Freescale Semiconductor
Application Note

Document Number: AN4263
Rev. 0, 03/2011

Modification of the eGUI (D4D) Driver for char Displays (D4CD)

Simple Source Code User Guide

by: Petr Gargulak
Roznov pod Radhostem, Czech Republic

Contents

1 Introduction

This application note describes the modified version of the eGUI (D4D) for standard char displays, mainly for the HD44780 driver. The modification has been done because of the popularity of char displays and the effective structure of an eGUI that simplify the whole project organization with display interface. The result is a port of an eGUI source code from graphic LCD to char displays.

The overall structure of an eGUI completely complies with the new D4CD driver. There is only some reduced functionality like touch screen support and some new added like cursor support that is usually used in applications with char displays.

This is a short application note that helps developers who are working on new projects with char display to quickly understand the D4CD driver, but it does not describe the overall structure. The structure and main idea of the D4D/D4CD graphic high level drivers are described in the eGUI design reference manual titled *Freescale Embedded GUI (D4D)* (document DRM116).

2 Features

The D4CD brings to the project char LCD topology from the eGUI that helps create the application with a professional appearance and logic structure.

- Supports character LCD displays of various sizes
- Small RAM (volatile) memory footprint
- Multiple platform support
- Driver object style
- Smart support-screen-oriented structure of the user code
- Custom screen sizes, position, and a header like window
- Objects:
  - Button
  - Check box and user handled radio button
  - Label
  - Menu
  - Number
  - ListBox
- User defined objects:
  - Time
  - Date
  - Password
- HW cursor support
- Multiple user font tables support
- Buffer for input keys

