Low Cost Universal Motor Drive Using Kinetis L family

Application note

Rev. 0
09/2012
By: Ivan Lovas
Roznov System Application Laboratory
Roznov pod Radhostem, Czech Republic

Contents

1 Introduction

Today the universal motor is still widely used in home appliances such as vacuum cleaners, washers, hand tools, and food processors. The operational mode, which is used in this application, is closed loop and regulated speed. This mode requires a speed sensor on the motor shaft. Such a sensor is usually an incremental sensor or a tachometer generator. The kind of motor and its drive have a high impact on many home appliance features like cost, size, noise, and efficiency. Electronic control is usually necessary when variables speed or energy savings are required. MCUs offer the advantages of low cost and attractive design. They can operate with only a few external components and reduce the energy consumption as well as the cost. This circuit was designed as a simple schematic using key features of a Kinetis L MCU. For demonstration purposes, the Freescale low cost Freedom KL25z development platform was used.

This application note describes the design of a low-cost phase angle motor control drive system based on Freescales's Kinetis L series microcontroller (MCU) and the MAC4DC snubberless triac. The low-cost single-phase power board is dedicated for universal brushed motors operating from 1000 RPMs to 15,000 RPMs. This application note explains both HW and SW design with an ARM Kinetis L series MCU. Such a low-cost MCU is powerful enough to do the whole job necessary for driving a closed loop phase angle system as well as many others algorithms.

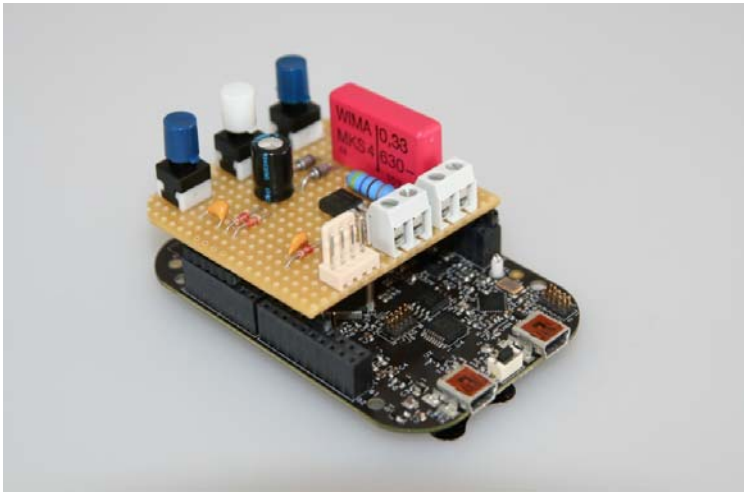

Figure 1-1 Freedom development platform with universal motor drive board extension

The phase angle control technique is used to adjust the voltage applied to the motor (refer to Figure 3-1 Power line to software synchronization). A phase shift of the gate's pulses allows the effective voltage, seen by the motor, to be varied. All required functions are performed by just one integrated circuit and a small number of external components. This allows a compact printed circuit board (PCB) design and a cost-effective solution.

1.1 Added Value Using a Microcontroller

Compared to a poor analog solution, an MCU-based drive shows many advantages. Some of them are:

- Choice of different control algorithms
- Choice of any shape of speed command (phase acceleration and deceleration)
- Choice of any type of tachometer
- Software that can make the hardware simpler
- Diagnostic functions
- Remote control by wire and communications protocol
- Open for innovation

1.2 Freescale Freedom development platform

The Freescale Freedom FRDM-KL25Z is a low-cost evaluation and development platform to demonstrate the capability of the Kinetis-L family of MCUs, ARM® Cortex™-M0+ based and targeting energy-efficient applications.

