

TP : INFORMATIQUE INDUSTRIELLE

TPMP3

Acquisition et exploitation de la température d'eau par le boitier

1) PRESENTATION GENERALE DU RESEAU NAVIGATION

Le système de navigation se décompose en sept sous-systèmes :

- Système de navigation basique,
- Système de données NAV,
- Système de radionavigation,
- Système de visualisation des données de navigation,
- Système environnement air et eau,

Un schéma synoptique de système de navigation est représenté ci-dessous

2) Le système environnement air et eau

Le système environnement air et eau se compose :

- D'un réseau (Environmental Air & Water System),
- D'un ensemble de bathymétrie.

Le réseau est un système électronique permettant la mesure et l'élaboration de la situation météo. Ses fonctions essentielles sont :

3) ARCHITECTURE DU RESEAU

Le réseau d'acquisition des paramètres météo et de traitement du vent se compose de 6 sous-ensembles qui se répartissent en 4 catégories.

- Le coffret Répétiteur (séquencement des capteurs et visualisation des mesures)
- Les Capteurs Anémomètre Girouette (mesure de la vitesse et de la direction du vent)
- Le Capteur PTU (mesure de la pression, de la température, et de l'humidité de l'air)
- Le convertisseur Analogique/Digital (mesure la température de l'eau) équipé de sa sonde PT100.

3.1) LE COFFRET REPETITEUR

Le répétiteur assure l'acquisition des données des capteurs, leur traitement, et l'élaboration d'informations nécessaires à la distribution des mesures à travers le bâtiment.

Ce sont les seuls équipements qui assurent la gestion du bus RS485.

Le répétiteur réalise 3 traitements distincts :

- La détermination du vent « relatif » manuel
- La détermination du vent « relatif » automatique
- La détermination du vent « vrai » (dit « absolu »)

En fonctionnement normal, le répétiteur interroge l'ensemble des capteurs MODBUS.
Il récupère en Multicast, via sa liaison Ethernet, les informations CAP et SOG du bâtiment venant du bus navigation et permettant de calculer le vent vrai.
A la mise sous tension du réseau, le répétiteur « Maître » séquence le réseau et le mode de fonctionnement est celui mémorisé en EEPROM.

Fonctions réalisées par l'équipement

- Gestion du bus RS485 du réseau de vent numérique
- Acquisition des mesures des capteurs
- Visualisation des mesures en face avant sous forme numérique et pseudo analogique
- Sélection automatique du réseau optimal

- Elaboration des réseaux de vent absolu
- Gestion de la liaison Ethernet en UDP
- Gestion de la seconde liaison série RS 422
- Réglage de luminosité

Ce coffret comporte une unité centrale équipé d'un microcontrôleur 32 bits assurant la gestion des trois liaisons de communication, le traitement de l'information, et le réglage des paramètres spécifiques de fonctionnement de l'appareil.
L'accès aux modifications du paramétrage est réalisé à partir d'un micro-ordinateur de type PC à travers une interface RS232/RS485

