485 fitted. To be changed in v5 PCB.

pwm controller **operational**

active = N,1,2,12	use 12 to synchronize PWM 1 and 2. N=none
pol1 = (H)igh or (L)ow	polarity = High or Low on first phase of PWM1
pol2 = (H)igh or (L)ow	polarity = High or Low on first phase of PWM2
cycle1 = "nnnnnn"	cycle time in microseconds of PWM1. Range 160Hz to 1MHz
cycle2 = "nnnnnn"	cycle time in microseconds of PWM2. Range 160Hz to 1MHz
duty1= "hh"	value of first phase as a percentage for PWM1 = 1-99
duty2= "hh"	value of first phase as a percentage for PWM2 = 1-99
delay= "hhh"	delay between first phase of PWM1 and first phase of PWM2 in microseconds

example setup

```
setup(pwm)
  {
  pol1=H;pol2=H;                //
  cycle1="100";cycle2="200";   //set frequency
  duty1=33;duty2=66;            // set duty cycle of each
  delay=50;                    // pwm2 starts 50 microseconds after pwm1
  active=12;                    //set both pwm1 and pwm2 as active
  }
```



adc- analogueto digital converters **operational**

The input voltage range is 0V to 3VDC max. Ref voltage is filtered from 3.3VDC.
ADC1 and ADC2 are sampled each 1ms using 10bit successive approximation.
If the result is copied to an 8 bit variable, the high order bits are used.

active = N,1,2,12	set none, ADC1, ADC2 or both
calib1 = function name	set user function to use for calibration/scaling of ADC1
calib2 = function name	set user function to use for calibration/scaling of ADC2
avg1 = 1-16	number of samples read and then averaged for ADC1
avg2 = 1-16	number of samples read and then averaged for ADC2

example setup

```
setup( adc )
  {
  active=12;
  calib1=ADC1LOG;
  calib2=ADC2LIN;
  }
```

example usage

```
LOAD (VOLTS, ADC2, " VDC");;    // Load text with new value and refresh screen;

//Control graph on screen ~ 400x200 pixels
VAR(PixXVal,5,U16);
VAR(PixelX,"Pixel0",TXT);
 ... create calib function to convert ADC value to range 5-205.
 ...then draw 400 pixels on page named Pixel5 to Pixel404 then

LOOP(GraphLoop,Forever) {
   POSN ( PixXVal, ADC1, PixelX);;       // Move pixel on graph to new y value and refresh screen
   IF(PixXVal=404?[LOAD(PixXVal,5);]:[CALC(PixXVal,PixXVal,1,"+");]); //Move to next X Pixel
   LOAD(PixelX, "Pixel",PixXVal);       // Use  to change name of pixel.
   }                              // will run at 20 samples per second with v34 firmware
```

Command Overview and Development Status

This product has been released to a limited market in Europe with 35 customers evaluating product prior to full release on 16th Sep 2010.
This page identifies the current and expected operating status of commands and styles with release dates which are subject to revision.
The commands have a YELLOW background and the styles a PURPLE background.

Command, Style, Variable	Description	Status	View
FPROG.....FEND	Store SDHC menu and image files in onboard flash	Plan	FPROG
LIB(Name,Source)	Load picture or font into library	OK	LIB
INC(FileName)	Include the contents of another menu, style or setup file	OK	INC
RUN(Func)	Run a function or user code	OK except custom code	RUN
RESET(Name)	Clear eeprom variable, delete list, library or reset system	OK except library	RESET
LOAD(Name,N2,N3,N..)	Multi function copy page, variable N2--N.. to Name.	OK except copy page	LOAD
SHOW(Name)	Show a page, entity	OK	SHOW
HIDE(Name)	Hide a page, entity	OK	HIDE
DEL(Name)	Delete a page, entity	OK	DEL
VAR(Name,Value,Style)	Create a variable of a specified type with a default value	OK	VAR
IF(Var~Var?Func1:Func2)	Evaluate condition and do func1 if true, func2 if false	OK	IF
LOOP(Name,Var){...}	Loop for a specified number of times	OK	LOOP
INT(Name,Buffer,Function)	If interrupt triggered do function	OK	INT
CALC(Result, Var1, Var2, Act)	Quick calculation and text manipulation	OK except string manipulation	CALC
FUNC(Name) {...}	Declare a set of commands	OK	FUNC
STYLE(Name,Type) {...}	Predefine parameters for page entities and variables	OK	
WAIT(Time)	Wait specified milliseconds before next	OK	WAIT
;	Terminate command	OK	SEMI
;;	Refresh current page	OK	DSEMI
[cmd();cmd();....cmd;]	Enclose commands as inline function in IF, INT, KEY, RUN	OK	INLINE
POSN(X,Y,Page/Name,Style)	Position cursor or re-position named entity	OK	POSN
PAGE(Name,Style) {..}	Specify contents of page	OK except large size and rotate	PAGE
sizeX, sixeY	Specify the size of the page	OK except large size	-
posX, posY	Specify the absolute position on screen	OK	-
back	Specify the background colour of page	OK	-
image	Specify a background image for the page	OK	-

TEXT(Name,Text,Style)	Define text	OK except embed cursor and rotate	TEXT
font	The ASCII based + extended fonts	OK	-
size	Size multiplier ie 24x24 to 48x48	OK	-
col	Specify the text color.	OK	-
maxLen	Specify the maximum number per row (Max 512)	OK	-
maxRows	Specify the maximum number of rows (Max 32)	OK	-
curRel	Specify the relative placement of the text	OK	-
rotate	Specify the rotation of the text 0,90,180,270	Plan	-

DRAW(Name,X,Y,Style)	Create box, circle, line, pixel, shape	OK except line, pixel, rotate	DRAW
type	Specify the type of shape to draw	OK except diagonal line and pixel	-
col	Specify the border colour of the shape	OK	-
back	Specify the back colour of the shape	OK	-
width	Specify the border width of the shape	OK	-
sizeX,sizeY	Specify the maximum width and height	OK	-
curRel	Specify the relative placement	OK	-
rotate	Specify the rotation of the shape 0,90,180,270	Plan	-

IMG(Name,Source,Style)	Image placement and manipulation	OK with style limitations	IMG
scale	The image can be cropped to centre or fit	Plan	-
sizeX, sizeY	Specify the maximum width and height	OK	-
curRel	Specify the relative placement .	OK	-
rotate	Specify the rotation of the image 0,90,180,270	Plan	-

KEY	Designation of touch or key matrix	OK	KEY
type	Specify the source of key data - touch or external	OK	-
debounce	Specify the time delay to allow a key	OK	-
delay	Specify the time delay for auto repeat	OK	-
repeat	Specify the time delay for auto repeat	OK	-
action	Specify action point as Down or Up	OK	-
curRel	Specify the relative placement	OK	-

To update a variable from a port using the equate sign e.g. VOLTS=34.5; will be available from end Jan 2011
Until then, please use LOAD(VOLTS,34.5);

System Commands

Command	Description and Status
FPROG **.....** **FEND**	FPROG and FEND are used to program subsequent commands into internal flash memory. Use the RESET (LIBRARY) command before FPROG if the existing structure is to be replaced, otherwise the commands are appended to the existing structure. **Plan 26th Nov.**
LIB(Name,Source)	Store image, font, user font or user code file in the library. **Image and Fonts from an SD Card (Onboard Flash)** Image and Font files can be BMP and FNT formats. Since BMP format does not contain transparency information, a colour can be specified after the file name. Example LIB(myimage,"SDHC/backimg.bmp?back=\\000007"); **v0.21.** LIB(16x16fnt,"SDHC/asc16B.fnt?start=\\0020"); **v0.27** LIB(24x24fnt,"SDHC/my24font.bmp?back=\\0000FF&array=24x24&start=\\0080"); **TBD** **Image and User Font loaded from a Serial Link TBD** Where the image or font is raw byte data, the format is Transparency/Colour Bits/SizeXY Colour resolutions are 24,16,8 and 1. Data increments horizontally left to right, top to bottom. When 1, the data is stored as 8 bits in a byte horizontally with D7 received first. The array of a user font defines how many rows and columns are laid out. Source can be a path to an SDHC file or specify serial interface and format parameters. Examples LIB(myimage,"rs2/bmp?back=\\FFFFFF&col=24&size=480X272"); LIB(myfont,"spi/fnt?back=\\000000&col=8&size=256X256&array=16x16&start=\\0000"); LIB(myimage,"spi/bmp?back=\\000000&col=1&size=480X272&order=HH7"); **User Code TBD** User code is submitted in 'C' and compiled by our firmware engineers subject to quotation and agreement. The resultant file is of type .BIN. The user code can then be used with the RUN(Name) command. LIB(myprog,"sdhc/ourprog.bin"); LIB(myprog,"rs2/bin?bytes=36574"); The system does not yet recognize directory structures in the SDHC card. Please put all active files in the root. All file names are 8 characters maximum length. **.BMP is operational v17.** **User Compiled Code and User Font Array TBD**
INC(Source)	Include another menu, style or setup file in the current file. This command can be used to reference a file containing styles and commands on the SDHC card so that it's contents are included at that point in the command process. This enables modular design of the menu system. The system does not recognize directory structures in the SDHC card. Please put all active files in the root. All file names are 8 characters maximum length. **Example:** INC("sdhc/submenu.mnu") specifies the file path on the SDcard. INC(File1,File2,File3,...FileN); multiple files are possible **Operational for up to 7 levels v35**
RESET(Name)	Clear the contents of the RunTime Counter, Delete List, Library Files or do a System reset. Reset the System so that it re-boots as at power ON using RESET(SYSTEM) Clear the runtime counter with RESET(RUNTIME); Clear the EEPROM and reload defined variables RESET(EEPROM); v34 Clear the deleted entity list with RESET(DELETED); Clear the library with RESET(LIBRARY); **Reset 'Deleted' and 'Library' on 26th Nov**
;	Command separator used in menu files and data sent or received via serial interfaces. Example: RUN(HELP); WAIT("1000"); **Operational v17**
;;	Refresh the current page. Can be used for refreshing a page after a series of entity updates without knowing which page is showing. LOAD(VOLTS,"34");LOAD(AMPS,"100");; **Operational v23**

[cmd(..); cmd(...);.....cmd(..);]	The commands which require a function as a parameter ie IF, RUN, INT and KEY can have the function code embedded inside the commands by enclosing the required code in square brackets
	This allows you to reduce the number of lines of code for simple functions and where the function is unlikely to be used elsewhere.
	