Features:

- KL25Z128VLK4--Cortex-M0+ MCU with:
- 128KB flash, 16KB SRAM
- Up to 48MHz operation
- USB full-speed controller
- OpenSDA- sophisticated USB debug interface
- Tri-color LED
- Capacitive touch "slider"
- Freescale MMA8451Q accelerometer
- Flexible power supply options

- Power from either on-board USB connector
- Coin cell battery holder (optional population option
- 5V-9V Vin from optional IO header
- 5V provided to optional IO header
- 3.3V to or from optional IO header
- Reset button
- Expansion IO form factor accepts peripherals designed for Arduino™-compatible hardware

Figure 1-2 Freedom low cost development

1.3 Freescale Kinetis L family

Kinetis L series MCUs combine the exceptional energy-efficiency and ease-of-use of the new ARM® Cortex™-M0+ processor with the performance, peripheral sets, enablement and scalability of the Kinetis 32-bit MCU portfolio.

The Kinetis L series frees power-critical designs from 8- and 16-bit MCU limitations by combining excellent dynamic and stop currents with superior processing performance, a broad selection of on-chip flash memory densities and extensive analog, connectivity and HMI peripheral options.

Kinetis L series MCUs are also hardware and software compatible with the ARM Cortex-M4-based Kinetis K series, providing a scalable migration path to more performance, memory and feature integration.

Devices start from 8 KB of flash in a small-footprint 4 x 4 mm 24 QFN package extending up to 256 KB in a 121 MBGA package. Each combines ultra-low-power performance with a rich suite of analog, communication, timing and control peripherals.

Kinetis L Series Family Graphic

Common Features

System
ARM® Cortex™-M0+ Core
Multiple Low-Power Operation Modes, Clock Gating, 1.71V–3.6V
DMA, Cross Bar Switch
Operating Temp: -40°C to +105°C [3]

Memory
90 nm TFS Flash Memory (High Reliability, Fast Access)
SRAM
Internal Memory Security/Protection

Analog Peripherals
16-bit ADC [1]
12-bit DAC
High-Speed Comparators
Low-Power Touch Sense Interface

Serial Interfaces
LPUART, UART [2]
SPI, I²C

Timers
RTC
Low-Power TPMs
Low-Power Timers
System Timers

Optional Features

CPU	Internal Memory		Communications	HMI
KL4 Family: USB, Segment LCD				
48 MHz	128 KB to 256 KB Flash	16 KB to 32 KB SRAM	USB OTG (FS)	Segment LCD
KL3 Family: Segment LCD				
48 MHz	64 KB to 256 KB Flash	8 KB to 32 KB SRAM	–	Segment LCD
KL2 Family: USB				
48 MHz	32 KB to 256 KB Flash	4 KB to 32 KB SRAM	USB OTG (FS)	–
KL1 Family: General Purpose				
48 MHz	32 KB to 256 KB Flash	4 KB to 32 KB SRAM	–	–
KL0 Family: Entry Level				
48 MHz	8 KB to 32 KB Flash	1 KB to 4 KB SRAM	–	–

[1] Feature not available on all KL1, KL2, KL3 MCUs (some have 12-bit ADC)
[2] Feature not available on KL0 MCUs (KL0 MCUs have LPUART)
[3] CSP packages -40°C to +85°C

Figure 1-3 Kinetis L family portfolio

2 Hardware description

2.1 Function description

In Figure 2-1 Schematic, the schematic of a phase angle motor control board is shown. The phase angle drive needs only one integrated circuit - the MCU. The snubberless triac MAC4DC is used as the power device. This triac has high dv/dt immunity and, therefore, there is no RC circuit around the triac. The MT1 pin of the triac is connected to V_{CC} and the GATE pin is connected directly to the microcontroller. The triac's turn-on level on pins PTC1–PTC3 is 0 V. This configuration was chosen for two reasons. The first is the current capability of the pins of port A. In the Kinetis L family specification can be found that the source capability is 5 mA and the sink capability is 10 mA. Our choice is the sink mode. The second reason is determined by the triac. A common law states that snubberless triacs with high dv/dt immunity need a higher gate trigger current I_{GT}. The MAC4DC needs at least 25 mA typically in case of negative I_{GT}. For an operational mode with positive I_{GT}, the gate trigger current is much higher. Our choice is to use a negative I_{GT}. Three pins, PTC 1,2,3 , are connected together and are powerful enough to cover the amount of current needed by the gate and to turn on the triac reliably.