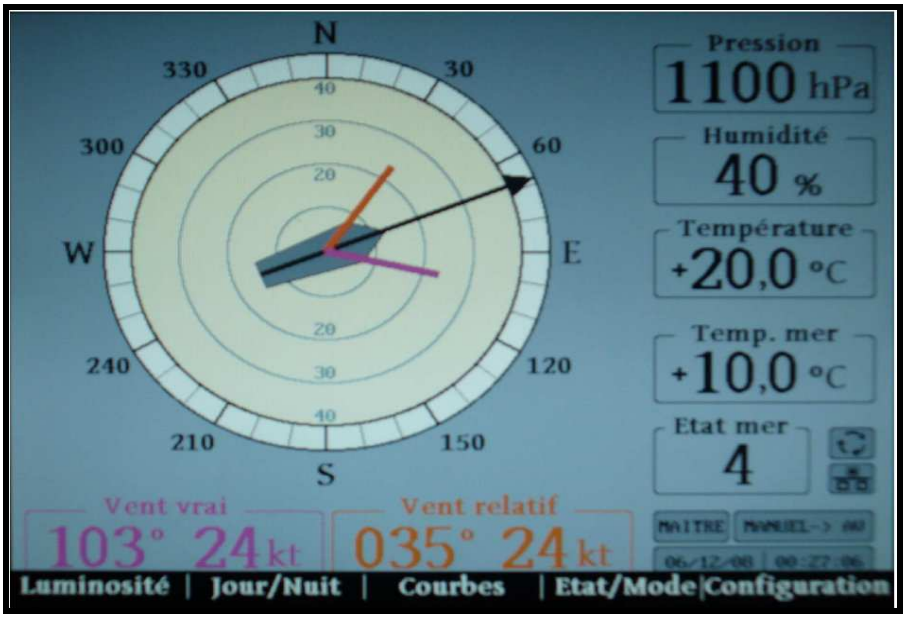
Caractéristiques d'affichage

- Affichage numérique de la vitesse
- Affichage numérique de la direction
- Affichage pseudo analogique de la vitesse et de la direction du vent
- Affichage numérique CAP / SOG
- Affichage numérique Pression / Température air
- Affichage numérique Hygrométrie
- Affichage numérique Température mer
- Affichage numérique de l'Etat de la mer

Liaisons numériques

- 1 liaison RS485 protocole MODBUS RTU maître (bus interne capteurs)
- 1 liaison RS422 protocole NMEA183
- 1 liaison Ethernet protocole UDP

L'interface homme machine



3.2) LE COFFRET CONVERTISSEUR

Le coffret est constitué d'un module incorporé dans un boîtier étanche.
Il permet la mesure de 3 voies analogiques dont une seule servira à notre application.
Cette unique voie assure la mesure de la température de l'eau de mer.
Ce capteur comporte une unité centrale à microcontrôleur assurant l'acquisition des trois mesures et leur traitement, ainsi que le réglage et la modification des paramètres spécifiques de fonctionnement de l'appareil.
L'accès aux modifications du paramétrage est réalisé à partir d'un micro-ordinateur de type PC à travers une interface RS232/RS485. Le format des trames est le suivant :

Interrogation du boîtier MEA3

Message d'interrogation du Répétiteur

07	04	00	00	00	03	CRC	CRC

Réponse du boîtier MEA3

			Voie 1		Voie 2		Voie 3			
07	04	06	xx	xx	xx	xx	xx	xx	CRC	CRC

Avec

Information	Nb octets	Codage	Bits utilisés	Fin d'échelle	Résolution
Voie 1 (Température eau mer)	2	binaire naturel	10 bits	59,99 °C	0.058 °C
Voie 2 (inutilisée)	2	*****	10 bits	*****	*****
Voie 3 (inutilisée)	2	*****	10 bits	*****	*****

Acquisition et exploitation de la température d'eau

1) But

A Partir d'une carte à base de processeur ARM, acquérir, traiter et afficher la température de l'eau

2) Objectifs : L'étudiant devra être capable :

-D'exploiter la documentation sur les Processeurs ARM afin de comprendre les mécanismes matériels et logiciels permettant d'acquérir la température de l'eau de mer du coffret MEA3C et d'afficher cette température

- De s'initier à une suite de développement logiciel µvision4 permettant de développer un applicatif pour un processeur ARM dans un contexte embarqué

-De monopoliser ses connaissances en langage C afin de répondre à l'application envisagée

