Freescale Semiconductor
Application Note

AN3587
Rev. 0, 05/2008

Cloth Iron Controlled by MC9RS08KA2

by: Petr Telcer
Freescale Czech System Application Laboratory
Roznov pod Radhostem, Czech Republic

1 Introduction

This application note shows it is possible to use an extra low-cost microcontroller in everyday equipment unaffected by any significant changes for a long time.

A cloth iron was chosen because it has a temperature control implemented using reliable mechanical bimetal, but with a big hysteresis and without any special emphasis on accuracy (there may be temperature drifts of about 30°C or more).

Much better temperature control can be obtained using an embedded control system.

For a cloth iron, this means some temperature sensor, a cheap and simple MCU, and a powerful TRIAC as an actuating device.

In this application, the new low-cost MCU MC9RS08KA2, a member of the RS08 family of 8-bit microcontroller units (MCUs), is inexpensive.

Contents

freescale semiconductor

The MCU (RS08KA2) does not have an ADC module, yet there is a comparator and also an internal bandgap reference voltage for comparing voltage levels on an MCU input. Therefore, it is possible to emulate an ADC with one resistor and one capacitor by measuring the time needed for charging the capacitor to the reference voltage level.

The benefit of this replacement is the change from an inexact on-off control system to a more precise and more sophisticated control system, significantly better (more stable) at sustaining adjusted temperatures.

One further thing is the minimization of disturbances going back to the supply network during switch-on and switch-off. Switching at zero (called zero crossing) is used when the electrical current is also at a zero level and the potential electrical disturbance doesn't arise.

By using the appropriate control method and components, ironing can be considerate of the clothes.

The RS08K2 has eight pins, and this application note uses seven of them. One general-purpose input/output pin is available for additional functionality. For instance, there may be some movement sensor connected and the information from this sensor can be used for implementing some safety feature. Typically, this is an automatic heating shutdown after some defined timeout.

2 Hardware Implementation

2.1 Electrical schematic

Figure 1 shows the electrical schematic used in the design.

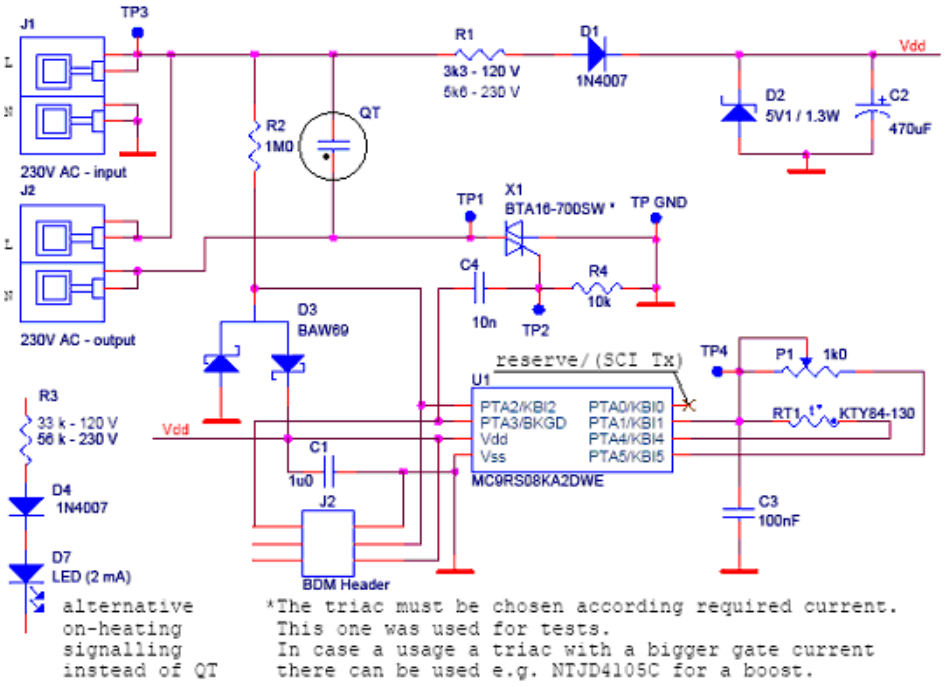

Figure 1. Electrical Schematic Diagram

2.2 User Inputs

The potentiometer P1 is used for adjusting the desired temperature. A value from the temperature potentiometer is read by the emulated ADC converter (see Section 2.7, “Measurement – Emulated ADC converter”).

The converter is realized by external components. On the negative input pin of the comparator, a capacitor is charged through the thermal sensor scanning the iron temperature or through the temperature potentiometer that you set to the required value.

The base position of the temperature potentiometer means that the cloth iron is turned off, but possible to add a mechanical switch to completely turn it off (including all the control system circuits).

2.3 Temperature Sensor

There are several kinds of temperature sensors of varying price, sensitivity, temperature range, and size.

Because high temperature sensors are generally more expensive, this design uses a low-cost silicon temperature sensor, the KTY84-130 (Philips Semiconductors), with a positive temperature coefficient of resistance and is suitable for use in measurement and control systems over a temperature range of 40 to +300 °C (a common iron works in a range of approximately 50 to 230 °C).

2.4 Status Indicators

An LED or a glow-tube indicator is connected directly to the heating element to indicate active heating status. When the indicator is lit, energy is being provided to the heating element.