	Without inline:
	KEY(keyFlr15,floor15fnc,104,84,TOUCH); //calls function floor15fnc
	
	FUNC(floor15fnc)
	{
	LOAD(vReqd,15); TEXT(txtCurFlr,"15"); RUN(fncGo);
	}
	
	With inline:
	KEY(keyFlr15, [LOAD(vReqd,15); TEXT(txtCurFlr,"15"); RUN(fncGo);],104,84,TOUCH);
	
	Operational v30

Page and Group Commands

Command	Description

PAGE(Name,Style) {.....}	Create a Page or Group of entities. Pages contain entities to be shown on the display plus functions that will run as a background task only on that page. Entities are listed so that they are layered from back to front.
	
	Example:
	In the Aircon example, the main page image has buttons which need a touch area located over each of them. Position the cursor then draw a touch key area.
	PAGE(MainPage,MainPgStyle)
	{
	POSN(400, 208); KEY(StopKey, StopEvent, 95, 95, TOUCH); //call function StopEvent
	POSN(76, 252); KEY(SaveKey, SaveEvent, 62, 24, TOUCH); //call function SaveEvent
	POSN(+80, +0); KEY(CalibKey, CalibEvent, 62, 24, TOUCH); /call function CalibEvent
	POSN(+80, +0); KEY(ClockKey, [Show(Clock);], 62, 24, TOUCH); //inline code to show clock
	}
	
	Page Styles
	Specify size of page
	sizeX = 1 to 3* LCD width
	sizeY = 1 to 3*LCD height.
	
	Specify the absolute position of the page on the screen. If the page is larger than the screen, the co-ordinates indicate the position of the screen on the page.
	posX = -4 * LCD width to 4 * LCD width
	posY = -4 * LCD height to 4 * LCD height
	
	Specify the background colour of the page
	back = \\000000 to \\FFFFFF or colour name from chart
	
	Specify a background image for the page
	image = "SDHC/xxxxx.bmp" or a variable containing the name of the image
	
	Example Style:
	STYLE(MainPgStyle, Page)
	{ sizeX = 480; sizeY = 272; posX = 0; posY = 0; back = black; image = BackImage; }
	
	Page with size 480x272 or smaller.

LOAD(Dest,Name,Name,....)	Copy Pages and Groups into a previously defined Page or Group . The background and page attributes for 'Dest' apply to the result so only entities are copied from previous pages. This allows simple templates to be merged to form a complex page.
	
	Combine Variables, Buffers and Text and copy the result to a Variable or Buffer. This allows absolute text and variables to be joined together and sent to an interface.
	
	Example:
	LOAD(EditText,EditText,"D"); v0.32 Adds character(s) to end of existing text
	LOAD(RS2,"DATE=", DTIME , "; TEMP=",ACTVAL, "; \\0D\\0A"); v0.20
	LOAD(NumImg,"Image",NUMVAR,".bmp"); v0.23
	LOAD(BasePage,BaseBack,BaseEnglish); not operational for page until 26th Nov
	
	Example Text to Integer/Float
	LOAD(MyInt,MyText); //The text string is parsed until a non-valid numeric value.
	If the string does not start with a number or +/- then the result is 0. v0.36
	
	Example Pointers
	To set/change which entity the entity pointer is pointing to you use '>' instead of ','.
	LOAD(EntPtr1>Var1); // Set EntPtr1 to point to Var1
	
	To put data or an entity name into the entity pointed to by the entity pointer use quotes.
	LOAD(EntPtr1> "ABC"); // Load the Entity pointed to by EntPtr1 with "ABC"
	
	Operational v22 - v34
	Page loading 26th Nov

SHOW(Name)	Show a Page on the Display or reveal a hidden Group or Entity This puts the selected page on the top layer of the screen. If the HIDE() command has previously been used for an entity, it will now appear on a page when the page is shown on the display. Show(Page) can also used to refresh a page if entities have changed. Reserved names provide relative navigation when the name of a page may not be known.. Show(PREV_PAGE); Show the page which launched the current page. Show(THIS_PAGE); Refresh the current page Show(Entity1, Entity2, Entity3...);; multiple show entities then refresh current page **Show is operational V17**
HIDE(Name)	Hide a Page, Group or Entity. If the page on which a small sized page, group or entity is placed is showing on the screen and the page refreshed, the named page, group or entity will disappear from view. Touch, external keys are disabled. Hide(Entity1, Entity2, Entity3...);; multiple hide entities then refresh current page **Hide is operational V17**
DEL(Name)	Delete a Page, Group, Entity, Variable or Buffer from SDRAM. If visible on the display, it will remain until the page is refreshed. If the name refers to an image, font or file stored in the flash library then this is set for memory to be freed using RESET(DELETED); The command DEL("LIBRARY") is used prior to renewing all the application files. Del(Entity1, Entity2, Entity3...); multiple delete entities **Delete is operational V17** **The function RESET(DELETED) to free memory is not available until 16th Dec**

Commands for Cursor Position, Text, Draw, Image and Keys

Command	Description
POSN(X,Y,Page/Name,Style)	Position Cursor +X or –X or X,Y or X, Y, Page with a defined style. The cursor can be positioned on the display using absolute co-ordinates or moved in relation to it's current position by using +/- offset values. If only the X value is changed and Y is to remain the same, only the X value need be specified. The origin is located at the top left of the screen. Re-position a previously placed entity by specifying the new coo-ordinates and it's name. This can be useful for indicator bars, simple movement animations and moving text. **Examples:** POSN(+25,0); moves the cursor 25 pixels to the right. POSN(236,48); absolute position of x=236, y=48. POSN(24,56,CalcPage); position cursor on calc page at x=24, y=56. POSN(VarX,Vary); use variables with absolute values to control position of cursor POSN(VarX,Vary,VertBar); use variables to move an entity - vertical bar **Operational v17-v30**

TEXT(Name,Text,Style)	Create or update Text. Use Carriage Return and Line Feed for multi line entry \\0A\\0D The font and colour are defined in the style. If the cursor relative position is 'CC' (Centre Centre) it is easy to locate text in the centre of images like buttons. Text areas can overlap other text areas when for example a 'drop shadow' is required. Text can include embedded hex codes to access Unicode fonts.
	
	Examples:
	TEXT(EditBox,"Hello World",st8Red12); //style defined else where
	TEXT(EditBox,"Hello People"); //modifies content of EditBox
	TEXT(EditBox,TextVar); //modifies content of EditBox with variable
	TEXT(EditBox,"Hello//w0020World"); // example of unicode embedded character (see fonts page)
	Text is operational except pitch and rotate are fixed.
	
	Editable Text and Visible Cursor
	A text can contain single byte hex of the form \\00 to \\FF
	A text can contain hidden codes for use in password and editable fields.
	\\01 defines the text as a PASSWORD so that only ***** are shown.
	\\02 defines a hidden cursor and \\03 a hidden cursor with insert ON
	\\04 defines an underline cursor and \\05 an underline cursor with insert ON
	\\06 defines a block cursor and \\07 a block cursor with insert ON
	Always place the cursor before the applicable character.
	When a page or text is hidden, the cursor remains at it's current location.
	The CALC command can then be used to manipulate the text and cursor in EditBox.
	User uploaded image arrays for animation and fonts can also handled as text.
	
	Example Editable Text:
	TEXT(EditBox,"Hello \\04World",8ptTextRed); this places an underline cursor at W
	Text manipulation expected 26th Nov
	
	TEXT Styles
	Fonts are available using single byte, 2 byte and UTF8 multi-byte coding.
	Built in ASCII fonts have the reserved names ASCII8, ASCII16, ASCII32.
	Other library fonts are uploaded using the LIB command and have file type .FNT
	These are available for download from the character fonts web page at www.itrontft.com.
	
	font="ASCII8"; to use built in fonts
	font="MyASCII,16PC858"; MyASCII could start at 20H-7FH and 16PC858 at 80H to FFH.
	font="ASC16B,16THAI"; ASC16B could start at 20H and Thai at 0E01H
	
	It is possible to overlay one font over another to enable single byte operation with ASCII from 20H to 7FH and Cyrillic, Greek, Hebrew, Bengali, Tamil, Thai or Katakana from 80H to FFH. The LIB command is used to load the extended font at 0080H instead of it's normal UNICODE location. The style for a text can then specify
	font="MyASCII,MyThai"; causing the Thai to overlap the ASCII from 80H to FFH
	-
	Fonts can be multiplied in size using 1,2,3,4,5,6 etc. Default = 1;
	size=2; a 24x24 font is expanded to a 48x48 font.
	-
	Specify the text color.
	col="\\000000" to "\\FFFFFF" or reserved words from the colour chart.
	-
	Specify the maximum number of characters on a row. Extra characters will be discarded.
	The maximum length = 512.
	maxLen=36;
	-
	Specify the maximum number of rows of the text since new line code \\0D\\0A (CRLF) can be used.
	The maximum rows = 32.
	maxRows=4;
	-
	Specify the relative placement of the text with respect to the cursor.
	curRel = CC Centre Centre , TC Top Centre, BC Bottom Centre, LC Left Centre,
	RC Right Centre, TL Top Left, BL Bottom Left, TR Top Right, BR Bottom Right
	-
	Specify the rotation of the text with respect to the page
	rotate=0, 90, 180, 270.
	-
	Example TEXT Style
	STYLE(Txt8White, Text)
	{ font = "Ascii8,16PC452"; col = white; maxLen = 32; maxRows = 1; curRel = CC; rotate= 0; }
	
	Status: Style operational v22 with rotate = 0.

