MOTOROLA
intelligence everywhere™

digital dna™

Open Loop Universal Motor Chopper Based on the MC68HC908QT4 Microcontroller Reference Design

M68HC08
Microcontrollers

Designer Reference Manual

DRM057
Rev. 0, 1/2004

MOTOROLA.COM/SEMICONDUCTORS

Open Loop Universal Motor Chopper Based on the MC68HC908QT4 Microcontroller

Designer Reference Manual — Rev 0

by: Ladislav Makovic
Motorola Ltd.
Roznov pod Radhostem

WARNING: ***This circuit is powered directly from the mains. It is dangerous to touch any part of the circuit, even if a 0% duty cycle is generated. Do not connect any computer, oscilloscope or development system to this circuit without using an isolation transformer.***

Revision history

To provide the most up-to-date information, the revision of our documents on the World Wide Web will be the most current. Your printed copy may be an earlier revision. To verify you have the latest information available, refer to:

http://www.motorola.com/semiconductors

The following revision history table summarizes changes contained in this document. For your convenience, the page number designators have been linked to the appropriate location.

Revision history

Date	Revision Level	Description	Page Number(s)
1/2004	0	Initial release	N/A

Table of Contents

Table of Contents

Appendix A. Abbreviations and Definitions

List of Figures

List of Tables

Section 1. Introduction

This reference manual describes a real application, which can be used in a low-cost product. Using a microcontroller allows the system to be software programmable by changing defined constants within the application software. Another advantage of using a microcontroller is that the control algorithm can be changed simply, by re-programming the microcontroller with new firmware. The unused memory and some performance capacity remain available for other purposes. These facts make this application particularly suitable for the appliance market.

1.1 Overview of the Application

The application presents a low-cost, open loop universal motor chopper control drive system based on the MC68HC908QT4 microcontroller. A PWM (pulse width modulation) technique is used to adjust the voltage applied to the motor. Modulation of the PWM's duty cycle allows the average value of the voltage applied to the motor to be varied. Compared to a phase angle drive, the chopper drive requires a more complicated power stage, with an input power rectifier, a power switch, and a power fast diode. The advantage of the chopper drive is higher efficiency, with less acoustic noise and better EMC behavior.

1.1.1 Uses of the Application

- Washing machines
- Vacuum cleaners
- Hand tools
- Food processors
- Dishwashers
- Single-phase variable-speed drives
- Low-cost, medium-power applications with variable voltage output

1.1.2 Features

- Capable of supplying universal/DC motors
- Variable output voltage 0-230 V AC rms with defined step and delay ramp
- Maximum output current 6.52 A AC rms (1500 W, 230 V) (short time peak current 8 A)
- Uses PWM technique to modulate output voltage (5.882 kHz switching frequency)
- Capable of supplying general DC inductive/resistive loads
- Electronic switch on/off PWM generation
- Easy to re-program the system behavior

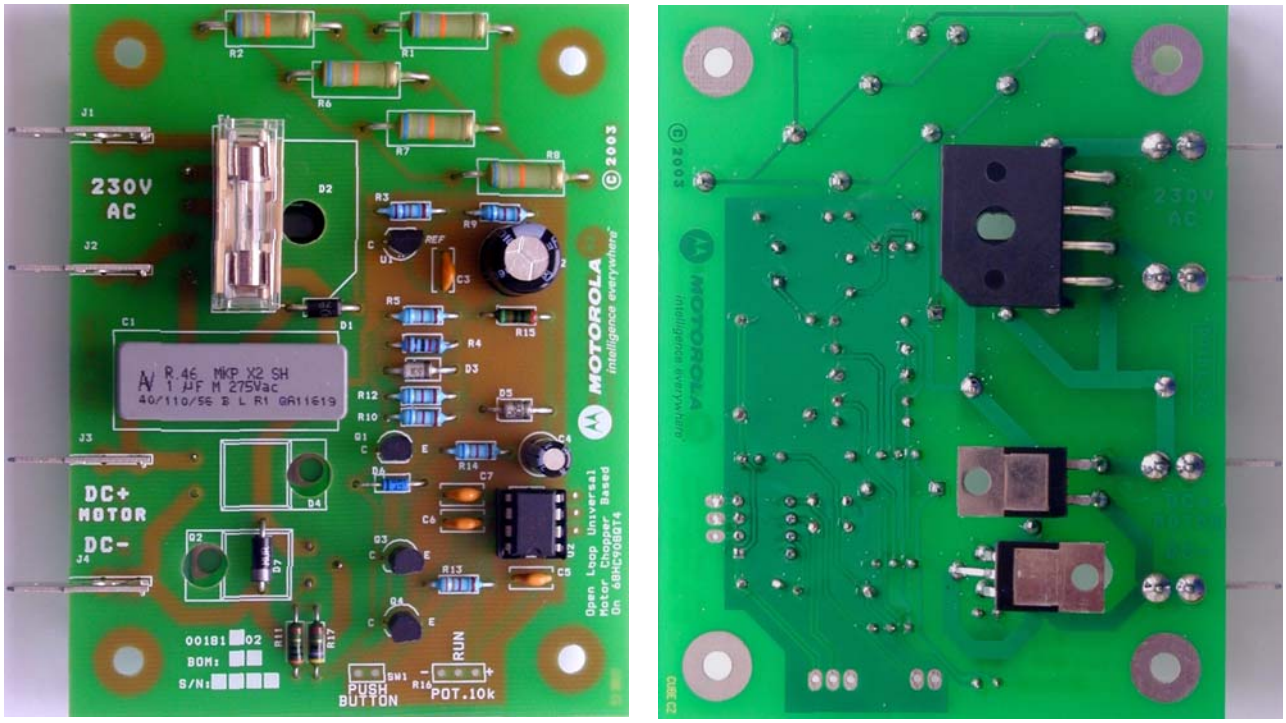

Figure 1-1. The Open Loop PWM Chopper

1.2 Intended Functionality of the Application

This reference manual describes the design of a low-cost chopper motor control drive system based on the MC68HC908QT4 microcontroller, an IGBT (Insulated Gate Bipolar Transistor), and an ultra-fast high voltage (freewheeling) diode. This low-cost single-phase power board is dedicated to universal brushed motors. The universal motor is the most widely used motor in home appliances, such as vacuum cleaners, washers, hand tools, and food processors. The operational mode used in this application is open loop and regulated input motor voltage. The type of motor and its drive have a high impact on many home appliance features, such as cost, size, noise and efficiency. Electronic control usually becomes necessary when variable speed or energy savings are required.

