

# **F<sup>2</sup>MC-16FX FAMILY**

## **16-BIT MICROCONTROLLER**

---

# **MOTORDRIVER WITH IRF7389**

## **APPLICATION NOTE**

## Revision History

Date	Issue
2009-09-07	V1.0, MSc, First version

This document contains 10 pages.

## Warranty and Disclaimer

To the maximum extent permitted by applicable law, Fujitsu Microelectronics Europe GmbH restricts its warranties and its liability for **all products delivered free of charge** (eg. software include or header files, application examples, target boards, evaluation boards, engineering samples of IC's etc.), its performance and any consequential damages, on the use of the Product in accordance with (i) the terms of the License Agreement and the Sale and Purchase Agreement under which agreements the Product has been delivered, (ii) the technical descriptions and (iii) all accompanying written materials. In addition, to the maximum extent permitted by applicable law, Fujitsu Microelectronics Europe GmbH disclaims all warranties and liabilities for the performance of the Product and any consequential damages in cases of unauthorised decompiling and/or reverse engineering and/or disassembling. **Note, all these products are intended and must only be used in an evaluation laboratory environment.**

1. Fujitsu Microelectronics Europe GmbH warrants that the Product will perform substantially in accordance with the accompanying written materials for a period of 90 days from the date of receipt by the customer. Concerning the hardware components of the Product, Fujitsu Microelectronics Europe GmbH warrants that the Product will be free from defects in material and workmanship under use and service as specified in the accompanying written materials for a duration of 1 year from the date of receipt by the customer.
2. Should a Product turn out to be defect, Fujitsu Microelectronics Europe GmbH's entire liability and the customer's exclusive remedy shall be, at Fujitsu Microelectronics Europe GmbH's sole discretion, either return of the purchase price and the license fee, or replacement of the Product or parts thereof, if the Product is returned to Fujitsu Microelectronics Europe GmbH in original packing and without further defects resulting from the customer's use or the transport. However, this warranty is excluded if the defect has resulted from an accident not attributable to Fujitsu Microelectronics Europe GmbH, or abuse or misapplication attributable to the customer or any other third party not relating to Fujitsu Microelectronics Europe GmbH.
3. To the maximum extent permitted by applicable law Fujitsu Microelectronics Europe GmbH disclaims all other warranties, whether expressed or implied, in particular, but not limited to, warranties of merchantability and fitness for a particular purpose for which the Product is not designated.
4. To the maximum extent permitted by applicable law, Fujitsu Microelectronics Europe GmbH's and its suppliers' liability is restricted to intention and gross negligence.

### **NO LIABILITY FOR CONSEQUENTIAL DAMAGES**

**To the maximum extent permitted by applicable law, in no event shall Fujitsu Microelectronics Europe GmbH and its suppliers be liable for any damages whatsoever (including but without limitation, consequential and/or indirect damages for personal injury, assets of substantial value, loss of profits, interruption of business operation, loss of information, or any other monetary or pecuniary loss) arising from the use of the Product.**

Should one of the above stipulations be or become invalid and/or unenforceable, the remaining stipulations shall stay in full effect

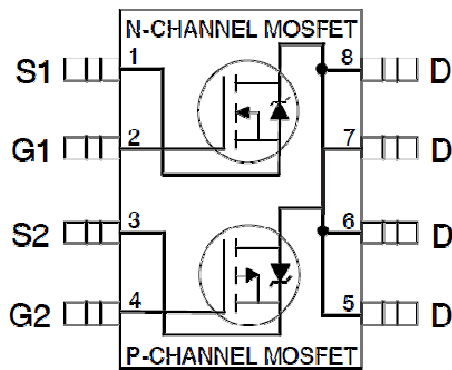
## Contents

<b>REVISION HISTORY</b> .....	<b>2</b>
<b>WARRANTY AND DISCLAIMER</b> .....	<b>3</b>
<b>CONTENTS</b> .....	<b>4</b>
<b>1 INTRODUCTION</b> .....	<b>5</b>
1.1 Overview .....	5
<b>2 HARDWARE IMPLEMENTATION</b> .....	<b>6</b>
2.1 Example Circuit .....	6
2.2 Example PCB Layout .....	7
<b>3 SOFTWARE EXAMPLE</b> .....	<b>8</b>
3.1 Initialisations .....	8
3.2 Motor Functions .....	9
3.3 Sample Program .....	9
<b>4 APPENDIX</b> .....	<b>10</b>
4.1 International Rectifier .....	10
4.2 Fujitsu Microelectronics .....	10
4.3 Additional Websites: .....	10