2.5 Output Stage

The design was tested with the BTA16/700SWRG TRIAC from STMicroelectronics (on 230 V of power line) with a specified maximum IGT of 10 mA. The TRIAC must be chosen according to required load/current to the heating element and with a low gate current to switch-on.

The system switches whole periods, dependent on the output from the PI regulator.

The switching is realized in zero to minimize disturbances going back to the supply network.

Power supply for the whole control system is also taken from the line voltage with the help of a simple voltage source (see Section 2.6, “Power Supply for the Control System”).

The supply is current limited so the low IGT of the TRIAC is significant.

When the temperature potentiometer is set to the position marked off, the duty is zero. However, the power supply and control system remain alive. This is why it is better to use the main switch.

2.6 Power Supply for the Control System

The power supply is designed to be as simple and inexpensive as possible.

The power supply voltage is taken directly from the AC power phase line through the reducing serial resistor. After stabilization by the Zener diode and filtration by the electrolytic capacitor, it is used as a power supply for the microcontroller (see Figure 1).

For synchronization of all operations there is a signal created directly from the power line frequency: The 230 V voltage is reduced by a large serial resistor (1.0 M) and limited by two Schottky diodes (or similarly quick diodes) for the KBI input to the MCU in WAIT mode. The signal on KBI awakes the MCU and program continues. This solution ensures that the switching is approximately at the same time when the power phase goes through the zero level, to prevent injection of undesirable disturbances into the power network.

2.7 Measurement – Emulated ADC converter

Correct functioning of the iron demands scanning two inputs. The first input is the required temperature you adjust (temperature potentiometer), and the second input is the actual temperature of the sole plate scanned by a temperature sensor. In both cases, the changing value is resistance.

Because the MCU does not have an ADC module, it is necessary to emulate such a circuit in other ways.

The RS08KA2 has an analogue comparator (ACMP) with an internal reference voltage (1.218 V) and the ACMP module is used to indirectly measuring the needed values. Components with a changing resistance are used for charging a capacitor, and the time necessary to recharge the capacitor to the internal reference level is measured. The measured time is then used as an index for reading a look-up table, where the corresponding temperatures are stored.

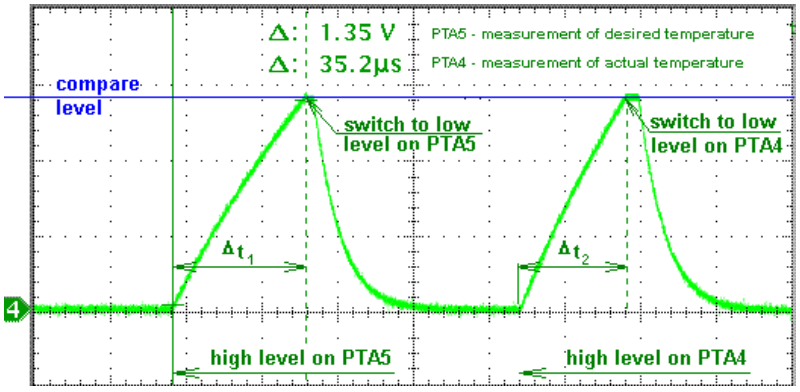

Figure 2. Measurement Pulses on the ACMP (TestPoint2)

The value is read is determined by two output pins, PTA4 – temperature sensor, PTA5 – potentiometer (see Figure 1).

Both measurements are placed between the 28th period and the 32nd period of the control frame (CF), always at the beginning of the positive half-wave (the control frame is 32 periods long that represents 0.64 s; more information about the CF can be found in chapter 4.), regardless of whether the periods are switched-on or not. The measuring is not affected by some transient phenomenon or voltage

drop due to a large load when the measurement is before switching-on the current to the heating element. The measurement pulses are shown in figure 2. and their position is shown in Figure 3.

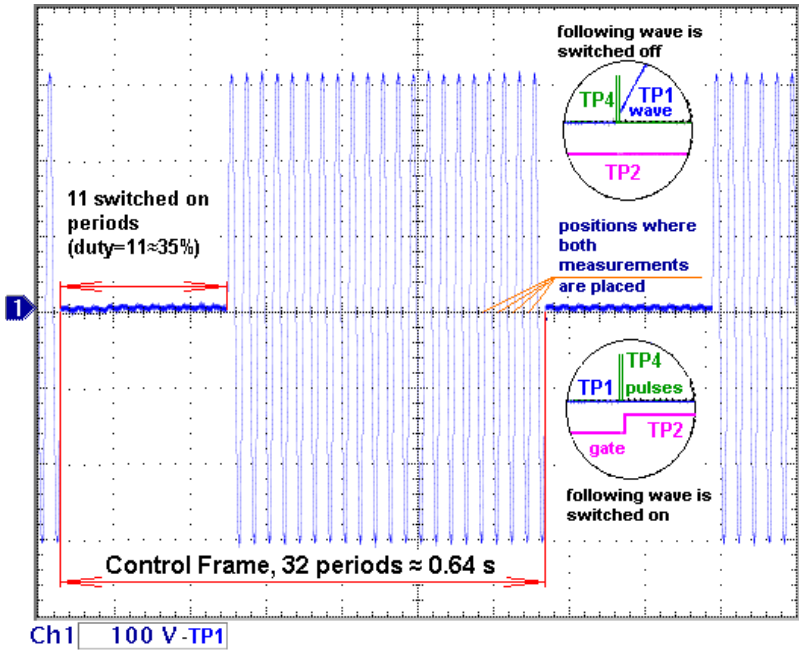

Figure 3. Control Frame (TP1)

NOTE

An oscilloscopic probe is at Test Point 1 (TP1, blue line), so that the switch-on periods are displayed at zero level, because the TRIAC switches-on against ground (the ground is used as a reference). The measurement pulses (green line) are captured at TP4 and the switch pulses are from a gate in TP2 (pink line).