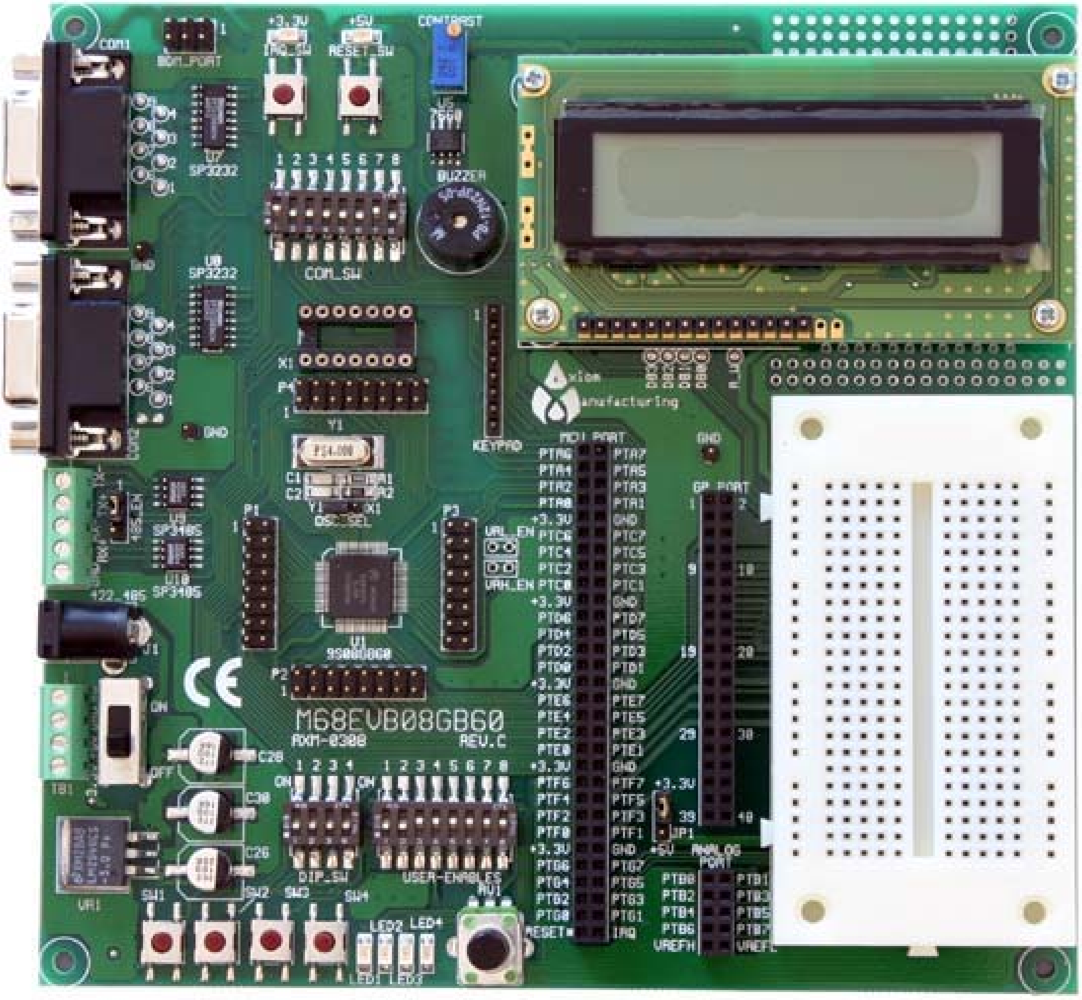


Figure 1. M68EVB08GB60 evaluation board with char display

3 The Main Changes

This chapter describes the main changes between the eGUI(D4D) and new created driver D4CD.

3.1 Source files

The source files organization of D4CD driver is the same as in the D4D, there are changes in some files:

- low_level_drivers — Contains new low level drivers to support low level driver needs (HD44780, and so on.)
- graphic_objects — Contains new and modified graphics adapted for char display
- common_files — Removed bitmap and color scheme source files

3.2 Features

There are a few changes in the graphic driver features that regard adaption to the D4CD for the display char organization. Some features have been removed because the char display does not use them.

Removed features:

- Touch screen support
- Bitmap and font decoders
- Color scheme support
- Orientation support

Added features:

- Hardware cursor support
- User char table support

3.3 Graphic object changes

The graphic objects have changed to reflect the capability of the char display. The list of objects is divided into two parts:

- Standard:
  - button
  - label
  - check box
  - menu
  - number
  - list box
- User defined:
  - time
  - date
  - password

The API of individual graphic objects is described in individual chapters.

3.4 Low level driver API changes

The low level drivers had to be changed to reflect different control and capabilities of the char display. The main idea and structure of low level drivers are the same as in the eGUI but the interface content has changed.