# 1 Introduction

## 1.1 Overview

This application note gives an idea how to simply drive a DC motor with two IRF7389 ICs as motor bridge. Goals are cheap small motor drivers that support left/right turn of DC motors and speed control via PWM. Following schematic view is given by the datasheet of the IRF7389, which shows two MOSFETS in one package:



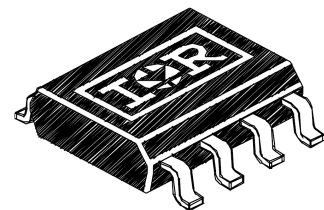
Top View

For detailed information see the datasheet (see chapter 4)

**Note:**

It has to be considered that a motor is an inductive load that produces high voltage- and current-peaks depends on the motor voltage.

The IRF7389 is available as SMD SO-8 package and saves space on a PCB. Heating sinks can be directly designed on the PCB layout.



For implementing a PWM signal with a Fujitsu 16FX MCU there are different possibilities. Some typical hardware macros which are used for PWM generation are the Output Capture Unit (OCU) and the Programmable Pulse Generator (PPG). In this example the PPG is used to generate a PWM signal.

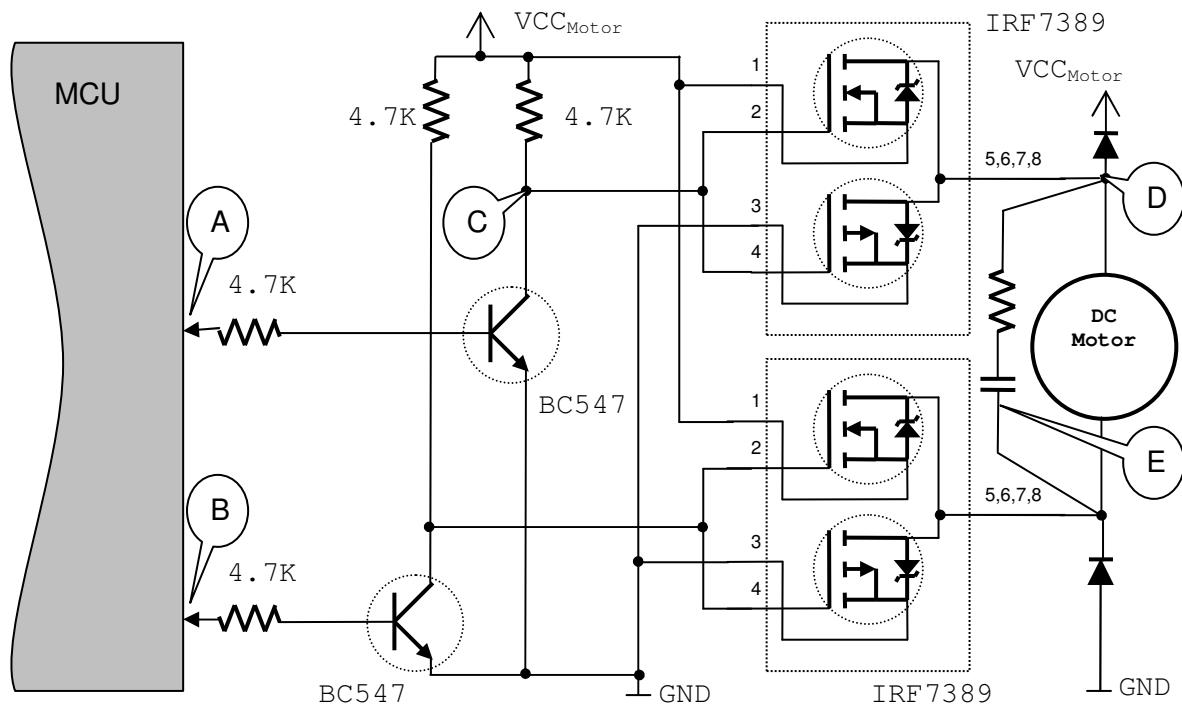
This example should be also easily ported to other Fujitsu MCU families.