3) Prérequis

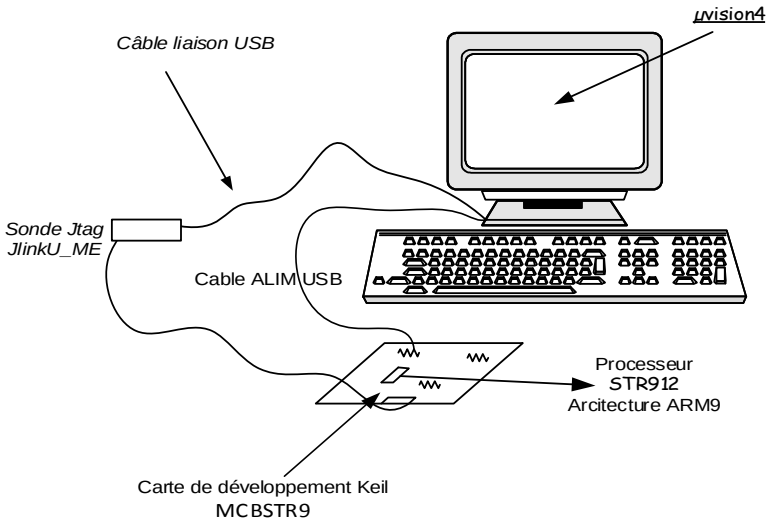
Connaissance de la documentation sur les Processeurs ARM et notamment le STR912Fx

Lecture du fascicule mode d'emploi de la suite logicielle µvision4.

Cour sur les convertisseurs analogiques numériques et le langage C.

Dossier technique sur le réseau navigation

Schéma de câblage



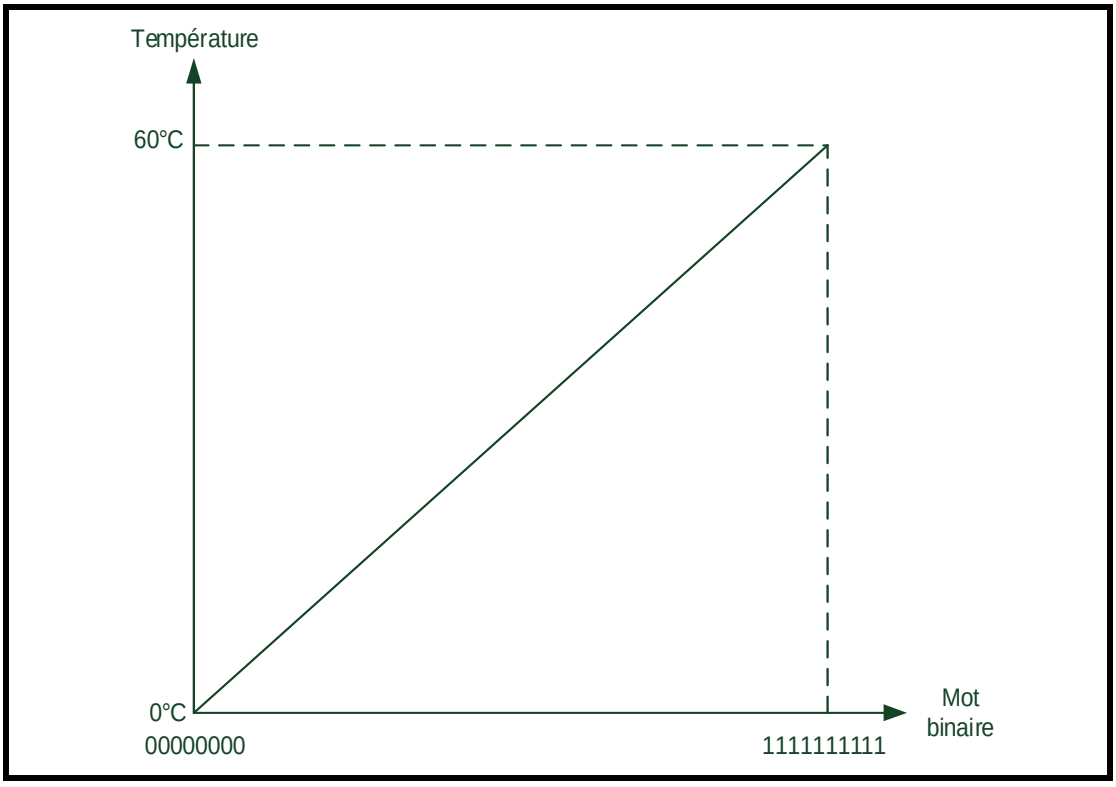
4) Présentation

On souhaite émuler le coffret NMEA3C. On suppose que ce coffret implémente une carte de développement de la société Keil intégrant notamment un processeur STR912 avec une architecture d'ARM9