The power supply includes only a few components (D1, D4, R2, C12, C13, and F1). The output voltage is +3.3 volts and despite its simplicity it is able to supply the microcontroller, the external control panel, and also the triac.

WARNING: ***This circuit is powered directly from the line. Do not touch any parts of this board. When working with this board, do not connect any computer, scope, or development system. In this case, it is necessary to use an isolation transformer.***

2.2 Schematic

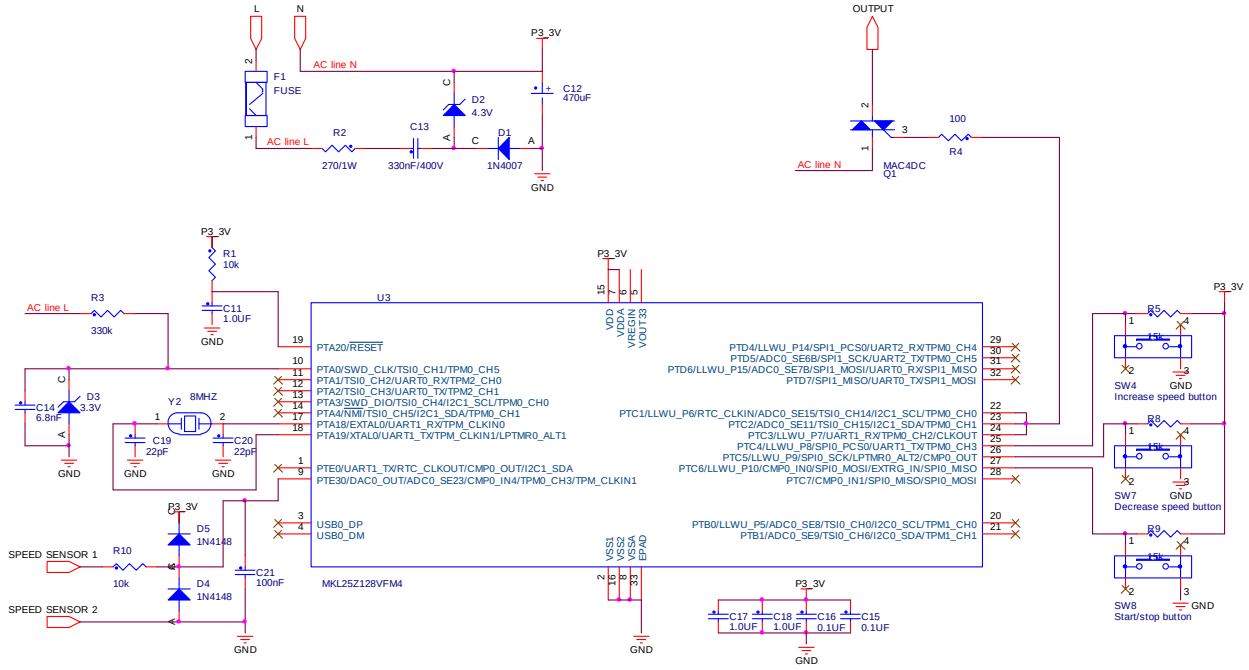

Figure 2-1 Schematic

The circuitry connected to pin PTA2 is needed for the acquisition of a synchronization signal. This signal provides the most important information to the microcontroller, which is the zero crossing of the line voltage. The point of the zero crossing is fundamental for the calculation of any triac's action. All actions and the functionality concerning the triac are controlled by software. Because this board provides the control algorithm in closed loop mode, some devices allow connection of a tachometer or incremental position sensor. An input filter with D5 protects the internal comparator against high voltage at high speed, and diode D4 protects the comparator against negative voltage. If the incremental speed sensor is needed, this protection circuit can be omitted. The output square wave signal from the internal comparator is internally connected to timer. With this arrangement, the input capture feature of the microcontroller can be used. The speed command can be set externally using buttons SW1 and SW3. A simple external control panel should be linked with the phase angle power stage when the external commands are needed.