2.8 TRIAC Switching

As already mentioned, a TRIAC with a low current to the gate was used. The TRIAC is controlled by a general-purpose output. There is a serial capacitor to help switch the negative half wave (see Figure 1).

The solution demands one pulse on the TRIAC gate for the whole positive half-wave with a fall edge where the negative half wave begins, which assures the TRIAC is switched-on for the whole period. The captured TRIAC gate switch pulses are shown in Figure 4.

The regulation is realized by switching a quantified number of periods within the control frame. The number of switched periods is calculated by a PI regulator algorithm with implemented limits with regards to the physical properties of the device. In the face of the large thermal accumulative character of the metal sole plate, the first periods in every control frame are always switched-on and the rest are switched off (in dependence on the output from the PI regulator).

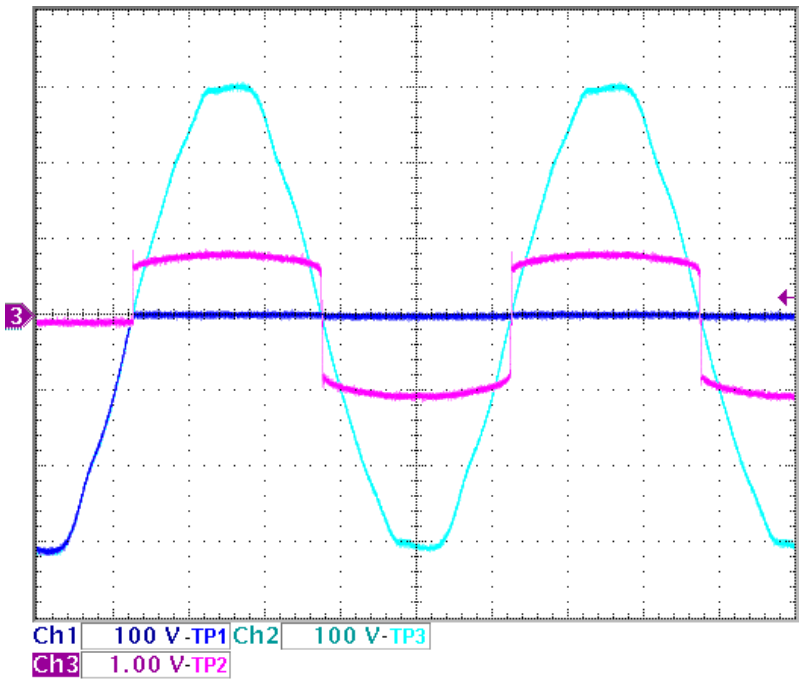

Figure 4. Switch Pulses on the TRIAC Gate

3 Development Tools

The programming language used for this application is C in combination with some functions in assembler. This application was developed using CodeWarrior 6.1 with the MC9RS08KA2 patch properly installed.

NOTE

The older releases of CodeWarrior do not support the C compiler for the RS08.

In the attached code, there is also a prepared software emulated SCI transceiver function that enables sending information to the outside world via a serial link (in one byte messages only) for debugging possibilities.

NOTE

To prevent irreparable damage to the PC, use a sufficient galvanic isolation for the serial line.

4 Software Implementation

The body of the software is written in C language, but there are mathematical functions and the software emulated SCI transceiver function (for debugging only) written in assembler for better efficiency.

Main functions are independent modules that are also useful separately.

The core of the whole code is an algorithm for the proportional-integral (PI) regulator (controller). The P element allows quick movement to the set point when the actual value is far away and the I element allows

accurate positioning around the set point, compensating for the proportional offset problem. In most cases, a mixture of the P and I elements are used to make a robust controller.

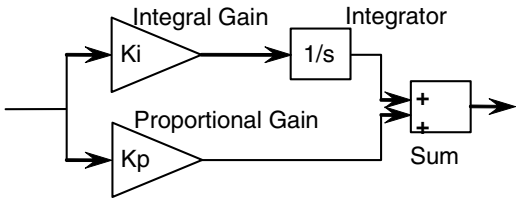

Figure 5. Standard PI Regulator

The main framework of the whole software is called a control frame and is 32-periods long. It always starts at the beginning of a period (see Figure 3). This is guaranteed by synchronization through the KBI interrupt that brings the MCU out of wait mode and triggers the control frame.