Un potentiomètre sur la carte nous permettra de simuler la température de l'eau de mer et la tension analogique image de la température sera connectée sur la ligne 0 du Port4

L'information température sera codée sur 2 octet (10 bits exactement) :

La courbe qui donne la température en fonction du mot binaire est donnée ci-dessous :



Un afficheur LCD nous permettra de visualiser la température

5) Questionnaire

Dans un premier temps on souhaite faire l'acquisition de la température. l'outil intégré de débogage de la carte nous permettra de visualiser simplement cette température. A chaque fin de conversion une interruption sera générée afin de mettre à jour la variable temperature.

5.1) Première partie : Questions technique et d'analyses fonctionnelle

5.1.1) Quels sont les principaux périphériques qu'intègrent le processeur STR912 Ainsi que sa capacité en mémoire Vive et morte

5.1.2) Le cœur de processeur du STR912 a une architecture du type ARM9, Quels sont les principales caractéristiques de ce cœur, quelles améliorations lui a-t-on apportées.

5.1.3) Quel type de liaison électrique a-t-on adopté entre le coffret MEA3C et le répétiteur. Quel protocole de bas niveau véhicule cette information.

5.2) Développement du programme d'acquisition de la température de l'eau de mer

5.2.1) ouvrir le projet MP2.uvproj, en cliquant sur le fichier STR91x.s.
Ouvrir l'onglet configuration wizard et clock configuration. En déduire le temps de conversion du CAN en sachant que le prédiviseur du CAN est réglé sur 255.

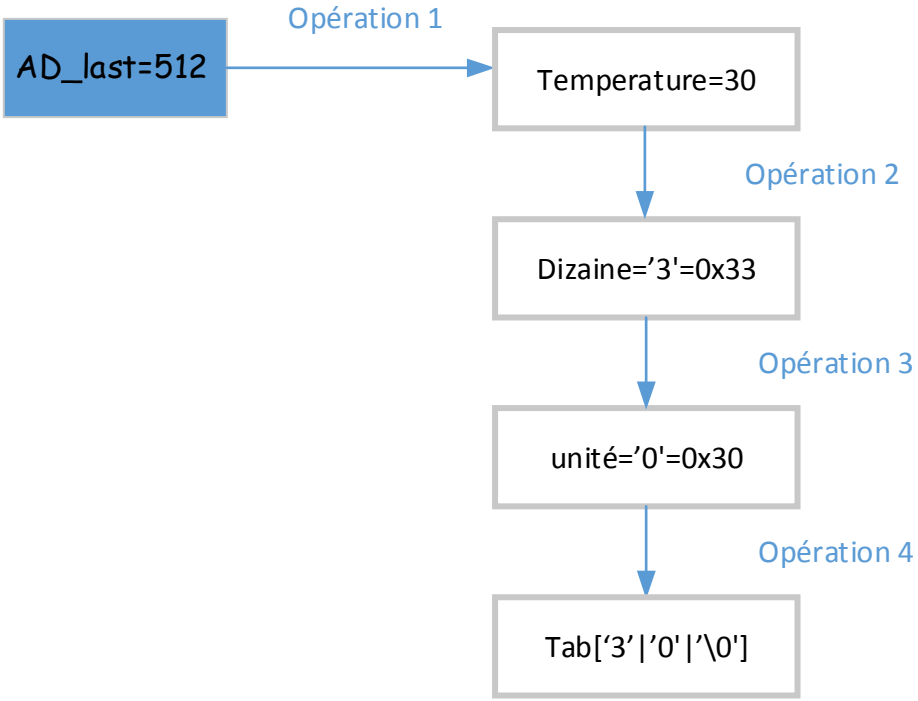
$$FPLL = \frac{2 \times N \times FOSC}{(M \times 2^{P})}$$