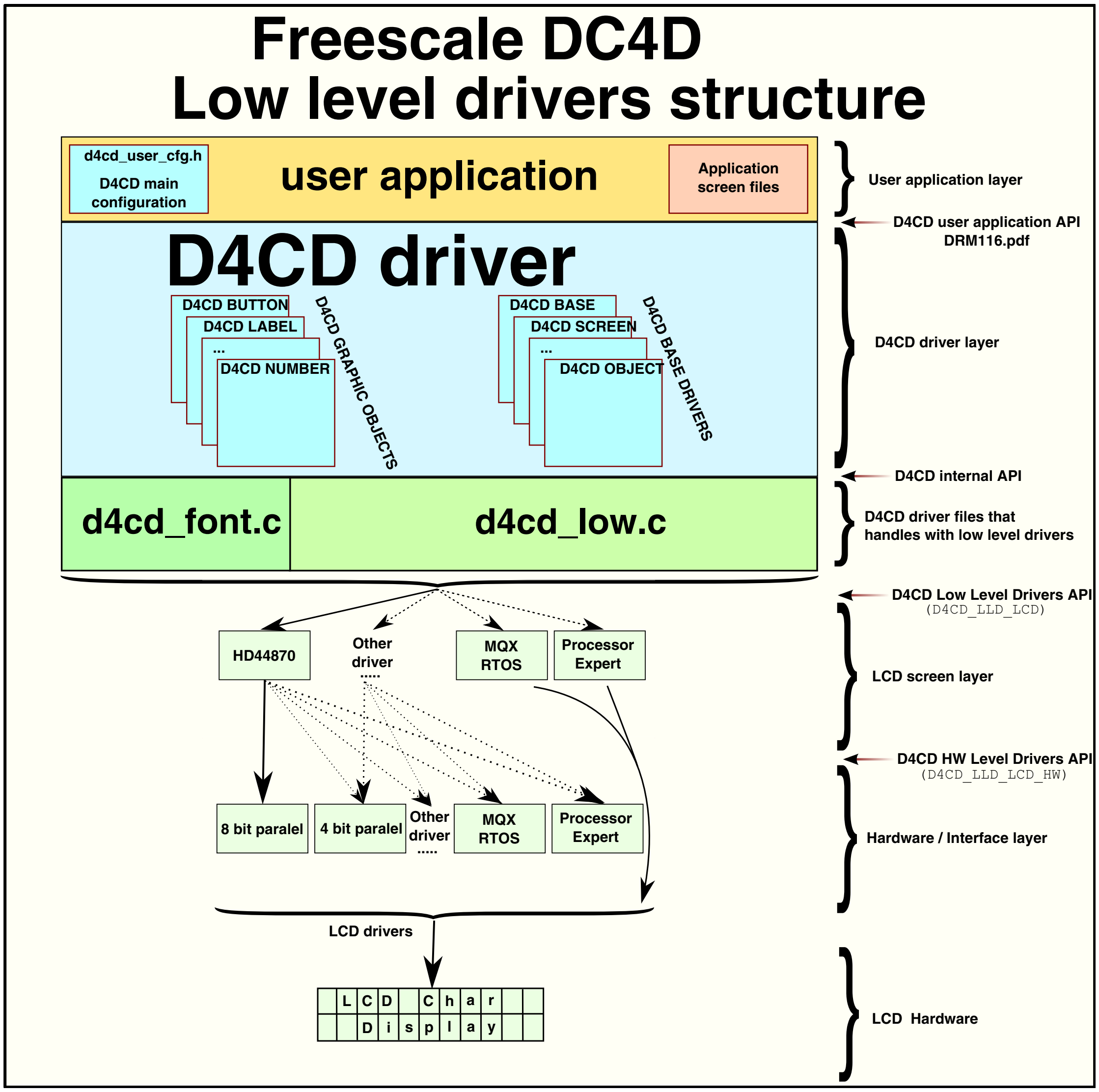


Figure 2. Low level driver structures

The LCD controller level API (application programming interface) structure content:

- D4CDLCD_Init — Standard init function
- D4CDLCD_GoTo — Function that sets the position of the hardware cursor
- D4CDLCD_SendChar — Function writes one character on the cursor position
- D4CDLCD_ReadChar — Function reads character on the cursor position
- D4CDLCD_WriteUserChar — Function writes the one char bitmap into the LCD driver user char memory
- D4CDLCD_SetCursor — Function sets the cursor appearance
- D4CDLCD_Delay_ms — Function calls simple delay
- D4CDLCD_Deinit — Standard de-initialization function

The LCD controller hardware interface level API structure content:

- D4CDLCDHW_Init — Standard init function
- D4CDLCDHW_SendData — Function sends one data byte
- D4CDLCDHW_SendCmd — Function sends one command byte
- D4CDLCDHW_SendNibbleCmd — Function sends one command in nibble mode
- D4CDLCDHW_ReadData — Function reads one data byte
- D4CDLCDHW_ReadCmd — Function reads one command byte
- D4CDLCDHW_PinCtl — Function controls the additional signals (RESET and BACKLIGHT)
- D4CDLCDHW_DeInit — Standard de-initialization function

For details check the source code in the d4cd_lldapi.h file.

4 Graphic Object Description

4.1 D4CD_BUTTON

The D4CD_BUTTON object is defined for standard button use—select and push. The button supports two types of visual aspects to demonstrate the state of the button (focused and unfocused), via the hardware cursor or the special printed characters on the sides of the button text (example: round and angle brackets).