In the first run of the CF, only the desired and actual temperatures are measured. When there is a difference between the desired temperature and the last desired temperature (set to zero), the change_flag is set. The TRIAC is not switched-on. In the next run of the CF, full power is already used on the heating element to reach approximately 89 % of the desired value (the limit is selectable in the code through a parameter). Then the PI regulator starts up and takes control of the heating. This solution was chosen to quickly reach the desired temperature without causing a huge overshoot. The regulator algorithm calculates the value used (after a modification) as the duty for switching.

When the desired temperature is changed by the potentiometer, the change_flag is set and a further process depends on the newly adjusted value. If the new value is higher than the actual temperature by more than 11 % (this value is used in the attached code), the duty is maximal. If the new value is significantly lower than the actual temperature, the duty is zero. If the change is not significant the transition to a new value is managed only by the control algorithm. Also, if there is a discrepancy between the desired and the measured temperatures due to a cooling by the ironed cloth, only the control algorithm keeps the adjusted temperature.

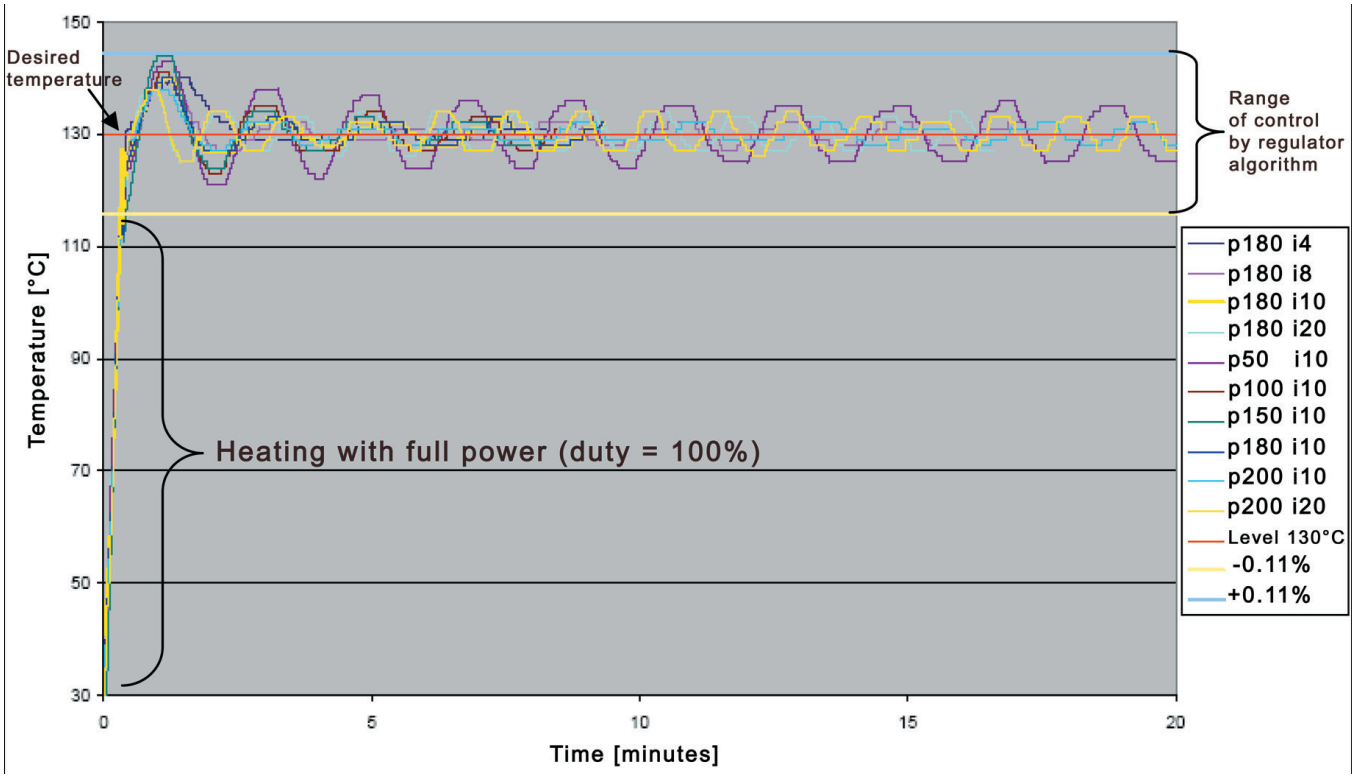

Figure 6. Time Curves of Controlled Temperature with Various Proportional and Integral Parameters

5 Execution Flow and System States

The operation of the control system (a framework only) can be described with a simple flow diagram as shown in Figure 7.