2.3 User interface

The pushbuttons SW1, SW2 and SW3 allow to control the drive. There are external pull-up resistors because this device doesn't support internal pull-up resistors on each pin. The SW1 and the SW3 are used to increase or decrease the motor speed and the SW2 is used to Start/Stop the drive.

NOTE: The control panel must be isolated from the user under all possible circumstances.

2.4 Synchronization HW

The basic principle of the phase angle control algorithm is simple: match the firing pulse time of the triac in relation with the zero crossing of the line voltage. A phase shift of this firing pulse produces a variable output voltage on the load. A structure with two interrupts has been chosen to ensure proper functionality and some additional CPU performance capacity. See Figure 3-1 Power line to software synchronization.

As it is well known, the phase angle drive system needs to have information about the line voltage and its zero crossing points. The appropriate signal is connected to pin PTA0. Only one resistor (R3), one capacitor (C14), and a diode (D3) are used. Due to its simplicity and its low-cost solution, the output signal is not a real square wave (purple signal). In Figure 2-2 Zero cross detection waveform, the relationship between the input signal on PTA0 (purple), zero cross detection (green) and line voltage (blue) can be seen.

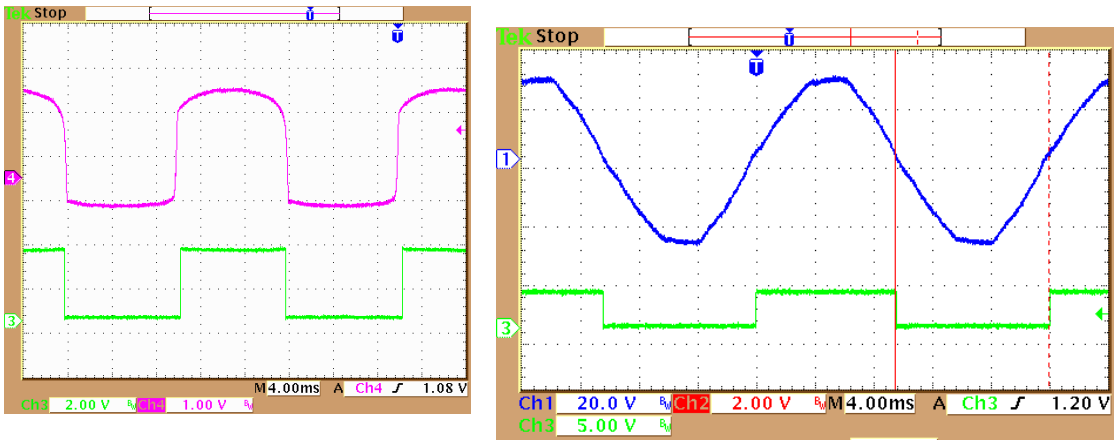

Figure 2-2 Zero cross detection waveform

For more information about synchronization software see chapter 3.2Synchronization SW.

3 Software

3.1 Peripheral configuration

For the motor control to properly function the following peripherals should be configured.

- CPU core
- Periodic interrupt timer (PIT)
- GPIO port and interrupt from GPIO
- Timer/PWM Module (TPM)

3.1.1 CPU core configuration

Default system startup configures the CPU clock to 48 MHz. To conserve energy consumed by CPU, the core clock frequency should be decreased. In our case the software is very simple, so the clock can be configured to 3MHz. It can be done by changing the core clock output divider in register SIM_CLKDIV1. However there is still enough power to execute motor control algorithm and other control software.