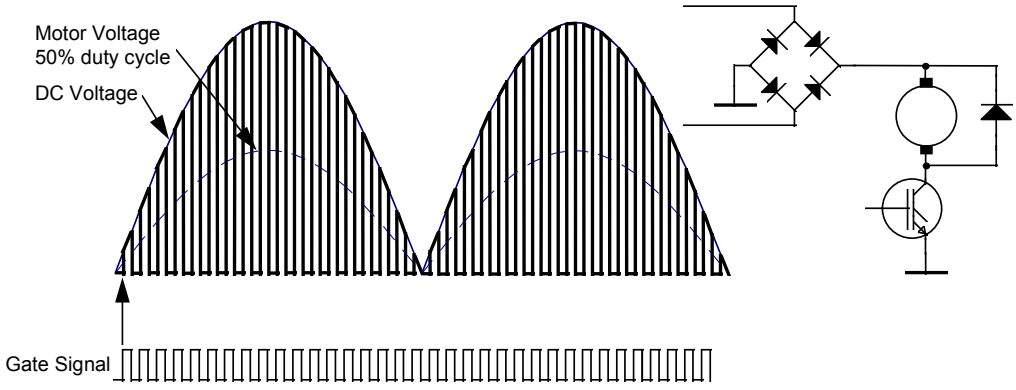

Figure 1-2. The Pulse Width Modulation ‘Chopper’ Technique

Microcontrollers offer the advantages of low cost and attractive design. They can operate with only a few external components, reducing the energy consumption as well as the cost. This circuit was designed as a very simple schematic, using all the features of a simple microcontroller. The microcontroller and this circuit scheme may be used in a wide variety of applications.

The Pulse Width Modulation technique (PWM) is used to adjust the voltage applied to the motor (**Figure 1-2**). Modulation of the PWM's duty cycle allows variation of the average voltage value seen by the motor. This PWM technique is often termed 'chopper', because of the chopped drive signal that is created.

This document also explains how to design the software implementation using an M68HC08 family microcontroller. A low-cost microcontroller such as this is powerful enough to handle the workload necessary for driving a closed loop chopper drive.

1.3 Benefits of Our Solution

Compared to a poorer analog solution, a microcontroller based drive exhibits many advantages. Some of these are listed below.

- Choice of different control algorithms
- Choice of any shape of speed command (acceleration and deceleration phase)
- Easy adaptation to a closed loop control algorithm
- Software can simplify the hardware
- Diagnostic functions
- Open for innovation

1.4 The MC68HC908QT4 Microcontroller

The MC68HC908QT4 is a member of the low-cost, high-performance M68HC08 Family of 8-bit microcontroller units (MCUs). The M68HC08 family is a Complex Instruction Set Computer (CISC), with a Von Neumann architecture. All MCUs in the family use the enhanced M68HC08 central processor unit (CPU08), and are available with a variety of modules, memory sizes and types, and package types.

Some key features of the MC68HC908QT4 are listed below.

- 4K bytes of in-application re-programmable FLASH and 128 bytes of RAM
- 2-channel, 16-bit timer with selectable input capture, output compare, and PWM
- High-performance, easy-to-use M68HC08 CPU
- Trimmable internal oscillator with ± 5% accuracy
- 4-channel, 8-bit analog to digital converter
- Selectable trip point low voltage inhibit (LVI)
- Computer operating properly (COP) timer with auto wake-up from STOP
- Flexible high-current I/O and keyboard interrupts
- Five bidirectional I/O pins and one input-only pin
- Available in 8-pin DIP or SOIC packages

1.5 References

1. PWM_setup.xls - placed into project **[2]** sources.
2. UM_Chopper.mcp - project, debugged under CodeWarrior CW08 V3.0.
3. Schematic, PCB.

Introduction

Section 2. Initial Design Conditions

This section deals with system hardware and software constraints and requirements that must be met by the design.

2.1 System Hardware Requirements

1. The power stage must be able to switch a 1.5 kW load at a line voltage of 230 V.
2. The solution is based on the MC68HC908QT4 microcontroller.
3. No external crystals or resonators are used.
4. Control circuit power supplies are derived from the rectified line voltage, without a transformer. Two DC power supplies are required: a 12 V power supply to feed the IGBT driver; and a 5 V power supply to feed the microcontroller and control circuits.
5. There is no feedback, although the solution allows for its addition.
6. The circuit is electronically switched on and off by the microcontroller $\overline{\text{IRQ}}$ pulse interrupt.
7. The required PWM duty cycle (a potentiometer output voltage) is sensed by the microcontroller's A/D converter.
8. A single layer PCB and through-hole devices are used.

2.2 System Software Requirements

1. The PWM duty cycle is set from 0% to 100% with an accuracy of 1/18 duty cycle.
2. The duty cycle accuracy can be set using a defined constant.
3. The PWM frequency can be set in a given range using a defined constant.

Initial Design Conditions

4. The PWM ramp has 18 steps (duty cycles), and the number of steps can be set using a defined constant.
5. The PWM ramp time duration can be set using a defined constant.
6. The PWM signal can be set to invert the IGBT driver by a defined constant.
7. The required PWM duty cycle is processed by an 8-sample moving average filter.

Section 3. Hardware Design

This section describes the system hardware solution, and provides a block diagram, circuit schematic and PCB (printed circuit board) design.

WARNING: ***This circuit is powered directly from the mains. It is dangerous to touch any part of the circuit, even if a 0% duty cycle is generated. Do not connect any computer, oscilloscope or development system to this circuit without using an isolation transformer.***

3.1 Hardware Description

Figure 3-1 shows a system block diagram of the chopper motor control board. The system comprises the following five major parts.

- Power stage
- 12 V power supply
- Driving stage
- 5 V power supply
- Microcontroller, with user interface.

Figure 3-2 is a schematic diagram of the chopper motor control board.

The power stage consists of the input power bridge rectifier D2 with capacitor C1, the power IGBT transistor Q2, and the freewheeling diode D4. Diode D7 protects the IGBT against negative spikes.

The 12 V power supply contains diode D1, and resistors for decreasing input voltage. This follows on to an adjustable precision shunt regulator TL431, with setting resistors R3 and R9, filtration capacitor C3, and smoothing capacitor C2. The primary function of the 12 V source is to feed the driving stage; a secondary function is to feed the 5 V supply.