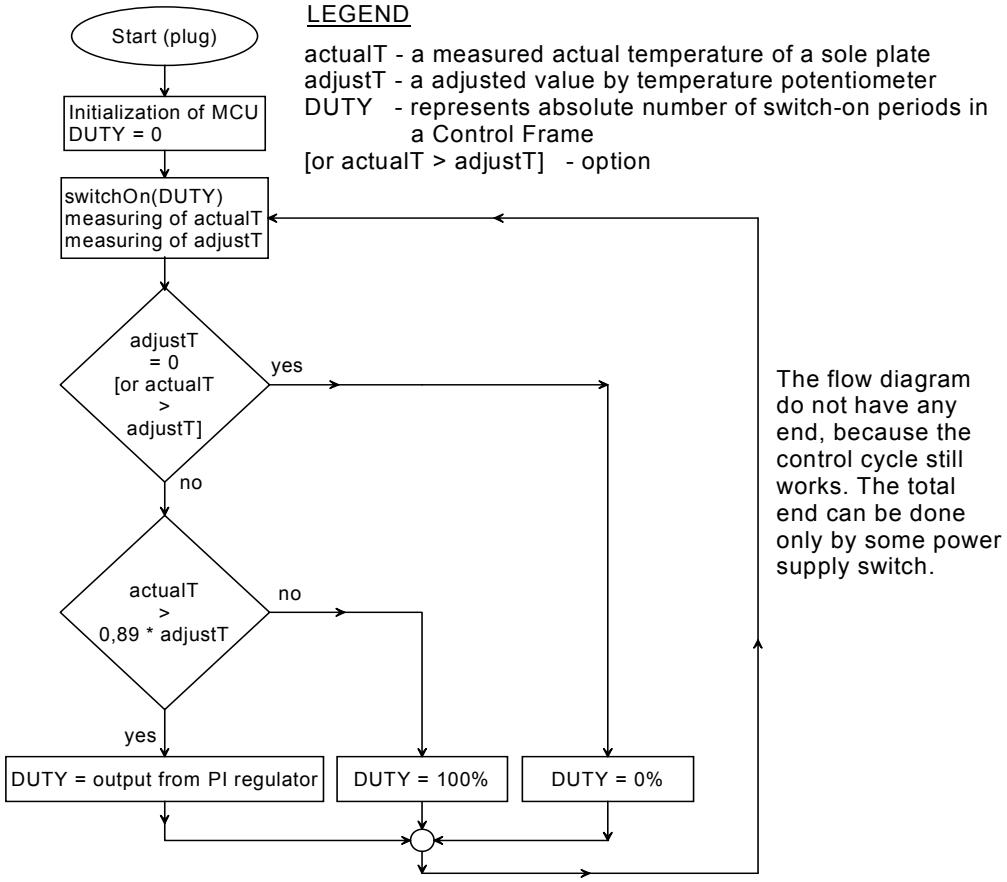

Figure 7. Framework Flow Diagram

6 Look-Up Tables, Adjustment, and Calibration

6.1 Look-Up Tables and Adjustment

With the suggested temperature sensor, the transfer characteristic between a measured temperature and the charging time is nearly linear. More precise values are available when look-up tables are used. The look-up tables correct the nonlinearity of the method as well as the nonlinearity of the sensor used.

The look-up tables are stored in two data arrays. As an index for reading them, the measured charging time is used. On account of this, there is a time offset used that is shorter than the minimum needed for both measurements. The offset time is followed by the intrinsic time measurement to the moment when the voltage on the capacitor reaches the voltage level of the internal reference. This time is then used as the index to read out the corresponding temperature. A construction of the look-up table is possible in several ways. For example, one way is to measure a number of points with auxiliary scanning of the sole plate temperature and construction of the transfer characteristic between the measured temperature and the charging time. It is then possible to define an approximation function from the characteristics and obtain

all the elements of the table from this function. More measurement points mean a better accuracy. The best accuracy may be reached by exactly measuring the temperature step-by-step. This may be done so that the MCU sends the measured time values (via the emulated serial link) to recording equipment concurrently with data from an independent thermometer during a slow increase of the sole plate temperature. This may be done for example by modifying the software so that only every tenth wave is switched-on. The data cells of the look-up table are then filled with the processed matched data.

The construction of the look-up table for the potentiometer is much simpler. Measuring only the end points is sufficient, and then it is possible to distribute the desired temperature according to needs or requirements. Start with a zero value that is in more data cells for a safer adjustment (the heating is switched-off), and then slowly increase to the highest temperature by repeating more significant points (such as, the recommended temperatures for ironing specific fabrics or according to regional customs). The graphical representation of the look-up table used during final testing is shown in a chart in Figure 8. In the source code, there are two tables available. The first one is of 64 items and the second one has 256 items for a potential possibility of a finer adjustment. In the event of using the bigger table, the parameter USED_TABLE (located in the adc_cfg.h file) must be changed from 0 to 2.

An example of a shorter look-up table (64 elements) for a potentiometer (extracted from the attached code):

```
const UBYTE ADC_potentiometerTable[] = {
  0,   0,   0,   0,  80,  82,  84,  86,  88,  90,  92,  94,  96,  98, 100, 102,
104, 106, 108, 110, 110, 110, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128,
130, 132, 134, 136, 138, 140, 142, 144, 146, 148, 150, 150, 150, 150, 152, 155,
160, 165, 170, 175, 180, 185, 190, 195, 200, 200, 200, 200, 205, 210, 215, 220,
};
#endif
110, 150, 200 - repeated values for the emphasis of more significant temperatures
```

Adjustment of the PI regulator can be done on an experience basis (trial and error), on a behaviour observation during testing, or with the help of some theoretical techniques. All regulator parameters and other parameters in parameters for a modification of control system behaviour section in the source code are dependent on specific hardware.