DRAW(Name,X,Y,Style)	Draw or update a Line, Box or Circle of size X,Y or Pixel at X,Y
	The entities can be an outline or filled.
	The pixel command is different in operation since it is possible to draw a shape or graph with the same entity name for each pixel. If the name is hidden, the whole shape is hidden. The X,Y co-ordinate is the position of the pixel relative to the cursor.
	
	It is possible to specify transparency values with colours if the colour is entered as a 32-bit hex number the top 8 bits specify the alpha blending level.
	col = \\aarrggbb; back = \\aarrggbb; where aa = alpha level.
	For example, col = \\80FFFF00; gives 50% transparent yellow.
	
	DRAW accepts VARs, signed/unsigned integers
	(U8, U16, U32, S8, S16, S32), floats (FLT) and pointers (PTR)
	DRAW(PTR, VAR|INT|FLT|PTR, VAR|INT|FLT|PTR, Style);
	Note PTR refers to the entity being pointed to by PTR and not the PTR itself. Use LOAD(PTR > "Name"); to set a pointer.
	
	Example Draw
	DRAW(MyCircle, 32, 32, DrawCircle);
	DRAW(MyCircle, 64, 64); //modified circle is double diameter.
	DRAW(MyBox,VarX,VarY); //modified box using variables. Should not exceed MaxX,maxY.
	
	Draw Styles
	Specify the type of shape to draw.
	type = B or Box , C or Circle, L or Line, P or Pixel
	
	Specify the border colour of the shape.
	col = \\rrggbb or transparent or none or COLOURNAME or add 32 bit alpha blending
	
	Specify the fill colour of the shape.
	back = \\rrggbb or transparent or none or COLOURNAME or add 32 bit alpha blending
	
	Specify the border width of the shape
	width = width in pixels
	
	Specify the maximum width and height of the shape. This is important for variable size bar graphs and for rotation where the total rotational area must be specified so that sufficient memory can be allocated.
	maxX=160;
	maxY=40;
	
	Specify the relative placement of the shape with respect to the cursor
	curRel = CC Centre Centre , TC Top Centre, BC Bottom Centre, LC Left Centre,
	RC Right Centre, TL Top Left, BL Bottom Left, TR Top Right, BR Bottom Right
	
	Specify the rotation of the shape with respect to the page.
	rotate=0, 90, 180, 270
	
	Example Draw Style
	STYLE(DrawCircle, Draw)
	{ type = circle; col= \\0000FF; back = white; curRel = CC; }
	
	Box and Circle operational v18. Line, pixel and rotate 26th Nov.

IMG(Name,Source,X,Y,Style)	Draw or update an Image of size X,Y. Source has several techniques. If an image is pre-stored in the library, it's entity name is used for Source. If it is on the SDHC card the path is the Source. If it is to be received over an interface see the LIB commands for syntax. The system does not recognize directory structures in the SDHC card. Please put all active files in the root. All file names are 8 characters maximum length. **Example:** IMG(MyPic,TopBtn,90,60,MyImage); **//**previously stored as TopBtn using LIB command IMG(MyPic,"sdhc/TopBtn.bmp",90,60,MyImage); **//**stored on SDHC card **Image Styles** The image may be larger than the size specified so it is necessary to define how it will be scaled. **scale** = F or Fit The image is scaled down to fit the size of the screen area specified **scale** =C or Center The image is center located and any excess border area is not shown. Specify the maximum width and height of the image. This is important for rotation where the total rotational area must be specified so that sufficient memory can be allocated. **maxX**=160; **maxY**=40; Specify the relative placement of the shape with respect to the cursor. **curRel** = CC Centre Centre , TC Top Centre, BC Bottom Centre, LC Left Centre, RC Right Centre, TL Top Left, BL Bottom Left, TR Top Right, BR Bottom Right Specify the rotation of the shape with respect to the page **rotate**=0, 90, 180, 270 **Example Image Style:** STYLE(MyImage, Image) {scale = Fit; curRel = CC; rotate= 0;} **Operational V17 except scale and rotate. Image size must be X,Y.**

KEY(Name,Function,X,Y,Style)	Create a Touch Area of size X,Y or define a Key on the external keyboard. The touch area can have a One Touch function by using the built in style TOUCH. If auto repeat is required, specify the built in style REPEAT. When specifying an external key action, the values for X and Y indicate the contact points on the key board matrix where K0 is \\00 through to K23 which is \\17 . This method allows dual key press capability as in SHIFT key operation The touch screen can be calibrated using the command setup(system) { calibrate=y; } The position of touch keys can be temporarily viewed as a grey area using setup(system) { test=showTouchAreas; } and hidden again using test=hideTouchAreas. **Examples KEY** Key(TopKey,TopFnc,90,50,MyTouch); Key(ExtKey,ExFunc,\\07,\\10,PushKey); This external key operates when K7 and K16 connect. Key(TKey,[Hide(SPage);Show(TPage);],50,50,TOUCH); Inline commands instead of function **KEY Styles** Specify the source of key data. **type**=touch or keyio Specify the time delay to allow a key press to stabilise. Value in milliseconds. **debounce**=250; Specify the time delay before auto repeat occurs. Value in milliseconds. **delay**=1000; If the value is set to 0 then auto repeat is disabled. Specify the repeat period while the key continues to be depressed. Value in milliseconds. **repeat**=500; Specify the up or down action point for the key. **action** = D or down and U or Up. If you require a dual action, specify 2 keys at the same location, one with D and one with U. Specify the relative placement of the touch area with respect to the cursor. **curRel** = CC Centre Centre , TC Top Centre, BC Bottom Centre, LC Left Centre, RC Right Centre, TL Top Left, BL Bottom Left, TR Top Right, BR Bottom Right **Examples KEY Style** STYLE(MyTouch, key) { type=touch; debounce=100; repeat=0; curRel = CC; } STYLE(PushKey, key) { type=keyio; debounce=200; repeat=1500; action=D; curRel=CC; } **Status:OK**

Function Commands

<table>
<tr><th>Command</th><th>Description</th></tr>
<tr><td>VAR(Name,Value,Style)
+ pointer usage
+ non volatile parameter storage</td><td>
Create a variable having a certain style and a default value
A variable contains text or numbers which can be amended but be referred to as a single name in a common equation or to show information on the display. Variable names must start with a letter or _.
Variables can be pointers to other variables and entities and use the '>' operator.
Non volatile parameter storage is also handled by VAR which initially loads the default value, then at subsequent power ON reloads the last stored value which was saved using LOAD(varname,newval);

Example Numbers
VAR(lowval,32.4,FLT1); define lowval as a single decimal float and default value 32.4
VAR(lowval,22.4,FLT1E); define lowval as a single decimal float and default value 22.7
or load EEPROM value if already exists.
Use RESET(EEPROM); to clear and reload only current values.

Example Pointers
To set/change which entity the entity pointer is pointing to you use '>' instead of ','.
LOAD(EntPtr1>Var1); // Set EntPtr1 to point to Var1

To put data into the entity pointed to by the entity pointer, enclose data / source entity in quotes.
LOAD(EntPtr1> "ABC"); // Load the Entity pointed to
by EntPtr1 with "ABC"

The following commands now support entity pointers where (| means 'or this')
> LOAD(name | ptr | "ptr", | > num | "txt" | var | ptr,...);
> CALC(var | ptr, var | ptr, num | var | ptr,"op");
> TEXT(name | ptr, "txt" | var | ptr,...);
> IF(var | ptr op num | "txt" | var | ptr ? func | func_ptr : func | func_ptr);
> KEY(name, func | func_ptr,...);
> INT(name, buf, func | func_ptr,...);
> SHOW(name | ptr,...);
> HIDE(name | ptr,...);
> RUN(name | func_ptr,...);
> IMG(name | img_ptr, lib | img_ptr,...);

VAR Data Styles
Specify the data type. Many are pre-defined for integer, pointer, text and float.
type = U8, U16, U32 - unsigned 8, 16 and 32 bit integer
= S8, S16, S32 - signed 8, 16, 32 bit integer
= TEXT for text strings
= FLOAT for higher resolution calculation
= POINTER for use with images

For text, specify the length from 1 to 8092
length =32; (default)

Specify the number of decimal places when type is float. Range 0 to 7.
decimal =2; (default)

Specify the data location as SDRAM (default) or EEPROM
location = eeprom;

Specify format (used for RTC only)
format = "dd mm YY"; //see RTC page for format character types

Example VAR Style
STYLE(ByteDataZero, Data) { type=U8;location=eeprom;}

Built In Styles (Add E for EEPROM types Example FLT4E)
The following pre defined 'built in' style names are available
U8/U8E - type = U8, U16/U16E - type = U16, U32/U32E - type = U32
S8/S8E - type = S8, S16/S16E - type = S16, S32/S32E - type = S32
PTR/PTRE - type = pointer, TXT/TXTE - type = TEXT, length=32
FLT1/FLT1E - type = float, decimal = 1, FLT2/FLT2E - type = float, decimal = 2
FLT3/FLT3E - type = float, decimal = 3, FLT4/FLT4E - type = float, decimal = 4

Operational v17 thru v34
</td></tr>
<tr><td>IF(Var~Var?Function1:Function2)</td><td>
Compare variables, buffers or text for value or length.
If true, do function1, if false do function2 (optional).

The operators allowed for numeric values are:
=, == equal to
<>, != not equal to
< less than
> greater than
<= less than or equal to
>= greater than or equal to
+ sum not equal to zero
- difference not equal to zero
* multiplication not equal to zero
/ division not equal to zero
% modulus not equal to zero
& logical AND
| logical OR
^ logical exclusive-OR
=- equal to the negative of
&& Boolean AND
|| Boolean OR

The operators allowed for text strings are:
=, == equal to
> greater than
< less than
>= greater than or equal to
<= less than or equal to
<>, != not equal
~= same length
~< length shorter than
~> length longer than
~! not same length

Examples:
IF(K0="L"?HELPFNC); //single condition
IF(HIGHVAL < ACTVAL ? HIGHFUNC : LOWFUNC);
IF(STRVAR~>0? SHOWFUNC); //if STRVAR length > 0 show data
IF(STARVAL >= -STARTMP?SHOWSTAR);
IF(STARVAL > 0? [LOAD(vReqd,15); TEXT(txtCurFlr,"15"); RUN(fncGo);]); //uses in line code [..]