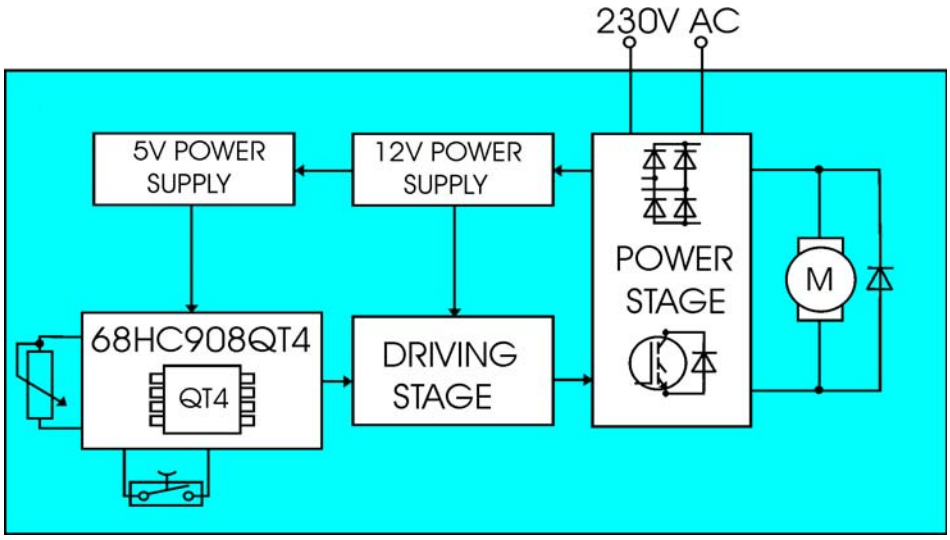

Figure 3-1. Block Diagram

The driving stage switches the power IGBT Q2 on the base of input logic signal from pin 7 of the microcontroller MC68HC908QT4. The IGBT is switched on via resistor R11; it is switched off via resistor R17 and a Schottky diode D6.

The 5 V power supply feeds the microcontroller circuit. To achieve 5 V power supply from a 12 V source, a decreasing resistor R15, 5V1 zener diode D5, and smoothing capacitor C4 are used.

The microcontroller circuit contains the MC68HC908QT4 microcontroller and supporting devices. The A/D converter, pin 6 of the microcontroller, provides the conversion of the input voltage from potentiometer R16 representing the required PWM duty cycle. Start and stop of the PWM signal generation is made on the strength of the interrupt generated by push button SW1 connected to pin 5 ($\overline{IRQ}$) of microprocessor. To filter voltage noise, capacitor C6 (pin 1) is connected to Vdd, C7 is connected to A/D converter input (pin 6), and C7 is connected to $\overline{IRQ}$ (pin 5).

NOTE: *The potentiometer R16 should be connected to the PCB in the required manner. This means that if the potentiometer R16 is in a zero position, a 0% duty cycle is required, and the 0Ω resistance must be between "runner" and "-" of the potentiometer R16.*