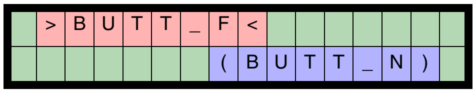


Figure 3. Button object

4.1.1 Declaration

D4CD_DECLARE_BUTTON (name, text, x, y, cx, flags, pUser, onClick, and pOnUsrMsg)

- name — Name of object
- text — Pointer to the button text
- x,y — Coordination of the button
- cx — Full length of button
- flags — System flags of object
- pUser — User data and pointer
- onClick — Pointer on OnClick function
- pOnUsrMsg — Pointer on the user message function

4.1.2 API functions

The only standard function to the runtime change text of the button—D4CD_SetText.

4.2 D4CD_LABEL

The D4CD_LABEL object is defined to demonstrate a one line text on the display. The main feature of the label object (extension to the original D4D) is the new fixed pretext and posttext portion of the label. This part allows creating the labels. There is a first and last part of the text constant, the middle can be changed. For example the preText and postText parts can be used to demonstrate the temperature–Temp: 26 °C. This label takes in only a RAM memory string of 26 and the remaining text can be in NVM (non-volatile memory).

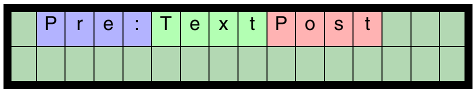


Figure 4. Button object

4.2.1 Declaration

D4CD_DECLARE_LABEL (name, preText, text, postText, x, y, cx, flags, pUser, and pOnUsrMsg)

- name — Name of the object
- preText — Pointer on the pretext part of the label
- text — Pointer to the main text of the label
- postText — Pointer to the post text part of the label
- x,y — Coordination of the label
- cx — Full length of the label
- flags — System flags of the object
- pUser — User data and pointer
- pOnUsrMsg — Pointer on the user message function

4.2.2 API functions

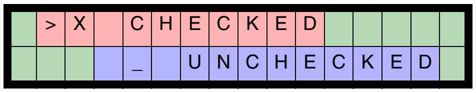
The only standard function to the runtime change text of the label — D4CD_SetText.

4.3 D4CD_CHECKBOX

The D4CD_CHECKBOX object is defined for standard check box use—select check and unchecked. The check box supports two types of visual aspects to demonstrate the state of the check box (focused and unfocused), via the hardware cursor or by the special printed character on the left side of the object (example: round and angle brackets). The object also allows setup of the characters for the checked and unchecked char.

Example:

- red — Checked or focused check box object
- blue — Unchecked and unfocused check box object



```
 >X CHECKED     
    _ UNCHECKED 
```

Figure 5. Check box objects

4.3.1 Declaration

D4CD_DECLARE_CHECKBOX (name, text, x, y, cx, flags, pUser, onchange, pOnUsrMsg)

- name — Name of the object
- text — Pointer to the text of the check box
- x,y — Coordination of the check box
- cx — Full length of the check box
- flags — System flags of the object
- pUser — User data and pointer
- onchange — Pointer on the OnChange function
- pOnUsrMsg — Pointer on the user message function

4.3.2 API functions

Standard function to runtime change text of check box—D4CD_SetText.

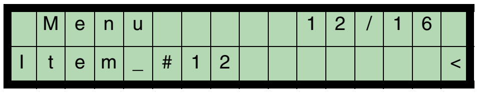
void D4CD_CheckBoxSetValue(D4CD_OBJECT_PTR pThis, D4CD_BOOL value) — Set the check box function that is used to set the state of the check box by runtime.

Byte D4CD_CheckBoxGetValue(D4CD_OBJECT_PTR pThis) — Get the check box function that is used to get the state of the check box by runtime.

4.4 D4CD_MENU

The D4CD_MENU object is defined for menu use—select the item and run action. The menu object is similar to the D4D_MENU in the eGUI, it has the same behavior including the title text and item index. The selection of the menu item is executed by the focus character on the right side of the object. The menu object can also be configured to two different visual modes:

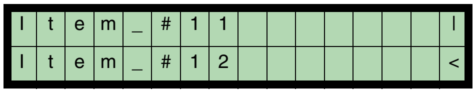
- With title and item index



```
 Menu     12/16 
Item_#12       <
```