## 2 Hardware Implementation

### 2.1 Example Circuit

For a complete motor bridge, two IRF7389 ICs are required. The following circuit gives an idea how to implement a very simple motor bridge. This bridge uses less I/O-pins to drive the DC motor. This has disadvantages while using a PWM to control the motor speed: For every channel or half-bridge, two capacitors have to be charged or discharged while changing the logic value. This means, that only slow PWMs are supported, because higher frequencies increases the dynamic power dissipation, which decreases the efficiency. The datasheet describes values about 1MHz for a MOSFET. In inverter mode this frequency should be divided by two, so the maximum would be under 500 KHz. With this circuit the motor can be driven left or right direction. Example:

- A=0; B=0     =>     Motor is in stop mode
- A=1; B=0     =>     Motor rotates left (A can be used also in slow PWM mode)
- A=0; B=1     =>     Motor rotates right (B can be used also in slow PWM mode)
- A=1; B=1     =>     Motor is in stop mode

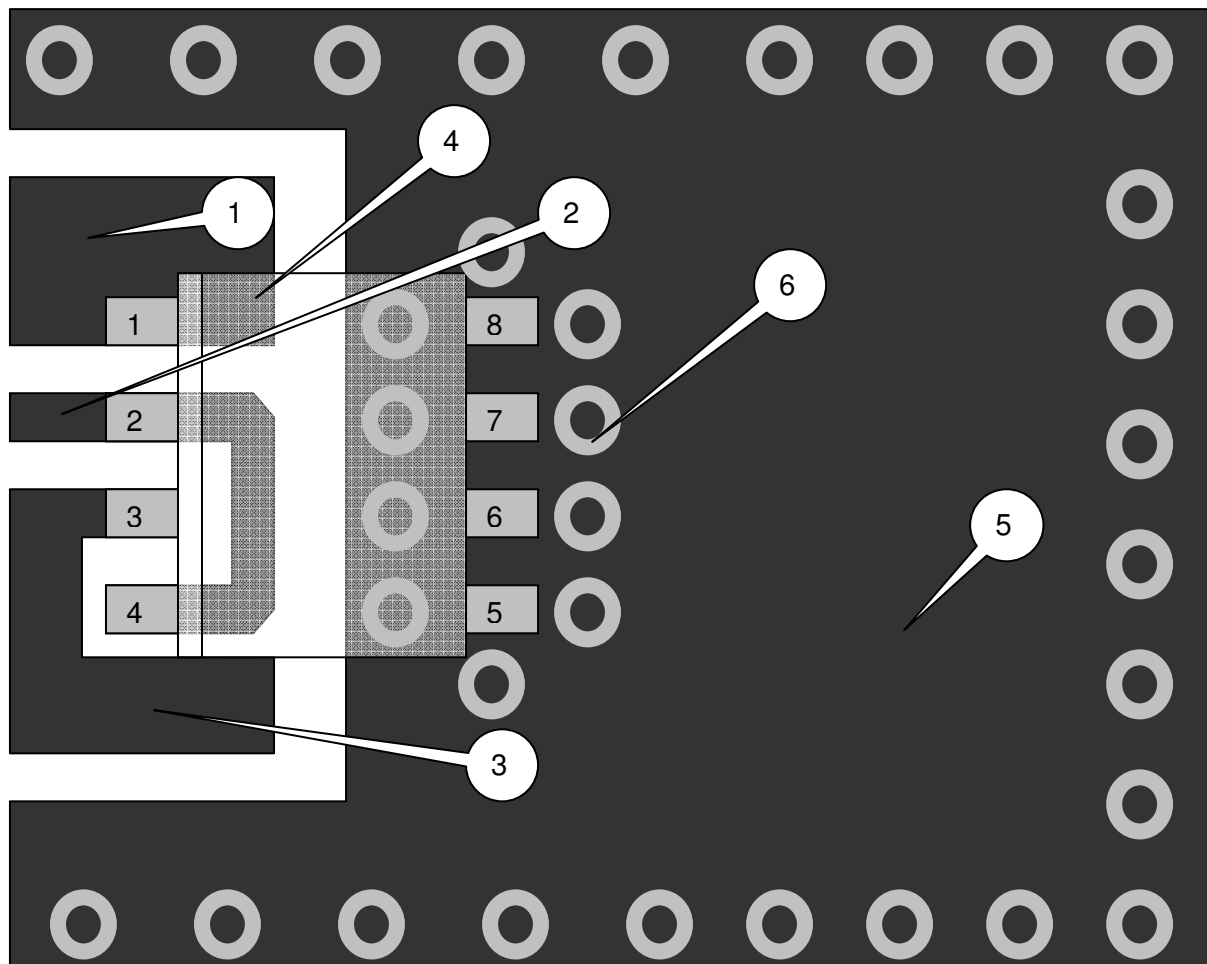


To reduce noise in the motor supply voltage, a RC filter depending of the PWM frequency (E) is added to the Motor. Diodes reminding the risk of destroying the MOSFETs or other semiconductors in the whole circuit including the MCU! In the circuit both IRF7389 are connected as an inverter. If one of the two IRF7389 ICs is considered, every signal that is connected to (C) will be inverted and amplified at (D). To have a good  $V_{GS}$  (C), the control pins from the MCU (A) and (B) are preamplified and inverted by the npn-transistor, pulled up by a 4.7KOhm resistor to  $V_{CC_{Motor}}$ .

## 2.2 Example PCB Layout

In the following view an aperture of an example PCB layout is shown in top view and shows only one driver IRF7389 (4). The critical components in PCB layout are the heat sink (5) and the voltage wires (1)/(3) to the MOSFET drivers. If possible, the heat sink (5) has to be big as possible and should use more than one layer. To connect the different heating sink layers, interlayer connections (6) are used.

Pin 2 of the IC (2) is connected to a NPN transistor pulled up by a resistor to VCC-Motor which is not shown in this aperture ((C) in schematic). To have low resistance and afford high currents from power supply, wires from VCC-Motor to Pin 1 (1) should be wide as possible. The same rule should be used while designing wires for the ground Pin 3 (3) of the IC.