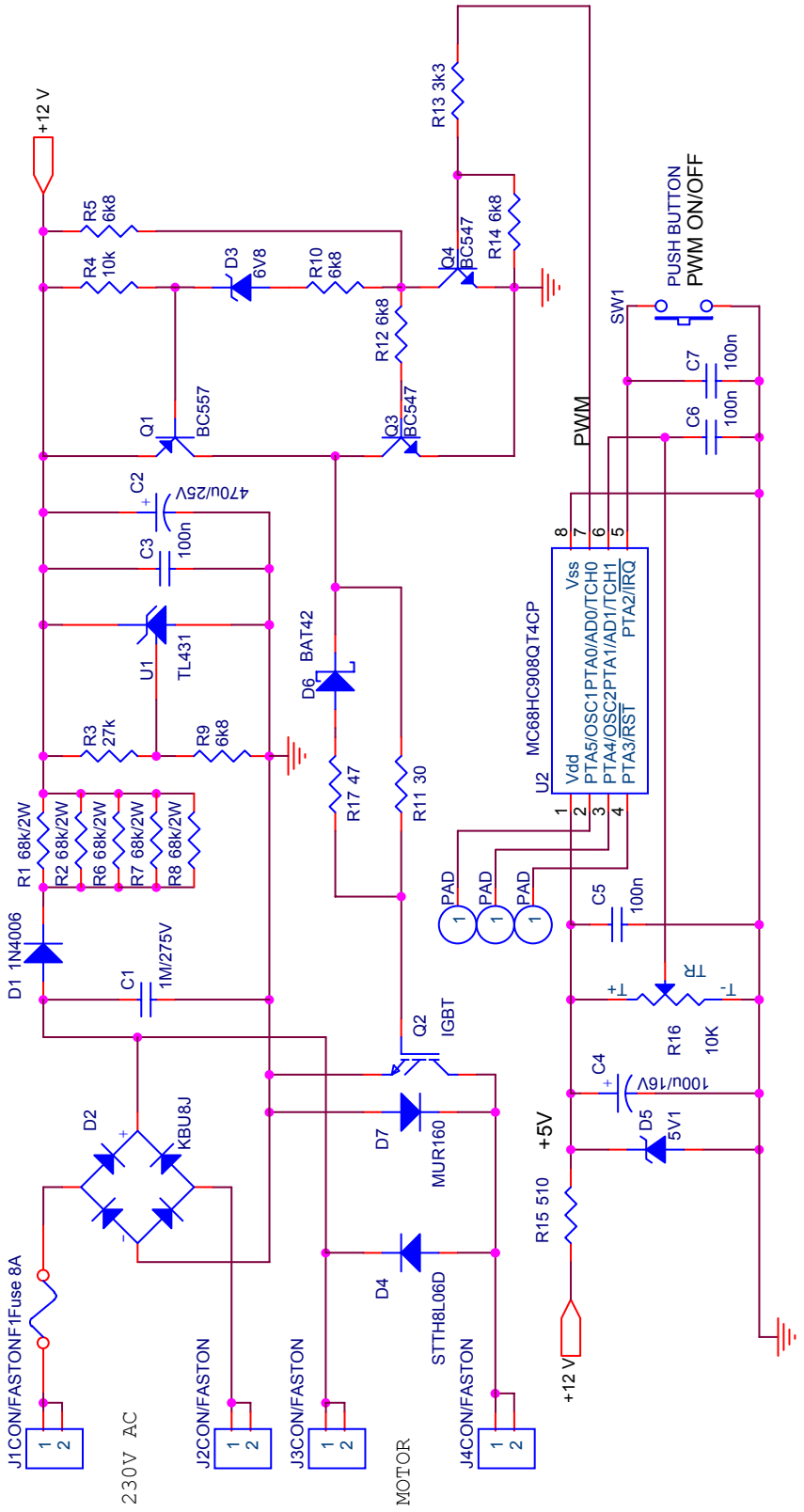

Figure 3-2. Circuit Schematic Diagram

3.2 Parts List

Table 3-1. Parts List

DESIGNATORS	QUANTITY	DESCRIPTION	MANUFACTURER	PART NUMBER
C1	1	1M/275V	ICEL	MKT - X2 CLASS
C2	1	470u/25V ELECTROLYTIC	-	ANY ACCEPTABLE
C3,C5,C6,C7	4	100n/CERAMIC	-	ANY ACCEPTABLE
C4	1	100u/16V ELECTROLYTIC	-	ANY ACCEPTABLE
D1	1	800V/1A	FAIRCHILD	1N4006
D2	1	600V/8A DIODE BRIDGE	FAIRCHILD	KBU8J
D3	1	6V8 ZENER DIODE	-	ANY ACCEPTABLE
D4	1	ULTRAFAST DIODE 600V/8A	ST	STTH8L06D
D5	1	5V1 ZENER DIODE	-	ANY ACCEPTABLE
D6	1	30V/4A SCHOTTKY DIODE	ST	BAT42
D7	1	600V/1A ULTRAFAST DIODE	VISHAY	MUR160
F1	1	FAST FUSE	8A	ANY ACCEPTABLE
J1,J2,J3,J4	4	CONNECTOR FASTON	-	ANY ACCEPTABLE
Q1	1	-45V/-100mA PNP TRANSISTOR	FAIRCHILD	BC557
Q2	1	600V/20A HIGH FREQ. IGBT	ST	STGP20NB60H
Q3,Q4	2	45V/500mA NPN TRANSISTOR	FAIRCHILD	BC547
R1,R2,R6,R7,R8,	5	68k/2W RESISTOR	-	ANY ACCEPTABLE
R3	1	27k/0.6W RESISTOR	-	ANY ACCEPTABLE
R4,R16	2	10k/0.6W RESISTOR	-	ANY ACCEPTABLE
R5,R9,R10,R12,R14	5	6k8/0.6W RESISTOR	-	ANY ACCEPTABLE
R11	1	30/0.6W RESISTOR	-	ANY ACCEPTABLE
R13	1	3k3/0.6W RESISTOR	-	ANY ACCEPTABLE
R15	1	510/0.6W RESISTOR	-	ANY ACCEPTABLE
R17	1	47/0.6W RESISTOR	-	ANY ACCEPTABLE
SW1	1	PUSH BUTTON	-	ANY ACCEPTABLE
U1	1	ADJUSTABLE VOLTAGE REGULATOR	ON SEMICONDUCTOR	TL431
U2	1	MICROCONTROLLER	MOTOROLA	MC68HC908QT4CP

3.3 PCB Design

Figure 3-3 shows the design of the tracking side of the PCB. The design of the component side is shown in Figure 3-4.

High current in the power stage is a limiting factor in the PCB design. Due to this current, the PCB copper thickness must be 0.35 µm at the minimum.

All power devices (rectifier, IGBT, freewheeling diode) and the board are mounted onto the heat sink placed under the board. The size of the heat sink depends on the size of the load, the switching devices used, and PWM frequency.

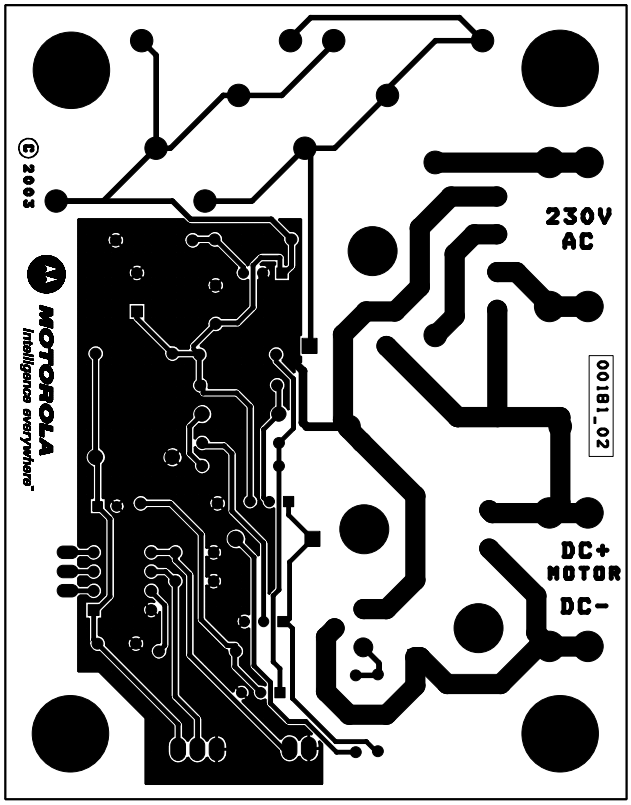

Figure 3-3. PCB — Track Side

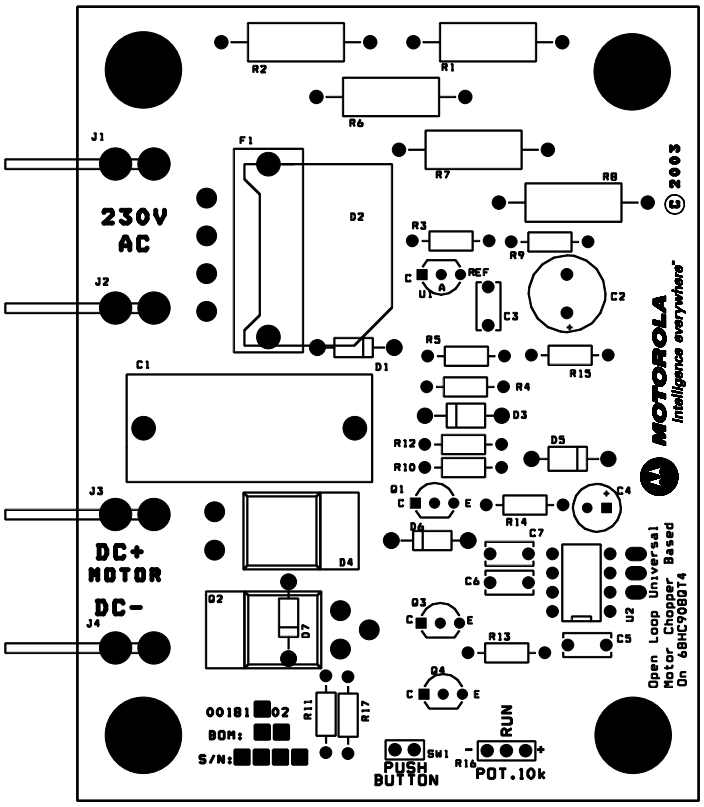

Figure 3-4. PCB — Component Side

Section 4. Software Design

This section describes the design of the driver software blocks. The software is described in terms of:

- Basic software description
- Constants and variables definition
- Flow chart description
- Total microcontroller resource usage

4.1 Basic Software Description

The application controls the universal motor. A chopper control is implemented. The software reads the user interface – the speed potentiometer and start/stop switch. It calculates the desired speed using a speed ramp and, accordingly, the PWM duty cycle is set and PWM is generated on the output. The system can be viewed on three levels (see **Figure 4-1**).

- Pre-programming phase
- Program restart
- Main program loop

During the pre-programming phase, the user can set the program behavior. For this purpose, some constants that determine system behavior are defined. More about constants setup is presented in this section.

Program restart — the program starts when the board is connected to the mains. During this phase, a delay for system stabilization is performed, and the constants set the microcontroller peripheral to the required state.

Software Design

The main program loop is performed as long as the board remains connected to the mains supply.