6.2 Calibration

When the look-up tables are built and the temperature sensor is installed in a definitive place in a specific iron that is mechanically complete, a calibration can be done by a similar process as building the look-up tables. The temperature potentiometer is set to a max value and the actual temperature of sole plate is measured by the thermosensor, converted by the look-up table, and filtered by the IIR filter. The processed temperature is then sent out continuously from the iron via the emulated serial link. It is necessary to collect the data together with an independently measured temperature, and after a comparison of both measured values, correction data is obtained (as a difference of the correspondent values). The ADC_temperatureTable look-up table must be corrected according to the differential values so that the sent values correspond with real temperatures. If the temperature dial on the iron has temperature marking on it, it is also good to check these points, but on full power and with full function of the control system. If the marked temperatures do not correspond with the real temperatures, the temperature dial can be

mechanically readjusted (if the offset is the same for all marked temperatures), or the ADC_potentiometerTable look-up table can be modified.

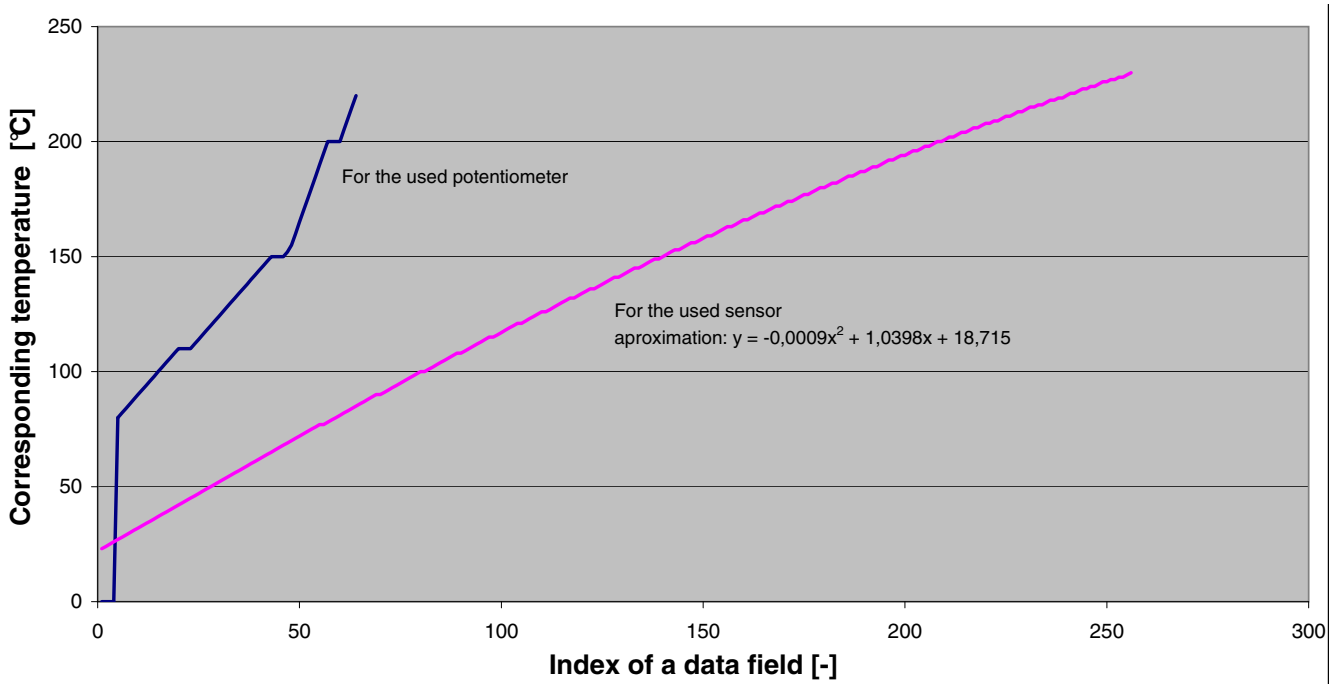

Figure 8. Graphical Representation of the Look-Up Tables from the Cattached Code

7 Two Benefits of the Solution

The first benefit is that the cloth iron holds the adjusted temperature better with the possibility of more precisely setting the behaviour when the sole plate is cooled by the ironed cloth.

The second benefit is that an iron with an embedded control system operates without any electrical disturbance back to the power supply network.

With new digital technologies, software samples and development tools, designers can quickly build up embedded solutions and replace mechanic control systems with embedded control systems.

8 Testing and Validation

The system was tested by placing the temperature sensor on the metal sole plate (input power of the iron was 1.4 kW). The main features of the solution were tested. The look-up tables must be built well and the regulator parameters set correctly for the actual iron for the correct functioning of the control system.

Also, it is possible to change the limits for the switch-over from full-power to regulated power, to change the values used as the starting values, etc. (other customised possibilities are indicated directly in the source code).

There is also a prepared function for serial communication for debug processing.

9 Conclusion

This application note shows designers interested in, but not familiar with, digital embedded systems how to implement a control system based on a low-end 8-bit processor.

It may serve as a suggestion, with one elaborated specific solution and some already prepared possibility for modification, according to the actual characteristics of a specific cloth iron.

The reached hysteresis with the embedded system was under ±5 °C with a look-up table that was built from the approximation of a curve measured in five degrees raster. Better results can be obtained with a look up table built degree by degree, a shorter control frame, etc.

Appendix A
List of main.c Source Code

The following code represents a construction of the main framework for the whole application called the control frame. The remaining part contains calculation of control algorithm, limits etc.