- 1) Motor VCC
- 2) (C) in schematic
- 3) GND
- 4) IC IRF7389 top view
- 5) Heat sink / (D) in schematic
- 6) Interlayer connections

## 3 Software Example

This example will describe how to realize two driven motors controlled by using the PPG macros in a Fujitsu 16FX MCU. The motors can be controlled by speed and direction.

### 3.1 Initialisations

The PPGs will be configured to be driven by a 48MHz clock. The frequency in this example is about 730Hz and is calculated as followed:

$$f_{\text{PWM}} = (f_{\text{CKSEL}} / \text{divider}) / (\text{Cycle\_Value\_Period})$$

with  $f_{\text{CKSEL}} = 48\text{MHz}$ ,  $\text{divider} = 1$ ,  $\text{Cycle\_Value\_Period} = 0\text{xFFFF}$

First all required PPGs has to be initialized. Following Registers are used:

- PCSR<n>: used to set the cycle value period (value: 0xFFFF)
- PDUT<n>: used to set the initial duty cycle (value: 0x0000)
- PCNL<n>: PPG Control Status register (Lower Byte) (value: 0x00)
- PCNH<n>: PPG Control Status register (Higher Byte) (value: 0xD0):

CNTE	STGR	MDSE	RTRG	CKS1	CKS0	PGMS	-
1	1	0	1	0	0	0	0
Enable	S-Trigger		Retrigger	divider			

```

void InitPPG0(void)
{
    PCSR0 = 0xFFFF; // always set cycle value PERIOD 1st
    PDUT0 = 0x0000; // setduty value DUTY CYCLE
    PCNL0 = 0x00; // Output disable
    PCNH0 = 0xD0; // Count enable, S-Trigger, Retrigger, CKSEL divided by 1
}
void InitPPG1(void)
{
    PCSR1 = 0xFFFF; // always set cycle value PERIOD 1st
    PDUT1 = 0x0000; // setduty value DUTY CYCLE
    PCNL1 = 0x00; // Output disable
    PCNH1 = 0xD0; // Count enable, S-Trigger, Retrigger, CKSEL divided by 1
}
void InitPPG2(void)
{
    PCSR2 = 0xFFFF; // always set cycle value PERIOD 1st
    PDUT2 = 0x0000; // setduty value DUTY CYCLE
    PCNL2 = 0x00; // Output disable
    PCNH2 = 0xD0; // Count enable, S-Trigger, Retrigger, CKSEL divided by 1
}
void InitPPG3(void)
{
    PCSR3 = 0xFFFF; // always set cycle value PERIOD 1st
    PDUT3 = 0x0000; // setduty value DUTY CYCLE
    PCNL3 = 0x00; // Output disable
    PCNH3 = 0xD0; // Count enable, S-Trigger, Retrigger, CKSEL divided by 1
}

void InitMotorDrivers(void)
{
    InitPPG0(); //Init PPG0
    InitPPG1(); //Init PPG1
    InitPPG2(); //Init PPG2
    InitPPG3(); //Init PPG3
}