Operational v18
</td></tr>
</table>

LOOP(Name,Var1){.....}	Repeats the specified actions a number of times then continue. Max 12 nested loops or functions.
	The value for Var1 can be a number from 1-65000 or the text FOREVER
	
	Examples:
	LOOP(MyLoop,12){Show(Page1);wait(100);show(page2);wait(100);} //repeat 12 times
	LOOP(MyLoop,FOREVER) {Show(Page1);wait(100);show(page2);wait(100);}
	
	Operational v17

INT(Name,Buffer,Function)	If an interrupt occurs for the specified buffer, do function.
	An interrupt will occur when a buffer's style parameters allow activity within the buffer and the appropriate type of interrupt is set.
	Serial interfaces can trigger on a byte received, a byte transmitted and a semi-colon (command separator) received. I/O can trigger on input change.
	Use HIDE(Name); to disable an interrupt.
	
	This is currently set to interrupt on each character received for the 'Buffer':
	> RS2RXC = RS232 Receive Character
	> RS4RXC = RS485 Receive Character
	> AS1RXC = Async1 Receive Character
	> AS2RXC = Async2 Receive Character
	> DBGRXC = Debug Receive Character
	> I2CRXC = I2C Receive Character
	
	NOTE: The Buffer must be read to clear the interrupt otherwise the Function will keep getting called!
	Example:
	PAGE(PageName, PageStyle)
	{
	INT(SerRxInt, RS2RXC, SerRxEvent);
	}
	FUNC(SerRxEvent)
	{
	LOAD(Var, RS2); // Must read RS2 to clear interrupt
	LOAD(RS4, Var); //send out of RS485 interface.
	TEXT (RecvTxt, Var);; //show received ASCII data on screen
	// and refresh
	}
	Interrupts for counters, transmit, I/O triggers 28th Oct

CALC(Result,VarA,VarB,Method)	Numeric Handling This provides a fast simple calculation placed in the Result variable according to the type of method using + , - , / , *, %(modulus) or logical functions | (OR) & (AND) ^ (EXOR) for non float. Example: CALC(NumUp,NumUp,1,"+"); increments NumUp CALC(FltNum,1.0,FltVal,"-"); first parameter defines type for 2nd and 3rd parameter. CALC(Result,Request,8,"&"); the Result will equal 0 or 8 if bit3 is set in Request. CALC(SumPtr,PtrA,PtrB,"/"); use pointers for the calculation **Calc is operational for integer, long and float from v0.22 Trig functions Sin, Cos, Tan TBA.** Text and Cursor Handling Calc can be used for text and cursor manipulation where editable text is to be placed on the screen as in a calculator or editable text field. Various methods allow cursor movement and type, text insertion and deletion, find or delete text, cursor position and length. VarA contains the existing text and VarB the modifier text, cursor position or a text length. Example: CALC(EditBox,EditBox, "A","INS"); Inserts the letter 'A' into the text at the cursor position Cursor and Text Types \\01 defines the text as a PASSWORD so that only ***** are shown until another \\01 or end;. \\02 defines a hidden cursor with over write and \\03 a hidden cursor with insert ON \\04 defines an underline cursor with over write and \\05 an underline cursor with insert ON \\06 defines a block cursor with over write and \\07 a block cursor with insert ON Method Types INS Add text in VarB at cursor position according to cursor type (Over write or Insert) DEL Delete text of length VarB at cursor position and shift remaining text left If VarB is negative then text is deleted before the cursor as in Back Space POS Move cursor to absolute position in text or relative to existing position specified in VarB Absolute VarB = 1 to n and for Relative VarB= -n to +n FIND Result gives the start position of text VarB in VarA REM Any occurrence of the text VarB in VarA is removed and the text shifted left. CUR The cursor or text type is changed at the current position to type VarB (\\01 to \\07) LEN Result contains the current length of the text in characters plus VarB. PIX Result contains the current length of the named text entity in pixels plus VarB. LOC Result contains the position of the cursor in the text plus offset in VarB (-n to +n) TYPE Result contains the type of text and cursor used - \\01 to \\07 or \\00 if none present. AFT Result contains VarB characters after cursor position in string VarA. If no cursor, use first Example CALC(result,"abc\\02defghij",4,"AFT"); result="defg" BEF Result contains VarB characters before cursor position in string VarA. If no cursor, use end Example CALC(result,"abc\\02defghij",2,"BEF"); result="bc" **Text Calc from 8th December**
FUNC(Name) {...}	Create a function called by commands which returns to the next command on completion. Functions can call other functions and themselves. No storing or passing of variables occurs as these are all global even if created in a function. Max 12 nested loops or functions. **Func is operational V17**
RUN(Name)	Run previously defined user code or functions. User code is supplied in C and compiled by our firmware department subject to order. Functions can be run as macros for compact menu design. RUN(Func1); or RUN(Func1,Func23,Func3...FuncN); or a pointer to a function RUN(func-ptr); **Operational v17.** **User code TBD**.
WAIT(Time)	Wait for a period of milliseconds before processing menu commands. Interrupts and key presses will still occur and be processed. **Wait is operational as a delay in v17.**

Character Fonts

Compact Narrow Fonts	Wide Rounded Fonts
ASCII Base Page	ASCII + European
PC437 (USA - European Standard)	Cyrillic
PC850 (Multilingual)	Greek
PC852 (Latin 2)	Arabic
PC858 (Multilingual)	Hebrew
PC860 (Portuguese)	Bengali
PC863 (Canadian French)	Tamil
PC865 (Nordic)	Thai
PC866 (Cyrillic)	Chinese/Japanese/Korean TBA
WPC1252	Hangul TBA
Katakana	Katakana

You can include the character fonts required for an application by downloading the attached files and use the LIB command to store them in memory. You can setup your system to process text as single byte, 2 byte UNICODE or multibyte UTF8. See the LIB command for installing fonts. System fonts ASCII8,ASCII16 and ASCII32 are built in. The wide rounded fonts are preferred for higher quality designs.

It is possible to overlay one font over another to enable single byte operation with ASCII from 20H to 7FH and Cyrillic, Greek, Hebrew, Bengali, Tamil, Thai or Katakana from 80H to FFH. The LIB command is used to load the extended font at 0080H instead of it's normal UNICODE location. The style for a text can then specify font="MyASCII,MyThai"; causing the Thai to overlap the ASCII from 80H to FFH

STANDARD ASCII - 20H to 7FH
Standard ASCII text in the range 20H to 7FH can by directly typed from the keyboard.
System fonts named ASCII8, ASCII16, ASCII32 are pre-installed.
Example TEXT(txt1, "Hello World", stTXT); //single byte access to 20H to 7FH ASCII characters

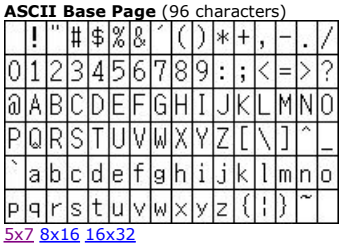
EXTENDED ASCII - 20H to FFH
2/ When using single byte ASCII in the range 20H to 7FH, you can access extended characters from 80H to FFH using hex code like \\AB
Example TEXT(txt1, "1. AB\\B0CDEF \\AB s", stTXT); //single byte access to 80H to FFH

UNICODE and UTF8
3/ When using single byte ASCII in the range 20H to 7FH, you can access UNICODE characters by using hex code like \\w0D7F
or a UTF8 character using hex code like \\mC2AB. The symbols <....> are used where more than one character is coded.
Examples

```
TEXT( txt2, "2. AB\\w00B0CDEF \\w00AB", stTXT );                  // UNICODE double byte access to 0080H to FFFFH
TEXT( txt3, "3. AB\\mC2B0CDEF \\mC2AB", stTXT );                  // UTF8 multi byte access to 80H to FFFFH
TEXT( txt5, "5. AB\\sB0C\\w<00440045>F \\w00AB", stTXT );   // <....> are used for long hex strings \\s is used for single byte in a UNICODE or UTF8 encoded system
TEXT( txt7, "\\<372E204142B04344454620AB>", stTXT );        //  string of single byte hex in the range 20H to 80H
TEXT( txt8, "\\w<0038002e00200041004200B0043004400450046002000AB>", stTXT );
TEXT( txt9, "\\m<392E204142C2B04344454620C2AB>", stTXT );
```

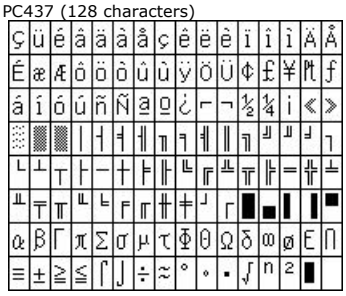
COMPACT NARROW FONTS (Single Byte Range 20H to FFH or UNICODE Range 0020H to 00FFH)
The ASCII base page is included automatically at 20H-7FH and the other fonts are automatically loaded to 80H to FFH.
This gives a single byte range of 20H to FFH.