5.2.2) On donne document réponse N°1, le code qui nous permet de mettre en œuvre le CAN. En vous servant du fascicule processeur ARM, complétez les lignes de code manquantes. La suite des instructions suit l'algorigramme page N°3 du fascicule processeur ARM

5.2.3) On donne document réponse N°2 les lignes de code qui permettent la mise en œuvre du module d'interruption. Commentez ces lignes de code

5.2.3) Compilez votre programme et vérifiez si la variable AD_last varie (on pourra utiliser les points d'arrêt.

La deuxième partie va nous permettre d'exploiter la variable température.
L'exemple suivant doit nous permettre de comprendre le traitement à réaliser sur la donnée
Supposant que la température se trouve en milieu de gamme 30°C



Proposez un programme dans la boucle sans fin de la fonction main nous permettant de réaliser ces 4 opérations.
Vérifiez en faisant varier le curseur, si le tableau Tab contient sous forme de chaine de caractère ASCII la valeur de la température.
Pour la dernière partie on souhaite afficher la température sur l'afficheur LCD.
Pour se servir de l'afficheur LCD de la carte Keil MCBSTR9, on aura besoin de programmer les ports d'entrée- sorties avec les instructions suivantes :

```
SCU->GPIOOUT[7]  = 0x5555;          /* P7.0..7 output - mode 1        */
 GPIO7->DDR      = 0xFF;            /* P7.0..7 Outputs (LED Data)     */
GPIO7->DR[0x3FC]=0x00;
 /* LCD Setup                                                         */
 GPIO8->DDR      = 0xFF;            /* P8.0..7 Outputs (LCD Data)     */
 GPIO9->DDR      = 0x07;            /* P9.0..2 Outputs (LCD Control)  */
```

On donne dans le dossier annexe les fonctions de haut niveau de l'afficheur LCD Fichier LCD.c. Proposez les modifications logicielles de notre programme afin d'afficher la valeur de la température. Compilez et testez votre programme.

Document réponse N°1

Programme ADC.C

```
int main (void) {
 unsigned int i, n;
  unsigned short AD_old, AD_value;
short temperature;

          char tab[3];

  /* ADC Setup                                                */
  SCU->GPIOIN[4]  |= _____________________;            /* P4.0 input  - mode 0          */
  SCU->GPIOOUT[4] &= _____________________;            /* P4.0 output - mode 0          */
  GPIO4->DDR      &= _____________________;            /* P4.0 direction - input        */
  SCU->GPIOANA    = _____________________________;     /* P4.0 analog mode ON           */

  ADC->CR         |= _____________________________;     /* Set POR bit  ADC WORKS        */
  for (n = 0; n < 100000; n ++);          /* Wait > 1 ms  (at 48 MHz)      */

  ADC->CR         &= _____________________________;     /* Clear STB bit0: Idle mode. The analog block is kept powered on  */
  for (n = 0; n < 1500; n ++);            /* Wait > 15 us (at 48 MHz)      */

  ADC->CR         |= _____________________________;     /* Enable end of conversion interupt 1: ECV interrupt enabled*/
  ADC->CCR        =_____________________________;      /* AD Conversion, No WDG choix du  Channel 0   */

while (1) {                       /* Loop forever                  */

 if (AD_value != AD_last) {               /* Read AD_last value            */
        i=0;

AD_value = AD_last;               /* Read AD_last value            */
                              }

ADC->CR |= 0x0403;                /* Set STR bit (Start Conversion)    */
                             /* on Channel 0, with interrupt      */
        }

 }
```