The list is without switch-on routine.

```
/*****************************************************************************
*
* Freescale Semiconductor Inc.
* (c) Copyright 2004-2005 Freescale Semiconductor, Inc.
* (c) Copyright 2001-2004 Motorola, Inc.
* ALL RIGHTS RESERVED.
*
*****************************************************************************//*!
*
* @file   main.c
*
* @brief  Embedded Control of Cloth Iron with MC9RS08KA2
*
* @author r79251
*
* @version 1.1.0.0
*
* @date Jan 2007
*
*****************************************************************************
*
* This is the main C file of the application.
*
*****************************************************************************/


/*****************************************************************************//*!
*
*  @mainpage
*
*  This application is concerned with the full control of a cloth iron by microcontroller
*  MC9RS08KA2, or else - by replacing the traditional bimetal control for an embedded
*  control system.
*
*  -# This application is divided into several modules according to their functions.
*     The modules are:
*     DRIVERS:
*     ADC -  emulated ADC converter for reading the desired temperature and
*            measuring the sole plate temperature
*     SCI -  by software emulated SCI transceiver for debugging purposes
*             both drivers are followed by configuration files (adc_cfg.h; sci_cfg.h)
*             where it's possible to change some module parameters
*
*     MODULES:
*     main        - the main file where the whole software is based, with
```

```
*                   a definition of the Control Frame determining the exact position
*                   of measurement, switch-on of the TRIAC, etc.
*     controller  - there is an algorithm for the PI regulator
*     math        - there are basic mathematical operations with limitation that are
*                   needed for the regulator module
*
*  -# usage of pines
*     PTA0/KBI0/ACMP+ reserve and SCI transmit PIN (output) for debug
*     PTA1/KBI1/ACMP- input for measuring
*     PTA2/KBI2       input wave for zero crossing
*     PTA3/BKGD       control of TRIAC
*     PTA4/KBI4       selection of Thermo sensor for measurement
*     PTA5/KBI5       selection of Potentiometer for measurement
*  -# both measured values by the ADC are filtered by an IIR filter
*  -# in the main.c files there are also some parameters for modification of control
*     system behaviour. The parameters are inside the marked section.
*  -# the defined duration of the Control Frame (CF) is 32 periods that represents
*     a time of 0.64 s. In every CF these steps are made:
*     four measurements of the desired and actual temperatures, including the IIR filtration;
*     calculating of the duty value;
*     depending on the duty, the TRIAC is switched-on.
*
*******************************************************************************
*
* @note 1. In the code, marked lines are left - for debugging. These lines can help
*          in the debug process. The lines are mainly in the SCI_putc function.
*          This function can be copied and edited for sending some
*          other information according to actual needs
*
* @note 2. All regulator parameters and other parameters in the section
*         'Parameters for a modification of control system behaviour'
*          are dependent on specific 'hardware' - on the actual iron.
*
*******************************************************************************/


#include <hidef.h>      /* for EnableInterrupts macro                      */
#include <stdlib.h>     /* includes header file of a standard library       */
#include "derivative.h" /* includes peripheral declarations                 */

#include "main.h"

#include "sci.h"        /* includes header file of an SCI module             */
#include "adc.h"        /* includes header file of an ADC module             */
#include "controller.h" /* includes header file of a controller module      */
#include "math.h"       /* includes base mathematical function declarations */

#pragma MESSAGE DISABLE C4301   /* 'Inline expansion done for the function call '*/
#pragma MESSAGE DISABLE C3604   /* 'Static <Internal Object> was not referenced' */


/* Function prototypes */
void MCU_init(void);            /* Device initialization function declaration */
void SwitchOn (void);           /* TRIAC switch function declaration          */

#pragma INLINE
/* IIR filtering  */
```

```
BYTE IIR_Filter (UBYTE filterInput, int * filterState, UBYTE filterShift) {
    *filterState += (filterInput - *filterState) >> filterShift;
    return (BYTE)*filterState;
}


/* Variable definition */
volatile APPSCSTR   _APPSC;/* Application Status and Control Register */

UBYTE  desiredTemperature;/* desired temperature from pot */
UBYTE  lastDesTemperature;/* desired temperature before on cycle */
int    desiredTemperatureBuffer;/* filtered desired temperature from pot */
UBYTE  actualTemperature;/* filtered measured temperature from sensor */
int    actualTemperatureBuffer;/* filtered desired temperature from pot */
BYTE   controlDifference;/* Difference between desiredTemperature and */
/* actualTemperature - input to regulator */
BYTE   outputReg;/* output from regulator */
UBYTE  startReg;/* target temperature - /CBAND/ when the regulator /*
/* takes control */
UBYTE  duty;/* number of switched wave */
UBYTE  wave;/* counting of wave inside controlFrame */