Figure 6. Menu object with title

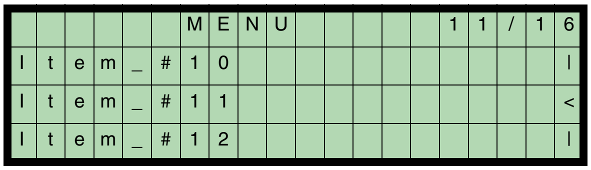
- Without title and item index



```
Item_#11       |
Item_#12       <
```

Figure 7. Menu object without title

- With title and item index (4 lines / 20 chars)



```
      MENU     11/16
Item_#10           |
Item_#11           <
Item_#12           |
```

Figure 8. Menu object with title (4 lines and 20 chars)

4.4.1 Declaration

The declaration is divided into three parts (begin, items, and end) to allow, declare the menu object with unspecified count of items. For more details check the eGUI design reference manual titled *Freescale Embedded GUI (D4D)* (document DRM116)

D4CD_DECLARE_MENU_BEGIN (name, title_text, x, y, cx, cy, flags, pUser, pOnClick, and pOnUsrMsg)

- name — Name of object
- title_text — Pointer to the text of title of the menu
- x,y — Coordination of the menu
- cx, cy — Size of the menu
- flags — System flags of the object
- pUser — User data and pointer
- pOnClick — Pointer on the pOnClick function
- pOnUsrMsg — Pointer on the user message function

D4CD_DECLARE_MENU_ITEM (text)

- text — pointer of item text

D4CD_DECLARE_MENU_ITEM_FULL(text, pUser)

- text — Pointer of the item text
- pUser q — Item user pointer

D4CD_DECLARE_MENU_END (name)

- name — Name of the object (must be the same as in the begin section)

4.4.2 API functions

void D4CD_MenuSetIndex (D4CD_OBJECT* pThis, D4CD_MENU_INDEX ix) — Function that sets the menu item index in runtime.

D4CD_MENU_INDEX D4CD_MenuGetIndex (D4CD_OBJECT* pThis) — Function that gets the menu item index in runtime.

D4CD_MENU_INDEX D4CD_MenuGetItemCount (D4CD_OBJECT_PTR pThis) — Function that gets the menu item counts in runtime.

D4CD_MENU_INDEX D4CD_MenuFindUserDataItem(D4CD_OBJECT_PTR pThis, void* pUser) — Function that tries to find the item index by the user data and pointer.

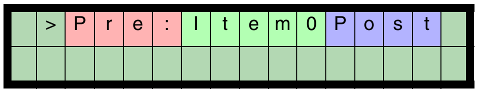
void* D4CD_MenuGetItemUserData(D4CD_OBJECT_PTR pThis) — Function that gets the menu item user data and pointer in runtime.

D4CD_CHAR* D4CD_MenuGetItemText(D4CD_OBJECT_PTR pThis) — Function that gets the menu item text pointer in runtime.

4.5 D4CD_LISTBOX

The D4CD_LISTBOX object is defined for standard list box use—select the item from a list. The list box object uses pre and post texts like the D4CD_LABEL to simplify creating the application screen with this object. The selection of the list box item is executed by a focus character on the left side of the object, the same as the D4CD_CHECKBOX uses.

List box example — red–pretext, green–item, blue–post text.



```
 >Pre:Item0Post 
```

Figure 9. Menu object

4.5.1 Declaration

The declaration is divided into three parts (begin, items, and end) to allow, declare the list box object with the unspecified count of items.

D4CD_DECLARE_LIST_BOX_BEGIN (name, preText, postText, x, y, cx, flags, pUser, pOnValch, and pOnUsrMsg)

- name — Name of the object
- preText — Pointer to the text that is printed before the list box item
- postText — Pointer to the text that is printed after the list box item
- x,y — Coordination of the list box
- cx — Full length of the list box object
- flags — System flags of the object
- pUser — User data and pointer
- pOnValch — Pointer on the pOnValch function
- pOnUsrMsg — Pointer on the user message function

D4CD_DECLARE_LIST_BOX_ITEM (pItemText, pItemUserPointer)

- pItemText — Pointer of the item text
- pItemUserPointer — Item user pointer

D4CD_DECLARE_LIST_BOX_END

4.5.2 API functions

void D4CD_ListBoxSetIndex(D4CD_OBJECT_PTR pThis, D4CD_LIST_BOX_INDEX index) — Function that sets the list box item index in runtime.