Programme IRQ.C

```
#include <91x_lib.h>


/* Global variable to be used in main program                       */
short AD_last;                    /* Last AD value read in interrupt   */

/* Variable used as semaphore, needed because TIM3 has higher priority than
   ADC interrupt and could restart the AD conversion that has not still
   finished                                                         */
unsigned char AD_in_progress = 0;       /* AD conversion in progress flag    */


__irq void ADC_IRQ_Handler (void) {     /* AD converter interrupt routine    */

  AD_last = ADC->DR0 & 0x03FF;          /* echantillon récupéré AD value for global usage (10 bit) */

  ADC->CR &= 0xFFFE;                /* Clear STR bit (Start Conversion)  */
```

```
ADC->CR &= 0x7FFF;              /* Clear End of Conversion flag      */
 VIC0->VAR = 0;                 /* Acknowledge Interrupt             */
VIC1->VAR = 0;
```

Document réponse N°2

```
VIC0->VAiR[15]  = (unsigned int)ADC_IRQ_Handler; /* ____________________________________________________________*/
 VIC0->VCiR[15] |= 0x20;            /* ____________________________________________________________ */
 VIC0->VCiR[15] |= 15;              /* ____________________________________________________________ */
 VIC0->INTER    |= (1<<15);         /* ____________________________________________________________ */
ADC->CR |= 0x0403;                  /* ____________________________________________________________ */
                                    /* ____________________________________________________________ */
```