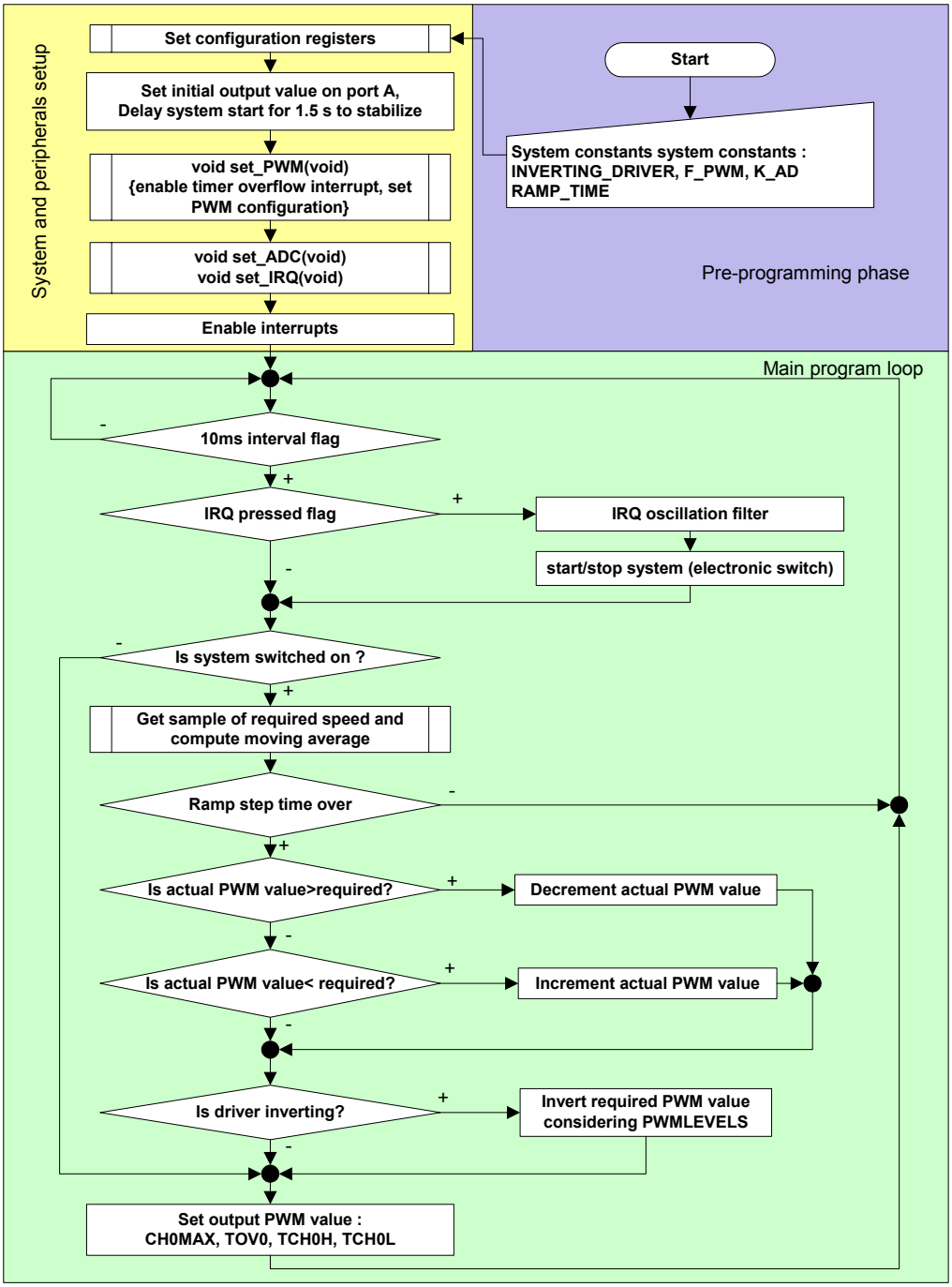

Figure 4-1. Main Program Flow Chart

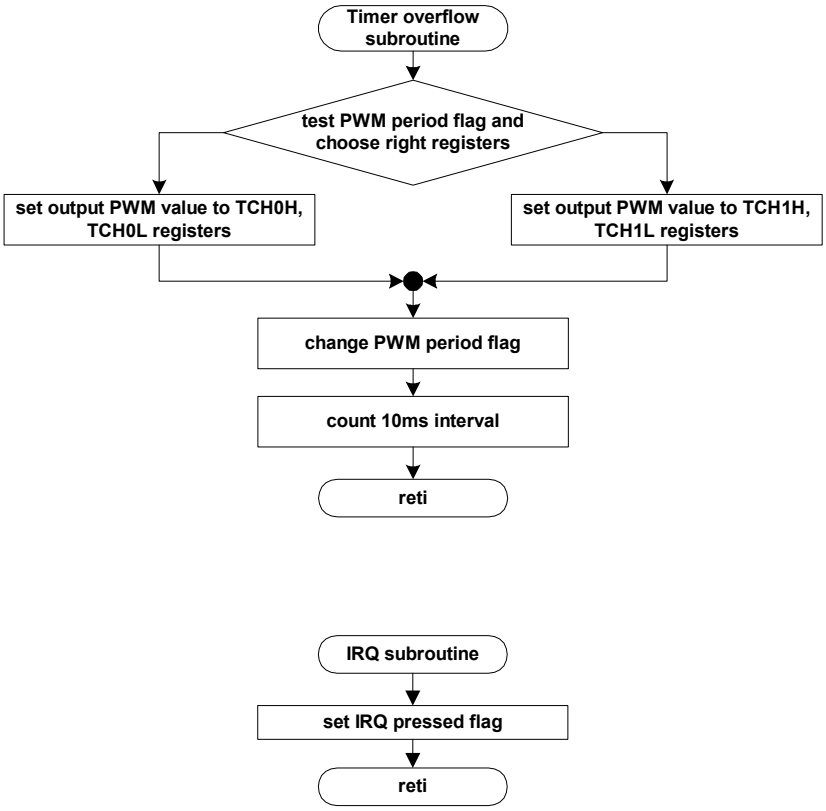

Figure 4-2. Interrupt Subroutine Flow Charts

4.2 Constant and Variable Definitions

This section describes definitions and variables used in the program.

Definitions can be grouped into two categories.

- System setup definitions — defined by programmer/user
- Auxiliary system definitions — dependent on the system setup definitions

4.2.1 System Setup Definitions

These constants are available for the user/programmer to setup system behavior, according to required conditions.

1. `#define PWMLEVELS 18 /*see PWM_setup.xls*/`

   The constant represents the number of possible PWM duty cycles from 0% to 100% duty cycle (see the following note).

   for example: If PWMLEVELS EQU 5 is defined,
   then possible duty cycles are 0/4,1/4, 2/4, 3/4, 4/4.

2. `#define F_PWM 32 /*1,2,4,8,16,32,64*/`

   The constant sets the timer clock prescaler, thus allowing changes to PWM frequency in a given range (see the following note).

3. `#define K_AD 15 /*see PWM_setup.xls*/`

   The constant transforms the value of A/D converter, from interval <0;255> to the interval given by constant PWMLEVELS. Then the required PWM duty cycle is given by the formula:
   PWM = ADC Data register / K_AD (see the following note).