ASCII Base Page (96 characters)



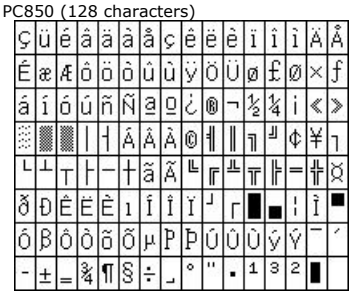
5x7 8x16 16x32

PC437 (128 characters)



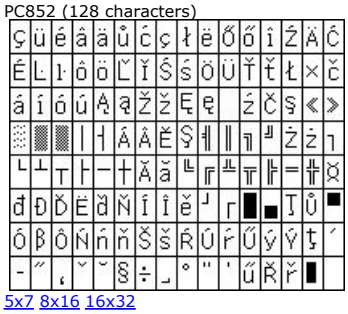
5x7 8x16 16x32

PC850 (128 characters)



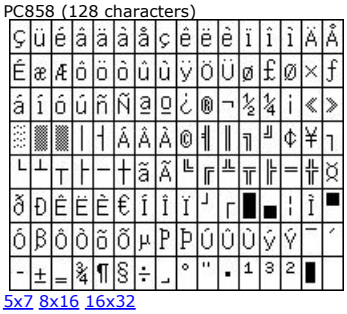
5x7 8x16 16x32

PC852 (128 characters)



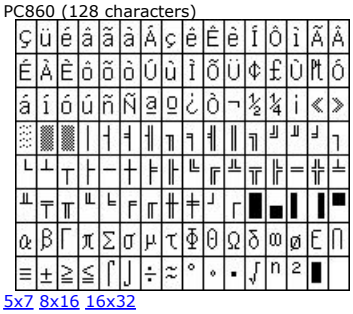
5x7 8x16 16x32

PC858 (128 characters)



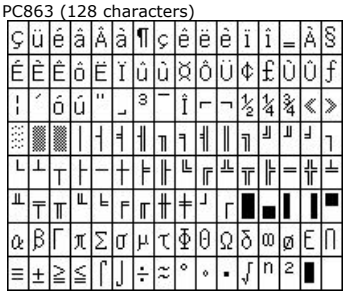
5x7 8x16 16x32

PC860 (128 characters)



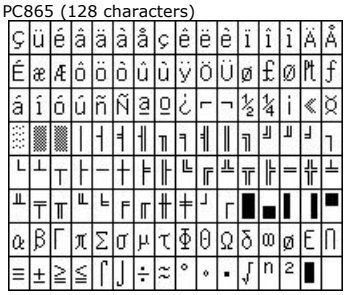
5x7 8x16 16x32

PC863 (128 characters)



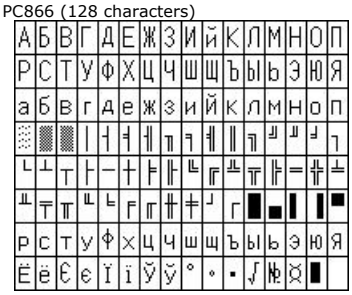
5x7 8x16 16x32

PC865 (128 characters)



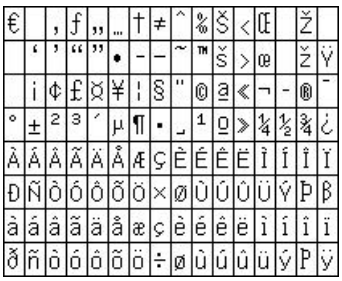
5x7 8x16 16x32

PC866 (128 characters)



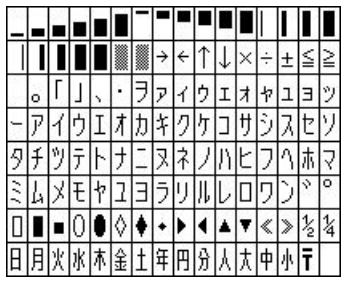
5x7 8x16 16x32

WPC1252 (128 characters)



5x7 8x16 16x32

Katakana (128 characters)



5x7 8x16 16x32

WIDE ROUNDED Fonts (Single Byte Range 20H to FFH or UNICODE Range 0020H to FFFFH)
When loading these fonts into library, it is necessary to specify the offset address for the first character of each font table. The supplementary characters above FFFF are not supported in UTF8.

ASCII + European (467 characters)

A
16px (3.2mm)
A
24px (4.8mm)
A
32px (6.4mm)
40px (8mm)
48px (9.6mm)
60px (12mm)
72px (14.4mm)

	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Unicode Range
0020 - 0217

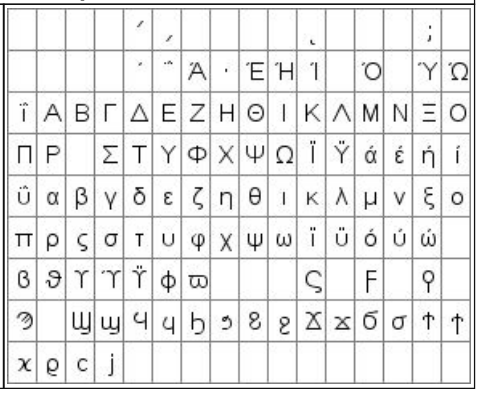
Cyrillic (226 characters)

Ъ
16px (3.2mm)
Ъ
24px (4.8mm)
Ъ
32px (6.4mm)

Unicode Range
0401 - 04F9

Greek (105 characters)

Ω
16px (3.2mm)
Ω
24px (4.8mm)
Ω
32px (6.4mm)



Unicode Range
0374 - 03F3

Arabic (194 characters)

ﺵ
16px (3.2mm)
ﺵ
24px (4.8mm)
ﺵ
32px (6.4mm)

Unicode Range
060C - 06F9

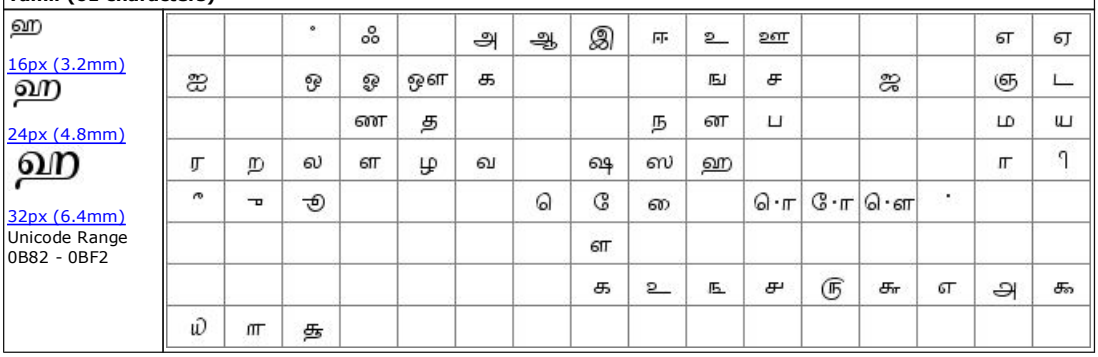
Hebrew (82 characters)

16px (3.2mm)
24px (4.8mm)
32px (6.4mm)
Unicode Range
0591 - 05F4

Bengali (89 characters)

16px (3.2mm)
24px (4.8mm)
32px (6.4mm)
Unicode Range
0981 - 09FA

Tamil (61 characters)



16px (3.2mm)
24px (4.8mm)
32px (6.4mm)
Unicode Range
0B82 - 0BF2

Thai (87 characters)

๘
16px (3.2mm)
๘
24px (4.8mm)
๘
32px (6.4mm)

Unicode Range
0E01 - 0E5B

Chinese/Japanese/Korean (21151 characters) TBA

挋
16x16
(3.2mm)
挑
24x24
(4.8mm)
挑
32x32
(6.4mm)

Unicode Range
3300 - 9FA5

Hangul (11172 characters) TBA

냿
16x16
(3.2mm)
냿
24x24
(4.8mm)
냿
32x32
(6.4mm)

Unicode Range
AC00 - D7A3

Katakana (94 characters)

ポ
16x16 (3.2mm)

ポ
24x24 (4.8mm)

ポ
32x32 (6.4mm)

Unicode Range
30A1 - 30FE

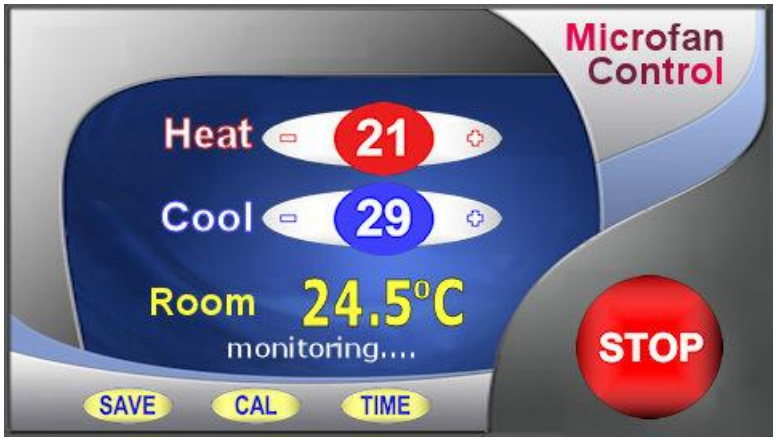
	ァ	ア	ィ	イ	ゥ	ウ	ェ	エ	ォ	オ	カ	ガ	キ	ギ	ク
グ	ケ	ゲ	コ	ゴ	サ	ザ	シ	ジ	ス	ズ	セ	ゼ	ソ	ゾ	タ
ダ	チ	ヂ	ッ	ツ	ヅ	テ	デ	ト	ド	ナ	ニ	ヌ	ネ	ノ	ハ
バ	パ	ヒ	ビ	ピ	フ	ブ	プ	ヘ	ベ	ペ	ホ	ボ	ポ	マ	ミ
ム	メ	モ	ャ	ヤ	ュ	ユ	ョ	ヨ	ラ	リ	ル	レ	ロ	ヮ	ワ
ヰ	ヱ	ヲ	ン	ヴ	ヵ	ヶ	ヷ	ヸ	ヹ	ヺ	・	ー	ヽ	ヾ	

Colour Chart

The colour chart below shows the built in colours of the TFT module. To clarify the reference name of a colour, hover over the hex code.