```

## 3.2 Motor Functions

```

/* setSpeed is used to set duty of the PWM.          */
/* - PPGno <1..n>: 1 means PPG0                    */
/* - level: 0..65535                               */
void setSpeed(unsigned char PPGno, unsigned int level)
{
    unsigned char onoff = 0;
    if (level) {                                     //check if level is 0... if it is 0, disable output
        onoff = 1;
    }
    switch(PPGno) {
        case 1:
            PDUT0 = level;                           //PPG0 was chosen, set duty cycle
            PCNL0_OE = onoff;                         //switch PPG on/off (if level=0 switch it off)
            break;
        case 2:
            PDUT1 = level;                           //PPG1 was chosen, set duty cycle
            PCNL1_OE = onoff;                         //switch PPG on/off (if level=0 switch it off)
            break;
        case 3:
            PDUT2 = level;                           //PPG2 was chosen, set duty cycle
            PCNL2_OE = onoff;                         //switch PPG on/off (if level=0 switch it off)
            break;
        case 4:
            PDUT3 = level;                           //PPG3 was chosen, set duty cycle
            PCNL3_OE = onoff;                         //switch PPG on/off (if level=0 switch it off)
            break;
    }
}

/* setMotor is used to set speed and direction of one of <n> dc motors. */
/* - motorno <1..n>                                                    */
/* - speed: -32768...+32767 (+/- for direction)                       */
void setMotor (unsigned char motorno, signed short speed) {
    if (speed >= 0) {                                                 //direction depending
        setSpeed((motorno)*2, (((unsigned int) speed)<<1));          //motor<n> = {PPG<n*2-1>,PPG<n*2>}
        setSpeed ((motorno)*2-1, (unsigned int)0);
    } else {
        setSpeed ((motorno)*2, (unsigned int)0);                   //motor<n> = {PPG<n*2-1>,PPG<n*2>}
        setSpeed ((motorno)*2-1, (((unsigned int) (-speed))<<1));
    }
}

```

## 3.3 Sample Program

The following sample program starts the motors in full speed while decreasing the speed of the motors until total stop, changes the direction and increases the speed to full speed and repeats this procedure.

```

void main(void)
{
    signed short int speed = 0;
    InitMotorDrivers();
    for(;;)
    {
        for(speed=-32768;speed < 32768;speed++)
        {
            setMotor(1, speed);
            setMotor(2, speed);
        }
        for(speed=-32768;speed < 32768;speed++)
        {
            setMotor(1, -speed);
            setMotor(2, -speed);
        }
    }
}

```

## 4 Appendix

### 4.1 International Rectifier

Homepage: <http://www.irf.com/indexsw.html>

IRF7389: <http://www.irf.com/product-info/datasheets/data/irf7389.pdf>

### 4.2 Fujitsu Microelectronics

Fujitsu Microelectronics Europe GmbH <http://www.fujitsu.com/emea/>

16 FX Series: [http://mcu.emea.fujitsu.com/mcu\\_product/overview\\_16bit.htm](http://mcu.emea.fujitsu.com/mcu_product/overview_16bit.htm)

Additional Application Notes can be found in:  
[http://mcu.emea.fujitsu.com/mcu\\_product/mcu\\_all\\_apnotes.htm](http://mcu.emea.fujitsu.com/mcu_product/mcu_all_apnotes.htm)

Application Note for 16FX Series: "How to use PPG": (application note 300201)

Generation of additional software PWM channel at IO ports using a reload timer and DMA transfer: (application note 300239)

### 4.3 Additional Websites:

Sample Application: Manuel Schreiner – MCUroboBoard:  
<http://www.io-experte.de/fujitsu/mcuroboboard>