Changing the frequency of core clock:

```
SIM_CLKDIV1 = SIM_CLKDIV1_OUTDIV1(15); // select core clock divider to 15
```

3.1.2 Periodic interrupt timer (PIT) configration

Configuration of PIT timer is very easy. Only the peripheral clock, timer start value, and interrupt enable bits should be configured.

```
// PIT configuration
    SIM_SCGC6 |= SIM_SCGC6_PIT_MASK;
    PIT_MCR = PIT_MCR_FRZ_MASK; // freeze timer when debug mode
    PIT_LDVAL0 = 24000; // default start value (333nS per one tick at 3MHz)
    PIT_TCTRL0 = PIT_TCTRL_TIE_MASK | PIT_TCTRL_TEN_MASK; //enable Timer and
                                                          timer interrupt
```

3.1.3 GPIO port and interrupt setup

At least 8 GPIO pins should be used for this application. Two versions of configuration can be used. The first one is for MKL25Z128VFM4 device which is packed in a QFN32 package. The second configuration is used for Freedom KL25Z development platform which contains the 80-pin package. The pins are set as shows Table 3-1 Microprocessor pins usage.

	KL25Z128VFM4	**Freedom KL25Z**
User interface	PORT C 4,5,6	PORT E 23,29,30
Triac gate pulses	PORT C 1,2,3	PORT E 20,21,22
Zero cross detection	PORT A 0	PORT A 17
Speed measurement	PORT E 30	PORT C 6

Table 3-1 Microprocessor pins usage

The interrupt should be configured properly. To do this, we need to write down an interrupt into the vector table.

```
#undef  VECTOR_038
#define VECTOR_038 PIT0_isr

#undef  VECTOR_046
#define VECTOR_046 Zero_Cross_isr
```

Then, we need to enable interrupt in the NVIC register:

```
/* Determine which of the NVICISERs corresponds to the irq */
      NVIC_ICPR |= 1 << ((38 - 16)%32);
      NVIC_ISER |= 1 << ((38 - 16)%32);

      NVIC_ICPR |= 1 << ((46 - 16)%32);
      NVIC_ISER |= 1 << ((46 - 16)%32);
```

3.2 Synchronization SW

Only one periodic interrupt timer is used to generate the delay after zero cross and the triac gate pulses. For an explanation how the synchronization works the "Figure 3-1 Power line to software synchronization" will be used. The PIT status is represented by the 5 status from the Figure.

1 – Zero cross detected – To detect the zero cross point, an interrupt from GPIO port is used. After the zero cross interrupt occurrs and the previous pulse is successfully generated, the PIT timer is reconfigured to have the required control angle. At that moment, the PIT starts to count. The "`controll_angle`" variable is used to change the actual control angle.

2 - Wait to achieve control angle - The timer status is switched to “wait for pulse” mode. In this mode the microprocessor waits for PIT interrupt.

3 – Control angle achieved – When the required phase angle is achieved, the PIT interrupt occurs. Timer is reconfigured to generate triac gate pulses according to the required pulse width. The “`pulse_wide`” variable is used to change the gate pulse width.

4 - Pulse generate – The gate signal is being generated. The number of pulses depends on PULSES_COUNT macro.

5 - Pulse OK – A pulse is generated successfully. The application waits for the next half wave.