#4682B4 steelblue	#041690 royalblue	#6495ED cornflowerblue	#B0C4DE lightsteelblue	#7B68EE mediumslateblue	#6A5ACD slateblue	#483D8B darkslateblue	#191970 midnightblue	#000080 navy	#00008B darkblue
#0000CD mediumblue	#0000FF blue	#1E90FF dodgerblue	#00BFFF deepskyblue	#87CEFA lightskyblue	#87CEEB skyblue	#ADD8E6 lightblue	#B0E0E6 powderblue	#F0FFFF azure	#E0FFFF lightcyan
#AFEEEE paleturquoise	#48D1CC mediumturquoise	#20B2AA lightseagreen	#008B8B darkcyan	#008080 teal	#5F9EA0 cadetblue	#00CED1 darkturquoise	#00FFFF aqua	#00FFFF cyan	#40E0D0 turquoise
#7FFFD4 aquamarine	#66CDAA mediumaquamarine	#8FBC8F darkseagreen	#3CB371 mediumseagreen	#2E8B57 seagreen	#006400 darkgreen	#008000 green	#228B22 forestgreen	#32CD32 limegreen	#00FF00 lime
#7FFF00 chartreuse	#7CFC00 lawngreen	#ADFF2F greenyellow	#9ACD32 yellowgreen	#98FB98 palegreen	#90EE90 lightgreen	#00FF7F springgreen	#00FA9A mediumspringgreen	#556B2F darkolivegreen	#6B8E23 olivedrab
#808000 olive	#BDB76B darkkhaki	#B8860B darkgoldenrod	#DAA520 goldenrod	#FFD700 gold	#FFFF00 yellow	#F0E68C khaki	#EEE8AA palegoldenrod	#FFEBCD blanchedalmond	#FFE4B5 moccasin
#F5DEB3 wheat	#FFDEAD navajowhite	#DEB887 burlywood	#D2B48C tan	#BC8F8F rosybrown	#A0522D sienna	#8B4513 saddlebrown	#D2691E chocolate	#CD853F peru	#F4A460 sandybrown
#8B0000 darkred	#800000 maroon	#A52A2A brown	#B22222 firebrick	#CD5C5C indianred	#F08080 lightcoral	#FA8072 salmon	#E9967A darksalmon	#FFA07A lightsalmon	#FF7F50 coral
#FF6347 tomato	#FF8C00 darkorange	#FFA500 orange	#FF4500 orangered	#DC143C crimson	#FF0000 red	#FF1493 deeppink	#FF00FF fuchsia	#FF00FF magenta	#FF69B4 hotpink
#FFB6C1 lightpink	#FFC0CB pink	#DB7093 palevioletred	#C71585 mediumvioletred	#800080 purple	#8B008B darkmagenta	#9370DB mediumpurple	#8A2BE2 blueviolet	#4B0082 indigo	#9400D3 darkviolet
#9932CC darkorchid	#BA55D3 mediumorchid	#DA70D6 orchid	#EE82EE violet	#DDA0DD plum	#D8BFD8 thistle	#E6E6FA lavender	#F8F8FF ghostwhite	#F0F8FF aliceblue	#F5FFFA mintcream
#F0FFF0 honeydew	#FAFAD2 lightgoldenrodyellow	#FFFACD lemonchiffon	#FFF8DC cornsilk	#FFFFE0 lightyellow	#FFFFF0 ivory	#FFFAF0 floralwhite	#FAF0E6 linen	#FDF5E6 oldlace	#FAEBD7 antiquewhite
#FFE4C4 bisque	#FFDAB9 peachpuff	#FFEFD5 papayawhip	#F5F5DC beige	#FFF5EE seashell	#FFF0F5 lavenderblush	#FFE4E1 mistyrose	#FFFAFA snow	#FFFFFF white	#F5F5F5 whitesmoke
#DCDCDC gainsboro	#D3D3D3 lightgrey	#C0C0C0 silver	#A9A9A9 darkgray	#808080 gray	#778899 lightslategray	#708090 slategray	#696969 dimgray	#2F4F4F darkslategray	#000000 black

Air Conditioning Control System



The Start button is stored separately and placed over the top of the stop button.
The commands HIDE and SHOW are used to control visibility.
The text areas are the '21' , the '29' , the '24.5' , the 'monitoring...'
The touch areas cover the buttons (+ - + - SAVE CAL TIME STOP).
The changed HEAT and COOL parameters are stored in EEPROM
Download Images <zip>

Air Conditioning Control System Code using V30 software
(Highlight, cut and paste below this line)

```
// Menu file AIRCON.MNU for Air Conditioner using TU480X272C
// Updated 20-Sep-2010

// ------------------------------------------------------------------------
// Air Conditioner Page
// ------------------------------------------------------------------------
LIB( libImgAcBg, "SDHC/AirConBg.bmp" ); // Load background picture
LIB( libImgAcStart, "SDHC/AirConSt.bmp?back=\\76bbfe" ); // Load start button + transparency

STYLE( stAcMainPg, Page ) { back=black; image=libImgAcBg; }
STYLE( stTxt32Yel06, Text ) { font=fntAscii32; col=yellow; maxLen=6; maxRows=1; curRel=CC; }
STYLE(StGenImg,Image) {curRel=CC;}

VAR( varAcHeat, 26, U8E );
VAR( varAcCool, 20, U8E );
VAR( varAcAct, 32.7, FLT1 );
VAR( varAcDif, 0.196, FLT1 );
VAR( varAcTmp, 0.0, FLT1 );
VAR( varAcCnt, 0, U8 );

PAGE( pgAirConMain, stAcMainPg )
{
// Heating Upper Limits
POSN( 238, 80 ); TEXT( txtAcHeat, varAcHeat, stTxt32Wht64 ); // Draw text for upper limit
POSN( -58, -2 ); KEY( keyAcHeatDn, fncAcHeatDn, 40, 30, TOUCH );
POSN( +110, +0 ); KEY( keyAcHeatUp, fncAcHeatUp, 40, 30, TOUCH );

// Cooling Lower Limts
POSN( 238, +52 ); TEXT( txtAcCool, varAcCool, stTxt32Wht64 ); // Draw text for lower limit
POSN( -58, -2 ); KEY( keyAcCoolDn, fncAcCoolDn, 40, 30, TOUCH );
POSN( +110, +0 ); KEY( keyAcCoolUp, fncAcCoolUp, 40, 30, TOUCH );

POSN( 200, +85 ); TEXT( txtAcMsg, "Set Limits or press START", stTxt8Wht64 ); // Draw text for prompts
POSN( 238, 182 ); TEXT( txtAcAct, varAcAct, stTxt32Yel06 ); // Draw text for actual value
POSN( 400, 208 ); KEY( keyAcStop, fncAcStop, 95, 95, TOUCH ); // Stop

// Load green start button over top of red stop button and start touch area
IMG( imgAcStart, libImgAcStart, 95, 95, stGenImg );
KEY( keyAcStart, fncAcStart, 95, 95, TOUCH ); // Start

POSN( 76, 252 ); KEY( keyAcSave, fncDemoExit, 62, 24, TOUCH );
POSN( +80, +0 ); KEY( keyAcExit, fncDemoExit, 62, 24, TOUCH );
POSN( +80, +0 ); KEY( keyAcTime, fncDemoExit, 62, 24, TOUCH );

// Commands run as loop on page
LOOP( lpAcMain, FOREVER )
  {
  IF( varAcCnt != CNTSECS ? fncAcUpd );
  RUN( fncDemoUpdate );
  }
}

FUNC( fncAcHeatUp) { RUN(fncDemoPause);CALC(varAcHeat,varAcHeat,1,"+");TEXT(txtAcHeat,varAcHeat);; }

FUNC( fncAcHeatDn)
 {
 RUN(fncDemoPause);CALC(varAcHeat,varAcHeat,1,"-");TEXT(txtAcHeat,varAcHeat);IF(varAcCool=varAcHeat?fncAcCoolDn);;
 }
```

```
FUNC( fncAcCoolUp )
 {
 RUN(fncDemoPause);CALC(varAcCool,varAcCool,1,"+");TEXT(txtAcCool,varAcCool);IF(varAcHeat=varAcCool?fncAcHeatUp);;
 }

FUNC( fncAcCoolDn ) { RUN(fncDemoPause);CALC(varAcCool,varAcCool,1,"-");TEXT(txtAcCool,varAcCool);; }

FUNC( fncAcUpd )
 {
 LOAD(varAcCnt,CNTSECS);
 CALC(varAcAct,varAcAct,varAcDif,"+");
 IF(txtAcMsg!="Set Limits or press START"?fncAcOn:fncAcOff);
 TEXT(txtAcAct,varAcAct);;
 }

FUNC( fncAcOn ) { IF(varAcAct>varAcHeat?fncAcCool);IF(varAcAct<varAcCool?fncAcHeat); }
FUNC( fncAcCool ) { TEXT(txtAcMsg,"Running... COOLING" );LOAD(varAcDif,-0.27); }
FUNC( fncAcHeat ) { TEXT(txtAcMsg,"Running... HEATING" );LOAD(varAcDif,+0.27); }

FUNC( fncAcOff )
  {
  CALC(varAcTmp,varAcHeat,10,"+");
  IF(varAcAct>varAcTmp?fncAcActHi);
  CALC(varAcTmp,varAcCool,10,"-");
  IF(varAcAct<varAcTmp?fncAcActLo);
  }

FUNC( fncAcActHi ) { LOAD(varAcDif,-0.12); }
FUNC( fncAcActLo ) { LOAD(varAcDif,+0.12); }
FUNC( fncAcStart ) { RUN(fncDemoPause);HIDE(imgAcStart,keyAcStart);TEXT(txtAcMsg,"Running...");; }
FUNC( fncAcStop ) { RUN(fncDemoPause);SHOW(imgAcStart,keyAcStart);TEXT(txtAcMsg,"Set Limits or press START");; }

//Run Main Page
SHOW( pgAirConMain );
```