D4CD_LIST_BOX_INDEX D4CD_ListBoxGetIndex(D4CD_OBJECT_PTR pThis) — Function that gets the list box item index in runtime.

D4CD_LIST_BOX_INDEX D4CD_ListBoxGetItemCount(D4CD_OBJECT_PTR pThis) — Function that gets the list box item counts in runtime.

D4CD_LIST_BOX_INDEX D4CD_ListBoxFindUserDataItem(D4CD_OBJECT_PTR pThis, void* pUser) — Function that tries to find the item index by the user data and pointer.

void* D4CD_ListBoxGetItemUserData(D4CD_OBJECT_PTR pThis) — Function that gets the list box item user data and pointer in runtime.

D4CD_CHAR* D4CD_ListBoxGetItemText(D4CD_OBJECT_PTR pThis) — Function that gets the list box item text pointer in runtime.

4.6 D4CD_NUMBER

The D4CD_NUMBER object is defined to demonstrate and set the number with or without a decimal point. The number object is using pre and post texts like the D4CD_LABEL to simplify creating the application screen with this object.

Number example — red–pretext, green–number, blue–post text.

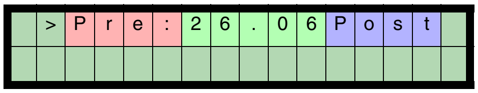


Figure 10. Number object

4.6.1 Declaration

D4CD_DECLARE_LIST_BOX_BEGIN (name, preText, postText, x, y, cx, flags, pUser, pOnValch, and pOnUsrMsg)

- name — Name of the object
- preText — Pointer to the text that is printed before the list box item
- postText — Pointer to the text that is printed after the list box item
- x,y — Coordination of the list box
- cx — Full length of the list box object
- flags — System flags of the object
- pUser — User data and pointer
- pOnValch — Pointer on the pOnValch function
- pOnUsrMsg — Pointer on the user message function

4.6.2 API functions

void D4CD_NumberSetValue(D4CD_OBJECT_PTR pThis, D4CD_NUMBER_VALUE value) — Function that sets the value of the number object in runtime.

void D4CD_NumberChangeValue(D4CD_OBJECT_PTR pThis, sWord incr) — Function that changes the value of the number object in runtime.

D4CD_NUMBER_VALUE D4CD_NumberGetValue(D4CD_OBJECT_PTR pThis) — Function that gets the value of the number object in runtime.

void D4CD_NumberSetLimits(D4CD_OBJECT_PTR pThis, D4CD_NUMBER_LIMITS* pLimits) — Function that set the limits of the number object.

4.7 D4CD_TIME

The D4CD_TIME user defined object is defined to demonstrate and set the time. The time object can demonstrate the hours, minutes, and seconds, all these items can be individually displayed or hidden in any combination. The object also supports setting the time with undo operation during setting phase.

Time example — red–hours, green–minutes, blue–seconds

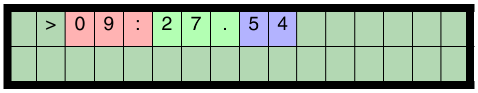


Figure 11. Number object

4.7.1 Declaration

D4CD_DECLARE_TIME (name, x, y, cx, flags, pUser, OnSet, and pOnUsrMsg)

- name — Name of the object
- x,y — Coordination of the time
- cx — Full length of the time object
- flags — System flags of the object
- pUser — User data and pointer
- OnSet — Pointer on the OnSet function
- pOnUsrMsg — Pointer on the user message function

4.7.2 API functions

void D4CD_TimeSetValue(D4CD_OBJECT_PTR pThis, Byte hours, Byte minutes, Byte seconds) — Function that sets the time value in runtime.

void D4CD_TimeAddSecond(D4CD_OBJECT_PTR pThis) — The help function that simplifies adding a new second increment time by one second.