Annexe

```
/*----------------------------------------------------------------------------*/
/* LCD.c: Functions for 2 line 16 character Text LCD with ST7066 controller   */
/*----------------------------------------------------------------------------*/
/* This file is part of the uVision/ARM development tools.                    */
/* Copyright (c) 2005-2006 Keil Software. All rights reserved.                */
/* This software may only be used under the terms of a valid, current,        */
/* end user licence from KEIL for a compatible version of KEIL software       */
/* development tools. Nothing else gives you the right to use this software.  */
/*----------------------------------------------------------------------------*/

#include <91x_lib.h>

/* 8 characters                                                               */
extern const unsigned char UserFont[8][8];


/*------------------------- Text LCD size definitions ------------------------*/

/* Width (in character) controlled by ST7066 controller                       */
#define LineLen    16

/* Hight (in lines) controlled by ST7066 controller                           */
#define NumLines    2

/*------------------------- Text LCD size definitions ------------------------*/

#define LCD_DATA    GPIO8->DR[0x3FC]  /* Data bits D0 = P8.0 .. DB7 = P8.7  */
#define LCD_E       GPIO9->DR[0x004]  /* Enable control                     */
#define LCD_RW      GPIO9->DR[0x008]  /* Read/Write control                 */
#define LCD_RS      GPIO9->DR[0x010]  /* Data/Instruction control           */
#define LCD_CTRL    GPIO9->DR[0x01C]  /* All 3 control lines (E/RW/RS)      */

#define LCD_DISABLE 0x00
#define LCD_ENABLE  0x01
#define LCD_WRITE   0x00
#define LCD_READ    0x02
#define LCD_INST_REG 0x00
#define LCD_DATA_REG 0x04


/*------------------------ Global function definitions -----------------------*/


/*----------------------------------------------------------------------------
* Delay in while loop cycles                                                  *
*   Parameter:    cnt:    number of while cycles to delay                     *
*   Return:                                                                   *
*----------------------------------------------------------------------------*/

static void delay (int cnt)
{
  while (cnt--);
}


/*----------------------------------------------------------------------------
* Read status of LCD controller (ST7066)                                      *
*   Parameter:    c:      command to be written                               *
*   Return:       Status byte contains busy flag and address pointer          *
*----------------------------------------------------------------------------*/
```

```
static unsigned char lcd_read_status (void)
{
  unsigned char status;

  GPIO8->DDR = 0x00;                /* P8.0..7 Inputs (LCD Data)        */
  LCD_RW     = LCD_READ;
  LCD_RS     = LCD_INST_REG;        /* Instruction                      */
  LCD_E      = LCD_ENABLE;
  delay(10);
  status     = LCD_DATA;
  LCD_E      = LCD_DISABLE;
  delay(10);
  return (status);
}


/*****************************************************************************
* Wait until LCD controller (ST7066) busy flag is 0                          *
*   Parameter:                                                               *
*   Return:      Status bye of LCD controller (busy + address)               *
*****************************************************************************/

static unsigned char wait_while_busy (void)
{
  unsigned char status;

  do  {
    status = lcd_read_status();
  }  while (status & 0x80);          /* Wait for busy flag               */

  return (status);
}


/*****************************************************************************
* Write command to LCD controller (ST7066)                                   *
*   Parameter:    c:      command to be written                              *
*   Return:                                                                  *
*****************************************************************************/

void lcd_write_cmd (unsigned char c)
{
  wait_while_busy();

  GPIO8->DDR = 0xFF;                /* P8.0..7 Outputs (LCD Data)       */
  LCD_RW     = LCD_WRITE;
  LCD_RS     = LCD_INST_REG;        /* Instruction                      */
  LCD_E      = LCD_ENABLE;
  LCD_DATA   = c;
  delay(10);
  LCD_E      = LCD_DISABLE;
  delay(10);
}
```

```
/*****************************************************************************
* Write data to LCD controller (ST7066)                                      *
*   Parameter:    c:      data to be written                                 *
*   Return:                                                                  *
*****************************************************************************/

static void lcd_write_data (unsigned char c)
{
  GPIO8->DDR = 0xFF;                /* P8.0..7 Outputs (LCD Data)       */
  LCD_RW     = LCD_WRITE;
  LCD_RS     = LCD_DATA_REG;        /* Data                             */
  LCD_E      = LCD_ENABLE;
  LCD_DATA   = c;
  delay(10);
  LCD_E      = LCD_DISABLE;
  delay(10);
}


/*****************************************************************************
* Print Character to current cursor position                                 *
*   Parameter:    c:      character to be printed                            *
*   Return:                                                                  *
*****************************************************************************/

void lcd_putchar (char c)
{
  wait_while_busy();
  lcd_write_data (c);
}


/*****************************************************************************
* Initialize the ST7066 LCD controller                                       *
*   Parameter:                                                               *
*   Return:                                                                  *
*****************************************************************************/

void lcd_init (void)
{
  int i;
  unsigned char const *p;

  /* Reset controller                                                       */
  lcd_write_cmd(0x38);              /* Function set: 8bit / 2 lines / 5x8 */
  lcd_write_cmd(0x0F);              /* Display control: Disp/Curs/Blnk=ON */
  lcd_write_cmd(0x01);              /* Display clear                      */
  lcd_write_cmd(0x06);              /* Entry mode: Move right, no shift   */

  /* Load user-specific characters into CGRAM                               */
  lcd_write_cmd(0x40);              /* Set CGRAM address counter to 0     */
  p = &UserFont[0][0];
  for (i = 0; i < sizeof(UserFont); i++, p++)  {
    lcd_putchar (*p);
  }
  lcd_write_cmd(0x80);              /* Set DDRAM address counter to 0     */
}
```

```
/*****************************************************************************
* Clear the LCD display                                                      *
*   Parameter:                                                               *
*   Return:                                                                  *
*****************************************************************************/

void lcd_clear (void)
{
  lcd_write_cmd(0x01);                /* Display clear                  */
}


/*****************************************************************************
* Set cursor position on LCD display                                         *
*   Parameter:    column: column position 0-15                               *
*                 line:   line position 0-1                                  *
*   Return:                                                                  *
*****************************************************************************/

void set_cursor (unsigned char column, unsigned char line)
{
  unsigned char address;

  address = (line * 40) + column;
  address = 0x80 + (address & 0x7F);
  lcd_write_cmd(address);             /* Set DDRAM address counter to 0     */
}


/*****************************************************************************
* Print sting to LCD display                                                 *
*   Parameter:    string: pointer to output string                           *
*   Return:                                                                  *
*****************************************************************************/

void lcd_print (unsigned char const *string)
{
  while (*string)  {
    lcd_putchar (*string++);
  }
}
```