Elevator Control System

The user can select a floor and travel from any floor to another floor.
The arrow is selected according to direction.
Warning signs for doors opening and closing are used.
Variables are used to store the current floor and destination floor.
An RS232 interface could be added to communicate with other floor indicators.
Download Image Files <zip>

Elevator System Code using V30 software
(Highlight, cut and paste below this line)

```
// Menu file TU480A.MNU for Elevator System using TU480X272C
// Updated 20-Sep-2010
// Floors are 15(15)..01(1), G(0), P1(-1), B2(-2)

// Load images into the library
LIB(libImgNum0,"SDHC/Lift0.bmp?back=\\0000CD"); // Load Number 0
LIB(libImgNum1,"SDHC/Lift1.bmp?back=\\0000CD"); // Load Number 1
LIB(libImgNum2,"SDHC/Lift2.bmp?back=\\0000CD"); // Load Number 2
LIB(libImgNum3,"SDHC/Lift3.bmp?back=\\0000CD"); // Load Number 3
LIB(libImgNum4,"SDHC/Lift4.bmp?back=\\0000CD"); // Load Number 4
LIB(libImgNum5,"SDHC/Lift5.bmp?back=\\0000CD"); // Load Number 5
LIB(libImgNum6,"SDHC/Lift6.bmp?back=\\0000CD"); // Load Number 6
LIB(libImgNum7,"SDHC/Lift7.bmp?back=\\0000CD"); // Load Number 7
LIB(libImgNum8,"SDHC/Lift8.bmp?back=\\0000CD"); // Load Number 8
LIB(libImgNum9,"SDHC/Lift9.bmp?back=\\0000CD"); // Load Number 9
LIB(libImgBChar,"SDHC/LiftB.bmp?back=\\0000CD"); // Load Character B
LIB(libImgGChar,"SDHC/LiftG.bmp?back=\\0000CD"); // Load Character G
LIB(libImgPChar,"SDHC/LiftP.bmp?back=\\0000CD"); // Load Character P
LIB(libImgDTri,"SDHC/LiftDown.bmp?back=\\0000CD"); // Load red triangle
LIB(libImgUTri,"SDHC/LiftUp.bmp?back=\\0000CD"); // Load green triangle
LIB(libImgPMTD,"SDHC/LiftClos.bmp"); // Load the warning message
LIB(libImgSelFlr,"SDHC/LiftSel.bmp"); //Load the Select Floor Page
LIB(libImgDoors,"SDHC/LiftOpen.bmp"); // Load the Doors Page

// Create styles
STYLE(stLiftPg,Page){back=\\0000CD;}
STYLE(stLiftMainPg,Page){back=\\0000CD;image=libImgSelFlr;}
STYLE(StGenImg,Image) {curRel=CC;}

// Create vars
VAR(vS8,0,S8);
VAR(ptrLiftArr>"libImgUTri",PTR);
VAR(ptrLiftTens>"libImgGChar",PTR);
VAR(ptrLiftOnes>"libImgNum1",PTR);
VAR(vReqd,0,S8);
VAR(vThis,0,S8);
VAR(vMove,0,U8);

// Create Select Floor Page
PAGE(pgLiftMain,stLiftMainPg)
{
POSN(69,78); KEY(keyFlr15,[LOAD(vReqd,15);TEXT(txtCurFlr,"15");RUN(fncGo);],104,84,TOUCH);
POSN(184,+0); KEY(keyFlr14,[LOAD(vReqd,14);TEXT(txtCurFlr,"14");RUN(fncGo);],104,84,TOUCH);
POSN(298,+0); KEY(keyFlr13,[LOAD(vReqd,13);TEXT(txtCurFlr,"13");RUN(fncGo);],104,84,TOUCH);
POSN(409,+0); KEY(keyFlr12,[LOAD(vReqd,12);TEXT(txtCurFlr,"12");RUN(fncGo);],104,84,TOUCH);
POSN(69,179); KEY(keyFlr11,[LOAD(vReqd,11);TEXT(txtCurFlr,"11");RUN(fncGo);],104,84,TOUCH);
POSN(184,+0); KEY(keyFlrG, [LOAD(vReqd, 0);TEXT(txtCurFlr, "G");RUN(fncGo);],104,84,TOUCH);
POSN(298,+0); KEY(keyFlrP1,[LOAD(vReqd,-1);TEXT(txtCurFlr,"P1");RUN(fncGo);],104,84,TOUCH);
POSN(409,+0); KEY(keyFlrB2,[LOAD(vReqd,-2);TEXT(txtCurFlr,"B2");RUN(fncGo);],104,84,TOUCH);
POSN(239,256);KEY(keyFlrExit,fncDemoExit,480,30,TOUCH);
POSN(160,249);TEXT(txtCurFlrLbl,"CURRENT FLOOR",stTxt32Wht64);
POSN(330,+0); TEXT(txtCurFlr,"G",stTxt32Wht64);
LOOP(lpLiftMain,FOREVER){RUN(fncDemoUpdate);}
}

// Level indication page
PAGE(pgIND,stLiftPg)
{
POSN(48,136);IMG(imgTri,ptrLiftArr,86,200,stGenImg);HIDE(imgTri);
POSN(199,+0);IMG(img10s,ptrLiftTens,172,240,stGenImg);
POSN(396,+0);IMG(img1s,ptrLiftOnes,172,240,stGenImg);
```

```
LOOP(lpLiftInd,FOREVER){IF(vMove=1?fncMove);}
}

// Lift is moving
FUNC(fncMove)
{
IF(vThis>vReqd?[LOAD(ptrLiftArr>"libImgDTri");IMG(imgTri,ptrLiftArr);SHOW(imgTri);RUN(fncShowFlr);CALC(vThis,vThis,1,"-");]);
IF(vThis<vReqd?[LOAD(ptrLiftArr>"libImgUTri");IMG(imgTri,ptrLiftArr);SHOW(imgTri);RUN(fncShowFlr);CALC(vThis,vThis,1,"+");]);
IF(vThis=vReqd?[LOAD(vMove,0);HIDE(imgTri);RUN(fncShowFlr);RUN(fncDoorOpen);SHOW(pgLiftMain);]);
}

// Start lift moving
FUNC(fncGo){RUN(fncDemoPause);LOAD(vMove,1);HIDE(imgTri);RUN(fncDoorClose);RUN(fncShowFlr);}

// Show Current Floor
FUNC(fncShowFlr)
{
IF(vThis>0?[CALC(vS8,vThis,10,"/");LOAD(ptrLiftTens>"libImgNum",vS8);CALC(vS8,vThis,10,"%");LOAD(ptrLiftOnes>"libImgNum",vS8);SHOW
(img10s,img1s);]);
IF(vThis=0 ?[LOAD(ptrLiftTens>"libImgGChar");SHOW(img10s);HIDE(img1s);]);
IF(vThis=-1?[LOAD(ptrLiftTens>"libImgPChar");LOAD(ptrLiftOnes>"libImgNum1");SHOW(img10s,img1s);]);
IF(vThis=-2?[LOAD(ptrLiftTens>"libImgBChar");LOAD(ptrLiftOnes>"libImgNum2");SHOW(img10s,img1s);]);
IMG(img10s,ptrLiftTens);IMG(img1s,ptrLiftOnes);
SHOW(pgIND);
WAIT(1000);
}

// Create Door Closing and Opening
FUNC(fncDoorClose){SHOW(pgShut);WAIT(400);SHOW(pgBlnk);WAIT(100);SHOW(pgShut);WAIT(400);SHOW(pgBlnk);WAIT(100);SHOW
(pgShut);WAIT(400);SHOW(pgBlnk);WAIT(100);}
FUNC(fncDoorOpen){SHOW(pgOpen);WAIT(400);SHOW(pgBlnk);WAIT(100);SHOW(pgOpen);WAIT(400);SHOW(pgBlnk);WAIT(100);SHOW
(pgOpen);WAIT(400);SHOW(pgBlnk);WAIT(100);}

// Create Door Closing / Opening / Blank Pages
PAGE(pgShut,stLiftPg){POSN(239,135);IMG(imgDC,libImgPMTD,480,272,stGenImg);}
PAGE(pgOpen,stLiftPg){POSN(239,135);IMG(imgDO,libImgDoors,480,272,stGenImg);}
PAGE(pgBlnk,stLiftPg){}

//RUN Main page
SHOW(pgLiftMain);
// Menu file LIFT.MNU for Elevator System using TU480X272C
// Updated 12-Oct-2010
// Floors are 15(15)..01(1), G(0), P1(-1), B2(-2)
```

Frequently Asked Questions

Please send your questions to tech@noritake-itron.com
We will try to respond within 24 hours (Monday to Thursday)

Product Status and Availability
Start Programming
Memory and SDCard
Interfacing
Others

PRODUCT STATUS and AVAILABILITY
- What is the availability of this product ?
TU480X272C-K6121TU prototypes have been supplied to 82 companies in Europe from May Yes
The planned market release: 16th September 2010 Yes
120 units will be available from the 5th October 2010. Yes
5.7 inch and 7inch prototypes will be available on the 10th November 2010. Yes
5.7 inch and 7inch prototypes will be available on the 10th December 2010. On schedule
3.5inch prototypes will be available in December 2010. Delayed to 14th January
1800 units of 4.3inch. Kitted awaiting PCB.

- What is the firmware development status of this product ?
We thank the 20 companies in Europe who helped beta testing
As at 12th Nov, 95% of the specified software functionality complete.
All existing customers are updated by email with latest release.
Customer identified bug fix is implemented in the next release (7 days).
USB interfacing, arrays and rotation are the main features still in development.
USB COM is aimed to be available by 26th Nov to make file update easier.
Future developments include automated animation plus audio and video processing.