NOTE: *The constants PWMLEVELS, F_PWM, and K_AD can be easily changed by means of the PWM_setup.xls file **[1]**. An example table of PWM setup values for 3.2 MHz bus frequency, plus setup procedure, are presented in section **4.3.2**.*

4. `#define RAMP_TIME 2000 /*[ms]*/`

   The constant defines the ramp duration from 0% PWM to 100% PWM. Unit is one millisecond.

5. `#define INVERTING_DRIVER 0 /*x=x/1*/`

   The constant defines whether a non inverting (<>1) or inverting (=1) IGBT driver is used. This feature makes it easy to apply the program to systems where inverting drivers are used. In this application, a non inverting driver is used.

4.2.2 Auxiliary System Definitions

The following auxiliary system constants are used in the program, and some of them depend upon the system setup definitions.

1. `#define TIME1 10 /*[ms]*/`

   Defines the system timing constant in milliseconds. All actions are timed by their own counters that count the number of defined time intervals.

2. `#define P_1 (PWMLEVELS-1)`

   An auxiliary definition for the following definitions.

3. `#define K_10MS ((TIME1*3200/(P_1*F_PWM))+1)`

   The constant defines the number of timer overflows during the defined timing constant. It depends on:

   i. PWM setting constants:
   - prescaler - F_PWM
   - number of PWM duty cycles - PWM_LEVELS

   ii. timing constant - TIME1

   The expression in the definition is a simplified version of the following expression:

$$K\_10MS = \left((int)\left(\frac{TIME1}{1000} \times \frac{BusFrequency}{((PWMLEVELS-1) \times TimerPrescaler)}\right) + 1\right)$$

4. `#define RAMPSTEPTIME (RAMP_TIME/P_TIME1)`

   The constant defines the time between two PWM levels change.

5. `#define K_RAMP ((RAMPSTEPTIME/TIME1)+1)`

   The constant defines the number of timing cycles between two PWM levels change.

6. `#define MAXPWMVAL P_1`

   The variable represents the maximal PWM duty cycle that can be set for a chosen system setup.

   MAXPWMVAL = PWMLEVELS - 1

4.2.3 System Variables

This section describes static global variables used in the program.

1. **`static Byte ucOn_Off_PWM`**

   The variable on_off_PWM is defined to realize an electronic switch. This variable switches on and off PWM generated on a given pin. If the value of this variable is 0x00, then PWM is off and a 0% duty cycle is generated. If the value of the constant is negated and its value is 0xFF, then the required PWM duty cycle is generated. The variable negation depends on $\overline{\text{IRQ}}$ interrupt (pressing push button SW1).

2. **`static Byte ucAct_PWM_val`**

   The variable represents the value that tracks the required PWM duty cycle by the given ramp.

3. **`static Byte ucEnd_PWM_val`**

   The variable represents the required PWM duty cycle. It is the result of the moving average from A/D conversions.

4. **`static Byte ucADCbuffer[8]`**

   The buffer saves the last eight A/D conversion values for use in moving average calculations.

5. **`static Byte uc_p_ADCbuff`**

   The variable points to the actual position in the ucADCbuffer.

6. **`static Word uwSum`**

   The variable represents the sum of all values of the ucADCbuffer.

7. **`static Byte ucAct_Period`**

   The variable enables writing the required PWM duty cycle value into the appropriate TCHxH, TCHxL registers to perform a buffered PWM generation. It symbolizes the just finished period (0x00 => TCH0H, TCH0L; 0xFF => TCH1H, TCH1L), The new duty cycle value can be written into thus marked registers.

8. **`static Byte ucOut_PWM`**

   The variable represents the actual resultant value of the PWM duty cycle that can be written into the registers TCHxH, TCHxL.

9. **`static Byte ucIRQ_Flag`**

   This flag is set (0xFF) when the $\overline{\text{IRQ}}$ interrupt is generated and it starts counting three 10 ms intervals by means of the variable ucCountIRQ_10ms. Having finished counting, the variable is cleared and the $\overline{\text{IRQ}}$ pin is tested again (button oscillations filter). If its value is still log.0, the variable ucOn_Off_PWM is inverted. Otherwise, no action is performed.

10. **`static Byte uc10ms_Flag`**

    The flag informs the system that the 10 ms interval has run. This happens when the variable ucCount_10ms, incremented at every timer overflow, reaches constant K_10MS.

11. **`static Byte ucCount_10ms`**

    The variable represents the number of timer overflows required to generate a 10 ms interval. When the variable value reaches constant K_10MS, the flag uc10ms_Flag is set.

12. **`static Byte ucCountIRQ_10ms`**

    The variable represents the number of 10 ms intervals required to perform an $\overline{\text{IRQ}}$ button oscillation filter. Counting starts after the $\overline{\text{IRQ}}$ interrupt is generated. When the value of the variable is 3 (30 ms oscillation filter), and the $\overline{\text{IRQ}}$ pin still has the value log.0, the variable ucOn_Off_PWM (software start/stop switch) is inverted.

13. **`static Word uwCount_RampStep_10ms;`**

    The variable represents the number of 10 ms intervals required to perform PWM ramp. When the value of the variable reaches constant K_RAMP, the PWM duty cycle can be changed.

4.3 Flow Chart Description

The software flow charts can be seen in **Figure 4-2.** The software solution can be divided into two parts — system plus peripherals setup and main loop. Source code can be found in **[2]**.

4.3.1 System Setup

Prior to running the main program loop the system setup sets the microcontroller operation mode and peripherals. The quickest way to set the microcontroller peripherals is to use the auxiliary excel sheet **[1]**, or for 3.2 MHz bus frequency, use **Figure 4-3.**

To achieve the required system behavior and communication with peripherals, the following registers and initial values are set. This action can be seen in the flow chart as subroutine calls.

1. *Set configuration registers - void set_OSC(void)*

   The microcontroller configuration registers Config2 and Config1 initialize the microcontroller to the user defined option. They can be written only once after each reset.

   The register Config1 sets:

   – disable COP

   – disable STOP instruction

   – disable LVI module

   and the register Config2 sets:

   – internal oscillator

   – interrupt request function active in pin (to start and stop the system by pressing push button SW1) (see **Figure 3-2**)

   – IRQPUD must be 0 to connect the internal pull-up resistor between $\overline{\text{IRQ}}$ pin and Vdd.

2. *Set ADC - void Set_ADC(void)*

   This subroutine sets the A/D converter clock.

   ADC clock = bus clock / prescaler.

The recommended value for the ADC clock is 1 MHz.

3. *Buffered PWM setup - void Set_PWM(void)*

   The subroutine initializes the PWM values, sets 0% duty cycle on the PWM output, and sets the variables for the moving average computation.
   On the strength of the constant F_PWM, the subroutine sets the prescaler of the timer clock and the buffered PWM in the TSC register. Subsequently, it sets the TMODH and TMODL registers according to the value of the constant PWMLEVELS. Register TMODH is cleared, and the value of PWMLEVELS-2 is written into register TMODL.