D4CD_TIME_VALUE* D4CD_TimeGetValue(D4CD_OBJECT_PTR pThis) — Function that returns the current value of the time object in runtime.

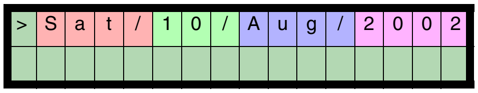
void D4CD_TimeSetVisibility(D4CD_OBJECT_PTR pThis, D4CD_BOOL hours, D4CD_BOOL minutes, D4CD_BOOL seconds) — Function that sets the visibility of the individual items of the time object.

void D4CD_TimeSetMode(D4CD_OBJECT_PTR pThis, D4CD_BOOL enable24mode) — Function that sets the mode (12/24) of the time object.

4.8 D4CD_DATE

The D4CD_DATE user defined object is defined to demonstrate and set the date. The date object can demonstrate the day, date, month, and year, all these items can be individually displayed or hidden in any combination. The object also supports setting the date with undo operation during the setting phase.

Date example — red –day, green–date, blue–month, magenta–year



>	S	a	t	/	1	0	/	A	u	g	/	2	0	0	2

Figure 12. Date object

4.8.1 Declaration

D4CD_DECLARE_DATE (name, x, y, cx, pDayNames, pMonthNames, flags, pUser, onSet, and pOnUsrMsg)

- name — Name of the object
- x,y — Coordination of the date
- cx — Full length of the date object
- pDayNames — Pointer on the strings of the days
- pMonthNames — Pointer on the strings of the months
- flags — System flags of the object
- pUser — User data and pointer
- OnSet — Pointer on the OnSet function
- pOnUsrMsg — Pointer on the user message function

4.8.2 API functions

void D4CD_DateSetValue(D4CD_OBJECT_PTR pThis, Byte day, Byte date, Byte month, Word year) — Function that sets the date value in runtime.

D4CD_DATE_VALUE* D4CD_DateGetValue(D4CD_OBJECT_PTR pThis) — Function that returns the current value of the date object in runtime.

void D4CD_DateSetVisibility(D4CD_OBJECT_PTR pThis, D4CD_BOOL day, D4CD_BOOL date, D4CD_BOOL month, D4CD_BOOL year) — Function that sets the visibility of individual items of the date object.

4.9 D4CD_PASSWORD

The D4CD_PASSWORD user defined object is defined to provide password functionality into embedded applications. The password object support variable length of password and three groups of characters that can be used (alphabet, numbers and special chars).

Date example — red–pretext, green–password, blue–postText.

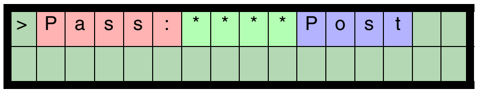


Figure 13. Password object

4.9.1 Declaration

D4CD_DECLARE_PSWRD(name, preText, postText, x, y, pswdLen, flags, pUser, pOnEnter, and pOnUsrMsg)

- name — Name of the object
- preText — Pointer to the text that is printed before the password
- postText — Pointer to the text that is printed after the password
- x,y — Coordination of the password object
- pswdLen — Password length
- flags — System flags of the object
- pUser — User data and pointer
- pOnEnter — Pointer on the pOnEnter function
- pOnUsrMsg — Pointer on the user message function

4.9.2 API functions

void D4CD_PswrdSet(D4CD_OBJECT_PTR pThis, D4CD_PSWRD_VALUE* pValue) — Function that sets the password value.

D4CD_PSWRD_VALUE* D4CD_PswrdGetValue(D4CD_OBJECT_PTR pThis) — Function returns the current value of the password.

void D4CD_PswrdSetRange(D4CD_OBJECT_PTR pThis, D4CD_BOOL num, D4CD_BOOL chars, D4CD_BOOL special) — Function that sets the range of password.

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals”, must be validated for each customer application by customer’s technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics as their non-RoHS-compliant and/or non-Pb-free counterparts. For further information, see http://www.freescale.com or contact your Freescale sales representative.

For information on Freescale’s Environmental Products program, go to http://www.freescale.com/epp.