IRTC R/W Battery Back-up RAM

The IRTC module integrates 32 bytes Battery Back-up RAM. This example shows usage of the driver's macros for reading and writing data from/to the IRTC's Battery Back-up RAM. At 10 ms rate the `counter_rtc` variable is incremented and saved into IRTC's Battery Back-up RAM. In the main software loop the content of the IRTC's Battery Back-up RAM is read and stored in the `counter_ram` variable. The FreeMASTER communicates via UART2 of the MKM34Z256VLx7 device which is configured for 9600/8-N-1 and serviced by application software in a polling mode. When using TWR-KM34Z75M board, connect USB port of your PC to the on-board OpenSDA's USB to serial bridge (J27).

Source code:

```
/*****************************************************************************
 * (c) Copyright 2010-2015, Freescale Semiconductor Inc.
 * ALL RIGHTS RESERVED.
 *****************************************************************************
 * irtcram_test.c
 *****************************************************************************/
#include "drivers.h"

/* LPTMR callback function declaration                                       */
static void lptmr_callback (void);

/* static data definition                                                    */
static uint32 counter_rtc;
static uint32 counter_ram;

void main (void)
{
  /* enable peripheral clocks                                                */
  SIM_EnableModule (LPTMR);
  SIM_EnableModule (UART2);
  SIM_EnableModule (PORTI);

  /* enable low voltage reset detection (1.6V), disable interrupts and bandgap*/
  PMC_Init (PMC_MODULE_LVDRE_ON_LVDINT_OFF_LVWINT_OFF_CONFIG(PMC_LVDL,PMC_LVW1),
            PMC_INTREG_BGEN_OFF_BGBE_OFF_CONFIG, PRI_LVL0, (PMC_CALLBACK)NULL);

  /* initialize UART and FreeMASTER                                          */
  PORT_Init (PORTI, PORT_MODULE_ALT2_MODE, PIN6|PIN7);
  UART_Init (UART2, UART_MODULE_POLLMODE_CONFIG(9600,1e6));
  FMSTR_Init();

  /* initialize LPTMR in counter reset mode                                  */
  LPTMR_Init(LPTMR_MODULE_TMR_CNT_RST_ONTCF_MODE_CONFIG(0,LPTMR_LPOCLK),5);
  LPTMR_InstallCallback (PRI_LVL1, lptmr_callback);
  EnableInterrupts();     /* enable interrupts on global level               */
  LPTMR_Enable();         /* enable LPTMR                                    */

  while (1)
  {
    FMSTR_Poll ();
    /* read variable from RTC register file - takes 75 us @ 2 MHz core clock  */
    IRTC_RdRam ((uint8*)&counter_ram, sizeof(counter_ram));
  }
}

/* LPTMR interrupt - called every 10 ms                                      */
static void lptmr_callback (void)
{
  counter_rtc++;          /* increment variable                              */
  /* store variable in RTC register file - takes 143 us @ 2 MHz core clock    */
  IRTC_WrRam ((uint8*)&counter_rtc,sizeof(counter_rtc));
}
```

Freemaster_cfg.h:

```
/*****************************************************************************/
#ifndef __FREEMASTER_CFG_H
#define __FREEMASTER_CFG_H

/*****************************************************************************
* Select interrupt or poll-driven serial communication
*****************************************************************************/
#define FMSTR_LONG_INTR        0    /* complete msg processing in interrupt   */
#define FMSTR_SHORT_INTR       0    /* SCI FIFO-queuing done in interrupt     */
#define FMSTR_POLL_DRIVEN      1    /* no interrupt needed, polling only      */

/*****************************************************************************
* Select SCI communication interface
```

```
*******************************************************************************/
#define FMSTR_USE_SCI          1    /* To select SCI communication interface  */
#define FMSTR_USE_PDBDM        0    /* To select Packet Driven BDM interface  */

//#define FMSTR_SCI_BASE 0x4006A000   /* UART0 base on MKM34Z7                  */
//#define FMSTR_SCI_BASE 0x4006B000   /* UART1 base on MKM34Z7                  */
#define FMSTR_SCI_BASE 0x4006C000   /* UART2 base on MKM34Z7                  */
//#define FMSTR_SCI_BASE 0x4006D000   /* UART3 base on MKM34Z7                  */

/*******************************************************************************
* Input/output communication buffer size
*******************************************************************************/
#define FMSTR_COMM_BUFFER_SIZE 200  /* set to 0 for "automatic"               */

/*******************************************************************************
* Receive FIFO queue size (use with FMSTR_SHORT_INTR only)
*******************************************************************************/
#define FMSTR_COMM_RQUEUE_SIZE 32   /* set to 0 for "default"                 */

/*******************************************************************************
* Support for Application Commands
*******************************************************************************/
#define FMSTR_USE_APPCMD       0    /* enable/disable App.Commands support    */
#define FMSTR_APPCMD_BUFF_SIZE 32   /* App.Command data buffer size           */
#define FMSTR_MAX_APPCMD_CALLS 4    /* num. of app.cmd callbacks? (0=disable) */

/*******************************************************************************
* Oscilloscope support
*******************************************************************************/
#define FMSTR_USE_SCOPE        1    /* enable/disable scope support           */
#define FMSTR_MAX_SCOPE_VARS   8    /* max. number of scope variables (2..8)  */

/*******************************************************************************
* Recorder support
*******************************************************************************/
#define FMSTR_USE_RECORDER     1    /* enable/disable recorder support        */
#define FMSTR_MAX_REC_VARS     8    /* max. num. of recorder variables (2..8) */

/* built-in recorder buffer (use when FMSTR_REC_OWNBUFF is 0)                 */
#define FMSTR_REC_BUFF_SIZE    4096 /* built-in buffer size                   */

/* recorder time base, specifies how often the recorder is called             */
#define FMSTR_REC_TIMEBASE     FMSTR_REC_BASE_MILLISEC(0) /* 0 = "unknown"    */

#define FMSTR_REC_FLOAT_TRIG   0    /* enable/disable floating point trigger  */

/*******************************************************************************
* Target-side address translation (TSA)
*******************************************************************************/
#define FMSTR_USE_TSA          0    /* enable TSA functionality               */
#define FMSTR_USE_TSA_SAFETY   0    /* enable access to TSA variables only    */
#define FMSTR_USE_TSA_INROM    0    /* TSA tables as const (put to ROM)       */

/*******************************************************************************
* Enable the byte access to communication buffer. All Cortex M0-based devices
* require this option to be set in order to avoid misaligned access to integer
* parameters which is unsupported on this platform.
*******************************************************************************/
#define FMSTR_BYTE_BUFFER_ACCESS 1

#endif /* __FREEMASTER_CFG_H */
```

Toolchain support:

IAR EWARM 7.40.7	KEIL uVision 5.15	CrossWorks 3.6	ATOLLIC TrueStudio 5.3.0	Kinetis Design Studio 3.0.0
♦	♦	♦	♦	♦