   For a better understanding of PWM setup and logic, see section **4.3.2 PWM Duty Cycle Setup Logic**.

   Moreover, timer overflow interrupt is enabled to perform buffered PWM and system timing.

4. *$\overline{IRQ}$ interrupt setup - void set _IRQ(void)*

   The subroutine sets $\overline{IRQ}$ interrupt on a falling edge.

4.3.2 PWM Duty Cycle Setup Logic

For setting unbuffered PWM, the timer registers must be set correctly.

Consider five levels of PWM and a non-inverting IGBT driver. That means the PWM duty cycle can be set to {0/4, 1/4, 2/4, 3/4, 4/4} *100%. Thus, the required PWM value can be 0, 1, 2, 3, 4.

According to MC68HC908QT4 data sheet PWM duty cycle:

- 0/4*100% (required PWM value 0). Requires clearing bits CHxMAX and TOVx in the register TSCx.
- 1/4*100% (required PWM value 1). Requires TOVx = 1, CHxMAX = 0, TCHxH = 0, TCHxL = required PWM value -1 = 0
- 2/4*100% (required PWM value 2). Requires TOVx = 1, CHxMAX = 0, TCHxH = 0, TCHxL = required PWM value -1 = 1

- 3/4*100% (required PWM value 3). Requires TOVx = 1, CHxMAX = 0, TCHxH = 0, TCHxL = required PWM value -1 = 2
- 4/4*100% (required PWM value 4). Requires setting bits CHxMAX and TOVx in the register TSCx.

If the constant `INVERTING_DRIVER` selects an inverting driver, the duty cycle percentage is in an inverse order.

Timer modulo register TMODL must be set to value PWMLEVELS-2 to ensure that the timer counts PWMLEVELS-1 times until it overflows. The PWM frequency is affected by the setup of the prescaler value in the TSC register. The constant F_PWM represents the prescaler value. Consequently, the PWM frequency is given by the equation:

$$\text{PWM frequency} = \left(\frac{\text{BusFrequency}}{(\text{PWMLEVELS} - 1) \times \text{TimerPrescaler}}\right) \qquad [\text{Hz; Hz, -, -}]$$

The timer prescaler is represented by the constant F_PWM.

As an example, **Table 4-3** presents the PWM setup for 3.2 MHz bus frequency. The sequence of constants selection is as follows:

1. Consider a PWM with an accuracy of 18 PWM duty cycles.

   (PWMLEVELS = 18 => 0/17,1/17,.....,16/17,17/17)
2. In the same line, choose one of the possible frequencies (for example, 5882 Hz). (*Make sure that the choice of PWM frequency is feasible for the system — the limiting factor is the IGBT switching frequency.*)
3. From the same column, read timer prescaler (bits PS2,PS1,PS0 of the register TSC). In this application, the constant K_AD represents the prescaler value of the timer.
4. If the duty cycle setup depends on the value from the A/D converter, the resultant A/D conversion value for TCHcH, TCHxL registers must be divided by the value K_AD read from the line given by PWMLEVELS.

As can be seen, another approach comes from the PWM frequency definition and the consequential configuration of the other parameters given by the equation given above.

f bus=	3,2	MHz
T bus =	3,125E-07	s

F_PWM (binary) {PS2,PS1,PS0}		000	001	010	011	100	101	110	
F_PWM clock devider		1	2	4	8	16	32	64	
PWMLEVELS	AD MAX	↓↓↓ available PWM frequencies [Hz] ↓↓↓							K_AD
3	2	1600,000	800,000	400,000	200,000	100,000	50,000	25,000	86
4	3	1066,667	533,333	266,667	133,333	66,667	33,333	16,667	64
5	4	800,000	400,000	200,000	100,000	50,000	25,000	12,500	52
6	5	640,000	320,000	160,000	80,000	40,000	20,000	10,000	48
7	6	533,333	266,667	133,333	66,667	33,333	16,667	8,333	40
8	7	457,143	228,571	114,286	57,143	28,571	14,286	7,143	32
9	8	400,000	200,000	100,000	50,000	25,000	12,500	6,250	29
10	9	355,556	177,778	88,889	44,444	22,222	11,111	5,556	26
11	10	320,000	160,000	80,000	40,000	20,000	10,000	5,000	24
12	11	290,909	145,455	72,727	36,364	18,182	9,091	4,545	22
13	12	266,667	133,333	66,667	33,333	16,667	8,333	4,167	20
14	13	246,154	123,077	61,538	30,769	15,385	7,692	3,846	19
15	14	228,571	114,286	57,143	28,571	14,286	7,143	3,571	18
16	15	213,333	106,667	53,333	26,667	13,333	6,667	3,333	16
18	17	188,235	94,118	47,059	23,529	11,765	5,882	2,941	15
19	18	177,778	88,889	44,444	22,222	11,111	5,556	2,778	14
20	19	168,421	84,211	42,105	21,053	10,526	5,263	2,632	13
22	21	152,381	76,190	38,095	19,048	9,524	4,762	2,381	12
24	23	139,130	69,565	34,783	17,391	8,696	4,348	2,174	11
26	25	128,000	64,000	32,000	16,000	8,000	4,000	2,000	10
29	28	114,286	57,143	28,571	14,286	7,143	3,571	1,786	9
32	31	103,226	51,613	25,806	12,903	6,452	3,226	1,613	8
37	36	88,889	44,444	22,222	11,111	5,556	2,778	1,389	7
43	42	76,190	38,095	19,048	9,524	4,762	2,381	1,190	6
52	51	62,745	31,373	15,686	7,843	3,922	1,961	0,980	5
64	63	50,794	25,397	12,698	6,349	3,175	1,587	0,794	4
86	85	37,647	18,824	9,412	4,706	2,353	1,176	0,588	3
128	127	25,197	12,598	6,299	3,150	1,575	0,787	0,394	2
256	255	12,549	6,275	3,137	1,569	0,784	0,392	0,196	1

Figure 4-3. PWM Setup for 3.2 MHz Bus Frequency

4.3.3 Main Program Loop

The functions of the main program loop can be summarized as follows.

- System timing.

  The whole system is timed in 10 ms intervals given by the definition of the constant K_10MS.

- PWM switch on/off testing.

  Testing consists of comparison value of the variable ucOn_Off_PWM. If the value is 0, then a 0% PWM duty cycle is generated immediately, and both the actual and required PWM values are set to 0. If the value is 0xFF, then the generated PWM duty cycle tracks the required PWM duty cycle with a given ramp. The variable ucOn_Off_PWM is hung on $\overline{\text{IRQ}}$ interrupt. When an interrupt is generated, the variable ucIRQ_Flag is set. Then, after a given time repeated $\overline{\text{IRQ}}$ pin checking eliminates push button oscillations. If the value of the $\overline{\text{IRQ}}$ pin is still log.0, the complement of the variable ucOn_Off_PWM is calculated.