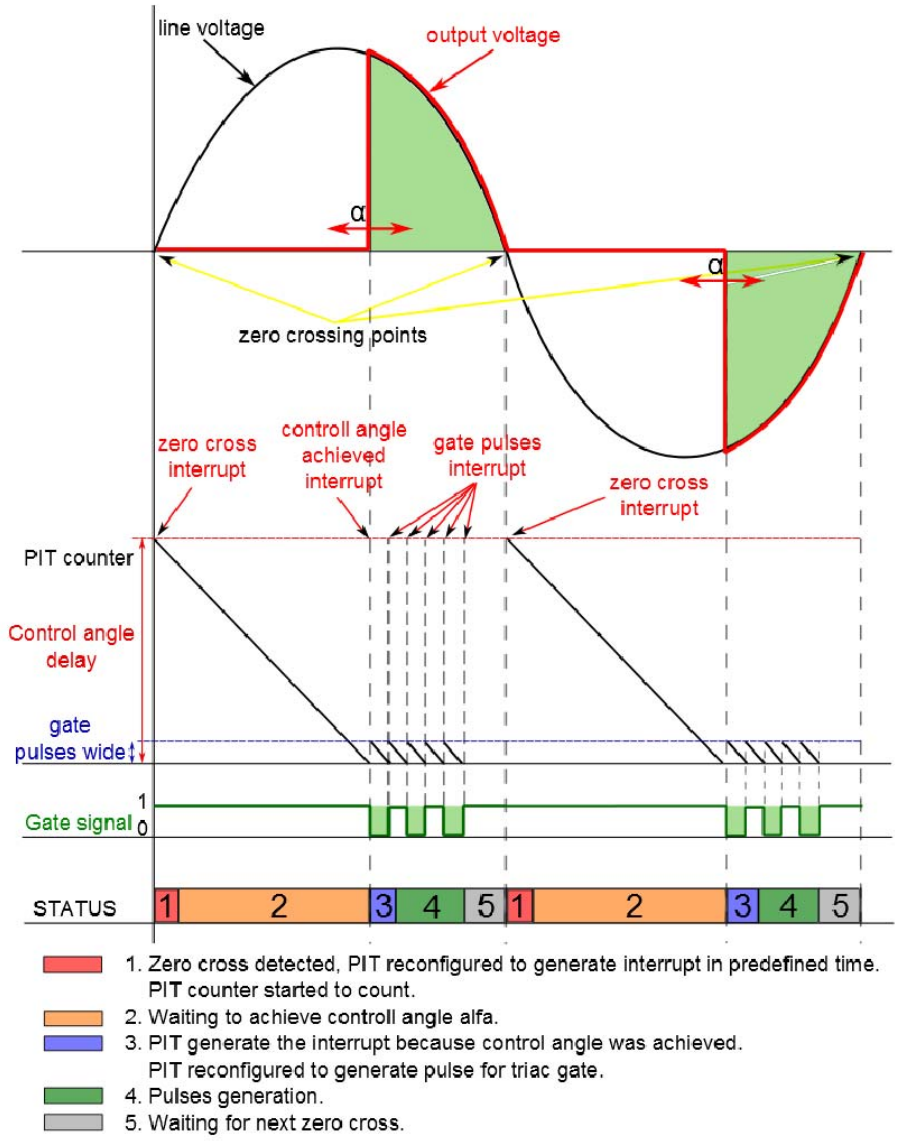

Figure 3-1 Power line to software synchronization

Interrupt routine for zero cross event service:

```
void Zero_Cross_isr (void)
{
  if (pit_state == PULSE_OK) // if previous pulse generated successfully
                                  setup PIT for next trigger
    {
        PIT_LDVAL0 = controll_angle; // for enter new value into PIT, timer
                                        should be disabled and enabled again
        PIT_TCTRL0 = 0;              // timer disable
        PIT_TCTRL0 |= PIT_TCTRL_TEN_MASK | PIT_TCTRL_TIE_MASK;   // timer enable
        pit_state = WAIT_FOR_PULSE;  // change PIT status
    }
  PORTA_ISFR |= (1UL);              // clear flag}
```

Interrupt routine for PIT event service:

The Pit timer is used for two purposes:

1. It generates an interrupt at a predefined time after the line voltage zero cross.

   In this case PIT counter is started in Zero_Cross_isr routine, when the line voltage zero cross is detected.