#define TIMEOUT_10MS 313/* 313 x 0.032 (the timer period) = 10.016 ms - offset */
/* for switch-on negative half wave to switch whole */
/* periods, if the switch-on is not quite sure, */
/* the value can possibly be increased. In case of some */
/* bigger disturbance (pulse before the */
/* switch-on), to decrease with checkout of a good */
/* switching. */

/* Parameters for a modification of control system behaviour */
#define CBAND               3/* control band 1 +/-50%; 2 +/-25%; 3 +/-11%; 4 +/-6%;  */
/* 5 +/-3%  */
#define HIGH_CONTROL_LIMIT   1/* 1 - when the desired value is exceeded, the duty =0; */
/* 0 -the feature is disabled */
#define CONTROL_FRAME       15/* number of wave in regulation frame including zero - 32 */
/* waves; quicker - 16 waves */
#define CONTROL_SCALE        3/* 2 - for Control Frame 32 waves; 3 - for Control Frame */
/* 16 waves */


/* parameters for controller algorithm    */
REG_PIparamsType regParams = {
/* UBYTE proportionalGain */   200,
/* UBYTE integralGain     */    10, /*
/* int   integral_K1      */0x1000
/* feed the controller at switch over from full power to control by the controller */
};                                  /*  (Ox1FFF is 25 % from Ox7FFF)   */
                                    /*  (Ox1000 is 12.5 % from Ox7FFF) */
                                    /*  (Ox0666 is  5 % from Ox7FFF)   */

#define FILTER_SHIFT    2           /*  a weighted coefficient of the IIR filter */

/* end of 'Parameters for a modification of control system behaviour' section  */


/*****************************************************************************//*!
```

```
*
* @brief  Application Loop
*
* @remarks
*
* 1) definition of Control Frame (32 periods)
* 2) switch-on TRIAC depending on calculated duty
* 3) in last four periods inside the Control Frame both measurements are executed
* 4) when the desired temperature is changed, the bChange flag is set
* 5) the controlDifference is calculated
* 6) the regulator algorithm gives a result
* 7) according to the controlDifference and regulator result, the duty is determined
* 8) flag for limitation to max possible power is set for the next run when the max power is
reached
*
*****************************************************************************/


#define GET_HIGH(x) (BYTE)((x)>>8)

void main(void) {

  MCU_init();/* call Device Initialization */
  duty = 0;/* starting value of the duty */
  for(;;) {


     __RESET_WATCHDOG();

/* construction of the control frame - start */

/* set the Control Frame - length of the CF is determined by the variable wave*/
for (wave=0; wave <= CONTROL_FRAME ; wave++)
    {
/* wait for the KBI interrupt - triggering by/synchronisation with a wave */
    WAIT();
/* check if the event is really a KBI interrupt */
if (KBISC_KBF == 1)
{
/* Clear the KBI pending interrupt */
    KBISC_KBACK = 1;
/* both measurements only in the last four waves */
    if (wave > (CONTROL_FRAME - 4))
    {
/* measure and filter both measured values by IIR filter */
desiredTemperature = IIR_Filter(ADC_Read(CHAN_POTENTIOMETER), anddesiredTemperatureBuffer,
FILTER_SHIFT);
actualTemperature  = IIR_Filter(ADC_Read(CHAN_THERMISTOR), andactualTemperatureBuffer,
FILTER_SHIFT);
            }
/* switch-on TRIAC according to the computed duty */
if ((wave < (duty+1)) andand (duty > 0)) SwitchOn();
/* set the change_flag to indicate a change of the desired temperature */
if (lastDesTemperature  != desiredTemperature) APPSC_bChange = 1;
/* store the desired temperature for next run - for change_flag settings  */
lastDesTemperature = desiredTemperature;
```

```
/* for debug only - it sends the actual temperature once every Control Frame. There is a possible
place for Transmission of any variable(s) according to needs.

if (wave == CONTROL_FRAME)
{
    SCI_putc(desiredTemperature);
    SCI_putc(actualTemperature);
    SCI_putc(duty);
}*/
}
    }
/* construction of the control frame - end */


__RESET_WATCHDOG();


/* temperature limit for a switching from full power to control of regulator   */
/* startReg = dT- dT/8 = 89% of desiredTemperature (example for CBAND=3 ~ 89%) */
    startReg = desiredTemperature - (desiredTemperature >> CBAND);

/* if actual temp. is lower then startReg and the bChange is set, the power is full*/
    if ((actualTemperature < startReg) andand (APPSC_bChange == 1)) duty = CONTROL_FRAME;
    else
    {    /* switch from full power to set integral_K1 as a start condition when */
         /* the desired value is near - CBAND is reached */


        APPSC_bChange = 0;

        /* calculation of difference between temperatures */
        controlDifference = uSUB8((UBYTE)desiredTemperature, (UBYTE)actualTemperature);


        /* Call PI controller - computing the duty */
        outputReg = GET_HIGH(regPIsat(controlDifference, andregParams, APPSC_bPmax));


        /* set saturation flag - full power is already reached */
        APPSC_bPmax = (outputReg == 0x1F ? 1 : 0);

        /* Scale controller output to heating frame (from +/- 128 to +/- 32 range) */
        duty  = (BYTE)(outputReg > 0 ? outputReg >> CONTROL_SCALE : 0);
       /* duty = 0 when desired value is exceeded - this is possible to turn-off by comment */
        if ((actualTemperature > desiredTemperature) andand HIGH_CONTROL_LIMIT) duty = 0;
    }

  } /* loop forever */
    /* please make sure that you never leave main */
}
```

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

Document Number: AN3587
Rev. 0
05/2008

freescale™
semiconductor