- A/D conversion.

- Required PWM duty cycle tracking by the actually generated PWM duty cycle with a delay ramp. The tracking is performed by means of the subroutine countADCavg() that computes the moving average from eight consecutive A/D converter values.

4.4 Microcontroller Resource Usage

4.4.1 ROM and RAM Usage

Table 4-1 shows software memory usage. A significant part of the memory is still available.

Table 4-1. Memory Usage

Memory	Available on QT4	Used
ROM	4096 Bytes	680 Bytes
RAM	128 Bytes	23 Bytes

4.4.2 I/O Usage

Table 4-2 summarizes the use of the I/O pins. It can be seen that three pins are still available.

Table 4-2. I/O Usage

I/O pin	Direction	Purpose
PTA0/AD1/TCH0/KBI0	OUTPUT	TCH0 (PWM)
PTA1/AD1/TCH1/KBI1	INPUT	AD1 (INPUT SPEED)
PTA2/$\overline{\text{IRQ}}$/KBI2	INPUT	$\overline{\text{IRQ}}$ (ON/OFF PWM)
PTA3/\RST/KBI3	UNUSED	UNUSED
PTA4/OSC2/AD2/KBI4	UNUSED	UNUSED
PTA5/OSC1/AD3/KBI5	UNUSED	UNUSED

Software Design

Section 5. System Setup

WARNING: ***This circuit is powered directly from the mains. It is dangerous to touch any part of the circuit, even if a 0% duty cycle is generated. Do not connect any computer, oscilloscope or development system to this circuit without using an isolation transformer.***

5.1 Hardware Setup

The hardware setup is very simple. Line voltage 230 V AC must be connected to the connectors labelled “230V AC“, and motor must be connected to the connectors labelled “DC+“ and “DC-”.

The potentiometer R16 is connected to the pads labelled “POT.10k“. For the right potentiometer connection, see the note on page **18**.

The push button SW1 is connected to the pads labelled “PUSH BUTTON”.

5.2 Software Setup

The software setup is covered in section 4.2.1 System Setup Definitions.

5.3 Required Software Tools

Metrowerks CodeWarrior for MC68HC08 microcontrollers version 3.0, or later is required for the development of the application.

5.4 Building and Uploading the Application

The application software is delivered as UM_Chopper_QY4.mcp project file with C-source and header files UM_Chopper_QY4.c and UM_Chopper_QY4.h.

The executable s19 file can be created using Metrowerks CodeWarrior.

5.5 Executing the Application

The application is prepared for operation when connected to a power supply and the 1.5 s software delay finishes. The delay is included to allow the system to stabilize after the mains power is connected, to prevent unwanted PWM generation.

Subsequently, the START/STOP push button SW1 can be pressed to start PWM generation. The PWM duty cycle is set by the potentiometer R16 with a given ramp.

Appendix A. Abbreviations and Definitions

AC	alternating current.
A/D converter, ADC	analog to digital converter: the module is a 4-channel, multiplexed-input successive-approximation analog to digital converter.
Buffered	a system that uses two registers for data; new data is read alternately.
Bus clock	the clock signal on the CPU's internal bus. See "CPU cycles".
Clear	to change a bit from logic 1 to logic 0; the opposite of to set.
Clock	a square wave signal used to synchronize events in a computer.
COP	computer operating properly.
CPU	central processing unit.
CPU cycles	a CPU cycle is one period of the internal bus clock, normally derived by dividing a crystal oscillator source by two or more so the high and low times will be equal. The length of time required to execute an instruction is measured in CPU clock cycles.
CISC	complex instruction set computer.
Cycle time	The period of the operating frequency: tcyc = 1/fop.
DC	direct current.
Duty cycle	the ratio of the amount of time the signal is on to the time it is off. Duty cycle is usually represented by a percentage.
EMC	electromagnetic compatibility.
Interrupt	a temporary break in the sequential execution of a program to respond to signals from peripheral devices by executing a subroutine.
Interrupt request	a signal from a peripheral to the CPU intended to cause the CPU to execute a subroutine.
IGBT	insulated gate bipolar transistor.
I/O	input/output.
Logic 1	a voltage level approximately equal to the input power voltage (Vdd).

Abbreviations and Definitions

Logic 0	a voltage level approximately equal to the ground voltage (Vss).
LVI	low voltage inhibit.
MCU	microcontroller unit: a complete computer system, including a CPU, memory, a clock oscillator, and input/output (I/O) on a single integrated circuit.
Oscillator	a circuit that produces a constant frequency square wave that is used by the computer as a timing and sequencing reference.
Overflow	a quantity that is too large to be contained in one byte or one word.
PCB	printed circuit board.
Peripheral	a circuit not under direct CPU control.
Prescaler	a circuit that divides input signal by given value.
Program	a set of computer instructions that cause a computer to perform a desired operation or operations.
PWM	pulse width modulation: controlled variation (modulation) of the pulse width of a signal with a constant frequency.
PWM period	the duration of one complete cycle of a PWM waveform.
RAM	random access memory. All RAM locations can be read or written by the CPU. The contents of a RAM memory location remain valid until the CPU writes a different value or until power is turned off.
Register	a circuit that stores a group of bits.
Reset	to force a device to a known condition.
Set	to change a bit from logic 0 to logic 1; opposite of clear.
Software	instructions and data that control the operation of a microcontroller.
Subroutine	a sequence of instructions to be used more than once in the course of a program. The last instruction in a subroutine is a return from subroutine (RTS) instruction. At each place in the main program where the subroutine instructions are needed, a jump or branch to subroutine (JSR or BSR) instruction is used to call the subroutine. The CPU leaves the flow of the main program to execute the instructions in the subroutine. When the RTS instruction is executed, the CPU returns to the main program where it left off.
Timer	a module used to relate events in a system to a point in time.
Unbuffered	a system that uses only one register for data; new data overwrites current data.
Variable	a value that changes during the course of program execution.
Word	a set of two bytes (16 bits).

HOW TO REACH US:

USA/EUROPE/LOCATIONS NOT LISTED:

Motorola Literature Distribution;
P.O. Box 5405, Denver, Colorado 80217
1-303-675-2140 or 1-800-441-2447

JAPAN:

Motorola Japan Ltd.; SPS, Technical Information Center,
3-20-1, Minami-Azabu Minato-ku, Tokyo 106-8573 Japan
81-3-3440-3569

ASIA/PACIFIC:

Motorola Semiconductors H.K. Ltd.;
Silicon Harbour Centre, 2 Dai King Street,
Tai Po Industrial Estate, Tai Po, N.T., Hong Kong
852-26668334

TECHNICAL INFORMATION CENTER:

1-800-521-6274

HOME PAGE:

http://motorola.com/semiconductors

MOTOROLA

DRM057