2. It generates a pulse sequence for triac.

   In this case timer is started in PIT0_isr routine and the interrupt is called periodically until the required pulse count is reached.

```
void PIT0_isr(void)
 {
 switch(pit_state)
  {
   case (WAIT_FOR_PULSE): // start to generate triac pulses after defined phase angle
      {
         PIT_LDVAL0 = pulse_wide;      // set pulse wide for triac gate in PIT cycles
              // for enter new value into PIT, timer should be disabled, flag cleared
              // and PIT enabled again
         PIT_TCTRL0 = 0;                          // timer disable
         PIT_TFLG0 |= PIT_TFLG_TIF_MASK;          // clear flag
         PIT_TCTRL0 |= PIT_TCTRL_TEN_MASK | PIT_TCTRL_TIE_MASK; // timer enable
        if(run == 1) { GPIOC_PCOR |= PORTC_1_2_3_MASK; }    // generate
                                        pulse on 3 pins simultaneously using mask
         pulse_counter = 0;
         pit_state = PULSE_GEN;                   // switch state
         break;
      }
   case (PULSE_GEN):
      {
         if(run == 1) {GPIOC_PTOR |= PORTC_1_2_3_MASK;}  // set pin level on three
                                                    pins at the same time, using mask
         pulse_counter ++;
         if( pulse_counter > PULSES_COUNT)//repeat until required pulse count achieved
            {
              PIT_LDVAL0 = controll_angle;    // after all pulses was generated wait
                                                 for next line voltage half wave
              PIT_TCTRL0 = 0;                 // timer disable
              pit_state = PULSE_OK;
```

```
            }
          break;
        }
    }
 PIT_TFLG0 |= PIT_TFLG_TIF_MASK;              //clear flag
}
```

4 Performance

Table-4-1 Memory usage shows how much memory was needed to run the phase angle drive. A significant part of the memory is still available.

Memory	Available	Used
SRAM	32Kbytes	65 bytes
ROM	256Kbytes	1.5 Kbytes

Table-4-1 Memory usage

"Figure 4-1Current waveform, triac pulses and zero cross detection" shows the final operation of this application. The blue line waveform shows drive input current. The red waveform shows triac gate pulses generated by the microcontroller and the green one shows the detection of the line voltage zero cross.

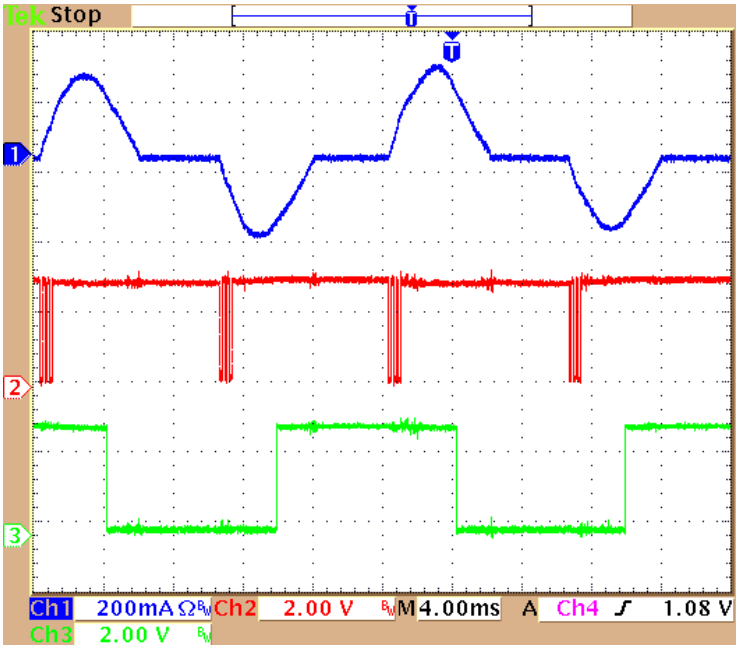

Figure 4-1Current waveform, triac pulses and zero cross detection

5 Conclusion

This application note describes an application that can be used in a low-cost product. The unused memory and performance capacity are still available for other customer purposes. These facts make this application especially suitable for the appliance market.

6 References

1. ***AN1663, Low cost universal motor Sensorless phase angle drive sysytem****, by Ivan Skalka, Andrzej Lara, Freescale Semiconductor, Inc.*

2. ***AN1662, Low cost universal motor phase angle drive sysytem****, by Ivan Skalka, Freescale Semiconductor, Inc.*

3. ***MAC4DCM Data Sheet from ON Semiconductor***

4. ***KL25 Sub-Family Reference Manual*** *by Freescale Semiconductor, Inc.*