START PROGRAMMING
- What is needed to start a development with this TFT? Need SW-Development, HW Development-Kit ?
Hardware: TFT module plus SDCard plus SD Card Reader
Software: PC wit text editor (notepad) and image editor (paint or paint-shop)
The main objective of this product was to avoid expensive development tools for the customer.
At this time the TU480A.mnu file on the SDCard must be edited externally using the supplied adaptor via a reader.
The USB cable is used only for easy supply of power.
When the SDCard is removed, the PC may ask for a USB driver. Press CANCEL to this request.
Once USB has been developed, this will allow the user to edit files via the USB as a Mass Storage Device.
1GB SD and SDHC 4G, 8G, 16G or 32G card can be used for im480c1.bin or other files.
2G SD not supported.

- When programming the display, the PC is used and the software will be memorize in the SD card ?
Yes, but then use it to program modules on the production line.
The module includes the 128Mbyte Flash ROM inside the TFT so the customer program is automatically loaded from SDCard to flash when SDCard is plugged in and has a file called TU480a.mnu. This is a ONE TIME action.
When the card is removed or .mnu file is not present, it uses the last loaded program.

- Can we program directly via the usb port into the flash ?
Yes, via SD card, then direct (Estimated from 10th Nov)
When the user connects USB, they must have SDCard inserted. They will then see the SDCard on their PC as an F: drive.
They can then copy files to the drive. When the power is reapplied, new files are loaded into flash.
The flash memory will also appear as H: drive and can be accessed directly.

- Is the TFT interpreting a text based program language without the user compiling the software?
This is correct, although the TFT module internally converts the data to achieve the necessary speed.

-Is it possible to do screen captures for manuals or to run the interface on a PC as a simulator?
We will be supplying iDevTFT for editing, local debugging and PC simulation.

- Can we find a complete example?
Users can download examples from the web and the module can be supplied with pre-loaded applications.
We are showing an Aircon and Elevator Systems at this time, with others available soon.
These are updated as the firmware permits.

- Is the module available with Linux operating system?
We understand that some Linux has been written for this processor.
Work on a Linux solution is scheduled for mid 2011.
There are support issues for Linux and Windows CE.

MEMORY and SD CARD
- Is it possible to download the text based program into Flash memory and then the TFT can be driven without SD-Card?
Yes, the SDCard is used as a convenient way of programming the module.
Like a CD on PC, you do not need the CD after installing a program.
The user can use instruction FPROG to store their program and images in FLASH. (available Dec 2010)

- The specification says that user code is generated at Noritake – will this be changed in the future?
The menu and function code is generated by the customer.
If the customer wants special functionality, we can produce a custom user software or (user code).
Some requests have been included as features in the main software.

- Can the module perform as a data logger via the SDCard ?
Yes, this is being developed for SD card only. EEPROM can store up to 50 variables only as standard.
SD Card is the preferred method since customer can replace the card easily.
Customised firmware for writing to onboard 128M flash will be available.

-GUI systems usually require dynamic memory (malloc and free), which has to be handled very carefully in real-time systems. Are you using dynamic memory and if so, what system are you using?
The freeing of memory is a key issue.
We do have an equivalent to malloc and free.
Entities are created in the library and data area assigned.
When a DEL command (free) is used, the entry is marked
for deletion however the actual clean up is under user control
using the RESET(DELETED) command.

INTERFACING
-How many RS232 serial ports can this support?
There are 4 async ports as standard
A CMOS level 4 line ON CN3 and 2line on CN6
RS232 4 line on CN1 (includes DTR/DSR or CTS/RTS selection)
The K611A version modules include an additional full duplex RS485 port on CN1.

-What sort of USB is fitted - host or device?
Device 2.0 as Mass Storage Device or COM Port from Nov 30th
Access to SDHC 4G+ card as drive F:, H: etc.
HID Print and Keyboard will also be available later.

OTHER
- What is the response time of a keystroke?
The touch screen has 2 modes of operation.
1/ Calculate key on press or release allowing for defined debounce.
2/ Auto repeat after a defined period (delay) and resent until released (repeat).
This can be sent to a host interface or used internally for data entry or navigation.

-How is language support handled - is it possible to switch sets of strings for different languages?
Strings can be declared as variables and switched to any text object.

-On the current product we use a hybrid character set that is mostly western European but includes some central European characters, so we can use the same fonts for Polish.
How would different character sets be handled?
Unicode fonts are available for 90% of the world's languages.
You can request selected additions for Eu50.
The user can create their own image fonts on a paint software and reference them to UNICODE values

Getting Stared with iSMART TFTs
If you received a development kit with USB cable and SD card inserted into a xxx-K612A1TU module, just plug in the USB cable between a PC and the display module. The boot code and operational software will load and then run the file TU480A.mnu from the SD card.
The supplied demonstration sequences through 4 screens. The elevator and aircon screens are working applications so you can press the touch keyboard to operate. After 20 seconds of inaction, the demonstration moves on to the next screen.

After experimenting with the demonstration, review the basic applications below. Do not hesitate to send us an email for further explanation. Key issues to understand..

1/ The system uses text commands rather than difficult to remember hex codes.
2/ All objects and functions are given a name for easy future referencing.
Interfaces are given pre-defined names like RS2 for RS232 and RS4 for RS485.
3/ Commonly used parameters are stored in 'styles' like in HTML web pages.
This reduces the number of commands from 250 in a conventional TFT module
to just 25 in iSMART TFTs with equal or better functionality.

A typical menu file's commands will be constructed and ordered as follows (detail removed for clarity):

LIB... //load in images and fonts from memory into library
LIB...
INC.. //include another menu file which may have global styles and setup.

STYLE... //define styles for pages, text, images used in this file
STYLE...

SETUP.. //setup system and external interfaces like RS232
SETUP...

VAR... //create variables used for calculation, temporary storage and pointing
VAR...

PAGE(MAIN,styleMain) { //create a main page with text, images and associated keys
POSN... TEXT //place text at a specified position on screen
POSN... IMG //place icon / image at a specified position on screen
POSN... KEY //place a touch key area on screen and define function to call
}

PAGE(SUB,stylePage1) { //create other pages
.....
LOOP(CntLoop,FOREVER) { IF(CNTMINS=0,FncZero); } // function calls associated with page
}

FUNC(FncZero) { LOAD(RS2,"Hour Count = ",CNTHRS,"\\0A\\0D"); //send message to host via RS232
FUNC(MyFunc) { .....} //other functions associated with key press or interfaces

INT... // Initialise interrupts for slave timers and inputs...not host interface - use setup

SHOW(MAIN); // After pre-loading all style parameters, pages and functions, start the application with first page.
After this point, functionality follows page key presses and functions or incoming command data from host or interfaces.

When creating an entity for the first time, include the style parameter. To update the entity omit the style parameter.
If you specify the style again, you will create a copy.

Entities are layered on the screen from back to front in the order they are listed in the menu with the screen background defined in the page style.
If you want a button image to change colour, include one colour button in your background and the other colour button as a separate image over the top. To change colour, just HIDE and SHOW the top button. This technique is used in the air conditioner project.

The examples below can be cut and pasted from their box into a text editor (NotePad).
Save the file as TU480A.mnu and copy onto the SD card.
Plug it into the iSMART TFT module, apply power and view the result.

Hello World from Internal Menu

```
// Menu file TU480A.MNU for Demo using TU480X272C and v32 firmware update
// Simple demo to display text

STYLE(BlackPg, Page) { Back=black;} //black background
STYLE( Txt32White, Text )
{
font=Ascii32; col=white; maxLen=32; maxRows=1; curRel=CC; //white system text 32 pixels high
}

PAGE( MainPg, BlackPg )
{
POSN( 240, 136 ); // Set writing position to centre of display
TEXT( Text1, "Hello World", Txt32White ); // Draw text
}

SHOW( MainPg );
//end
```

Hello World via RS232

```
// Menu file TU480A.MNU for Demo using TU480X272C
// 07-Oct-2010
// This example is identical to example 1 except RS232 is defined
// using setup for command mode at19200 baud, no parity

STYLE(BlackPg, Page) //define page style
 {
 Back=black;  //background is black
 }

STYLE( Txt32White, Text ) //define text style
{
font=Ascii32; //use built in font
col=white;      //text colour is white
maxLen=32;
maxRows=1;
curRel=CC;      //centre position
}

PAGE( MainPg, BlackPg )
{
POSN( 240, 136 ); // Set writing position to centre of display
TEXT( Txt1, " ", Txt32White ); // Create empty text area at the writing position
}

SETUP( RS2 )
{
set = "192NC"; // 19200 bps, no parity, command mode
}

SHOW( MainPg );


// Send text command to the display via RS232 :  TEXT( Txt1, "Hello World" );;\\0D
// Note :-
// Text command does not include style so the named text is updated
// Sending 2 semicolons is equivalent to SHOW (currentpage);
// All command lines must be followed by CR (\\0D)
//If your system can send binary \\0D can be sent as 0DH
```

Images loaded, flashed and moved

```
// Menu file TU480A.MNU for Demo using TU480X272C
// 11-Oct-2010
// This example places 2 images on the display which are controlled by RS232
// using setup for command mode at 9600 baud, no parity.

LIB(Image1,"SDHC/lift1.bmp?back=\\0000CD");  //load image1 from SD card
LIB(Image2,"SDHC/lift2.bmp?back=\\0000CD");  //load image2 from SD card
STYLE(BluePg, Page) {back=\\0000CD;}    //define style of page with blue background
STYLE(StImg, Image) {curRel=CC;}         //centre image with respect to POSN cursor

PAGE( MainPg, BluePg )
{
POSN( 199, 136 ); IMG( LeftImg, Image1,172,240,StImg); // Position and draw 1st image on display
POSN( 396, 136 ); IMG( RightImg, Image2,172,240,StImg); // Position and draw 2nd image on the display
}

SHOW( MainPg );  //show page

WAIT(2000);        //wait 2 seconds
HIDE( LeftImg );;  //hide left image and refresh page
WAIT(2000);
SHOW( LeftImg );; //show left image and refresh page
WAIT(2000);
POSN( 396,136,LeftImg);;    //position left image under right image and refresh page
// Sending 2 semicolons is equivalent to SHOW (currentpage);
//You will see a blue border around the right image due to background transparency differences.
```