AN3860

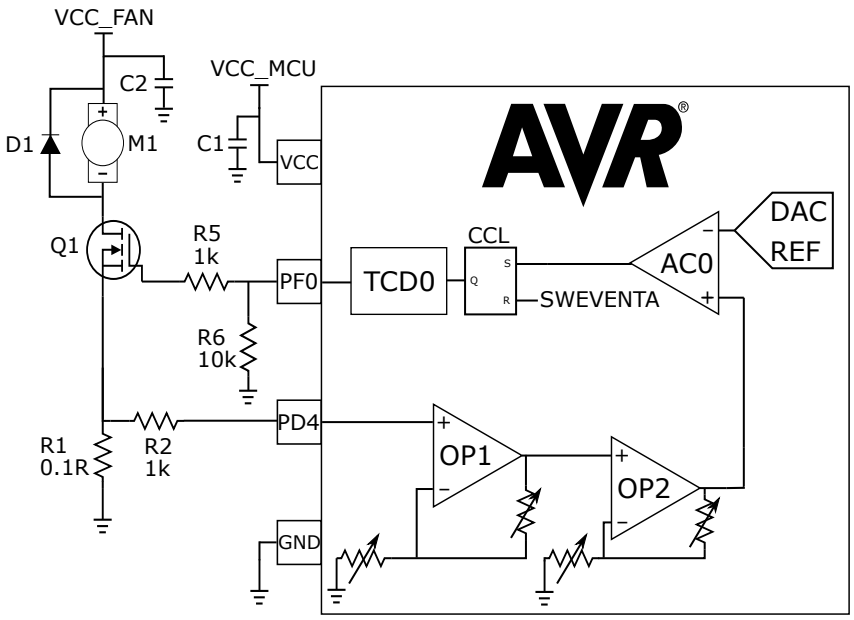
Overcurrent Protection Using the Integrated Op Amps on AVR® DB

Introduction

Author: Johan Vaarlid and Martin Mostad, Microchip Technology Inc.

The Timer/Counter type D (TCD) on the AVR® DB features an asynchronous event input that can be used to shut down the output under fault conditions. This application note will demonstrate and explain how to detect overcurrent, by using the configurable internal OPAMP (Analog Signal Conditioning) and Analog Comparator (AC) to detect a fan motor failure and send this detection as a signal to the asynchronous event input of the TCD, shutting down the load. Figure 1 outlines the working principle:

Figure 1. Overcurrent Protection Circuit Schematic



In normal operation, the TCD output pin is controlling the speed of a fan via a transistor gate. The Analog Comparator (AC) monitors the fan current through a current-sense amplifier implemented using the integrated operational amplifiers on the microcontroller.

R1 is the current shunt resistor, which generates a voltage from the current that passes through the motor. R2 limits the current into the microcontroller in the event of a high-voltage spike on R1.

The AC is configured to send an event as soon as the fan current rises above a selected threshold. A possible cause for such a current spike is a fan stopped by a mechanical obstacle. The event system of the device is configured to route the event from the AC to an RS-latch in the Configurable Custom Logic (CCL) which stops the TCD output.

On fan failure, the rising current triggers the AC to send an event to the S-port of the RS-latch making the output Q high. This signal is then sent to the TCD failure event. Thanks to its asynchronous event input, the TCD will abort driving the fan and shut off the supply current.

After the fan supply was interrupted by the TCD, application-specific actions can be implemented, depending on the application. Some possible actions include attempted restart, operation abort, or failure signaling. In this example, we have made a simple restart algorithm initiated by clicking the button on the AVR128DB48 Curiosity Nano.

This application is one example of an implementation of functional safety. One way a current spike can occur in a fan motor is when the motor experiences strain, for example from a finger or object stuck in or pressing on the fan blades. In such an event, an automatic shutdown can be used to protect the end user from injury and to reduce the chance of damage.

The example code for replicating the results described in this application note is available on GitHub.

Additional details on AC, TCD, CCL, OPAMP, device performance and general configuration are available in the device data sheet.

Features

- Auto-calibration of the AC to a wide range of normal load current draws
- Integrated op amp with internal resistor ladders for adjustable gain makes the solution even more adaptable
- Low BOM and PCB area implementation
- MPLAB® Mindi™ model and schematics provided for simplified development and adaptation

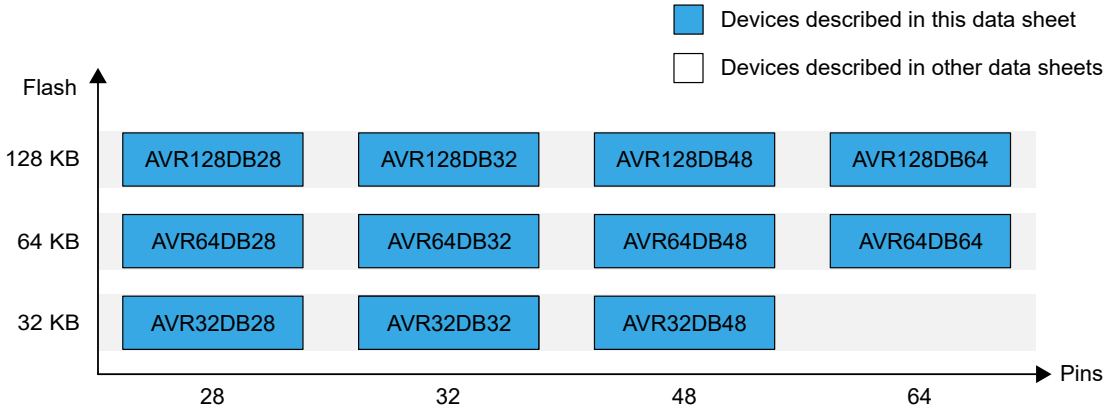
Table of Contents

1. Relevant Devices

This section lists the relevant devices for this document. The following figures show the different family devices, laying out pin count variants and memory sizes:

- Vertical migration upwards is possible without code modification, as these devices are pin-compatible and provide the same or more features
- Horizontal migration to the left reduces the pin count and, therefore, the available features
- Devices with different Flash memory sizes typically also have different SRAM and EEPROM

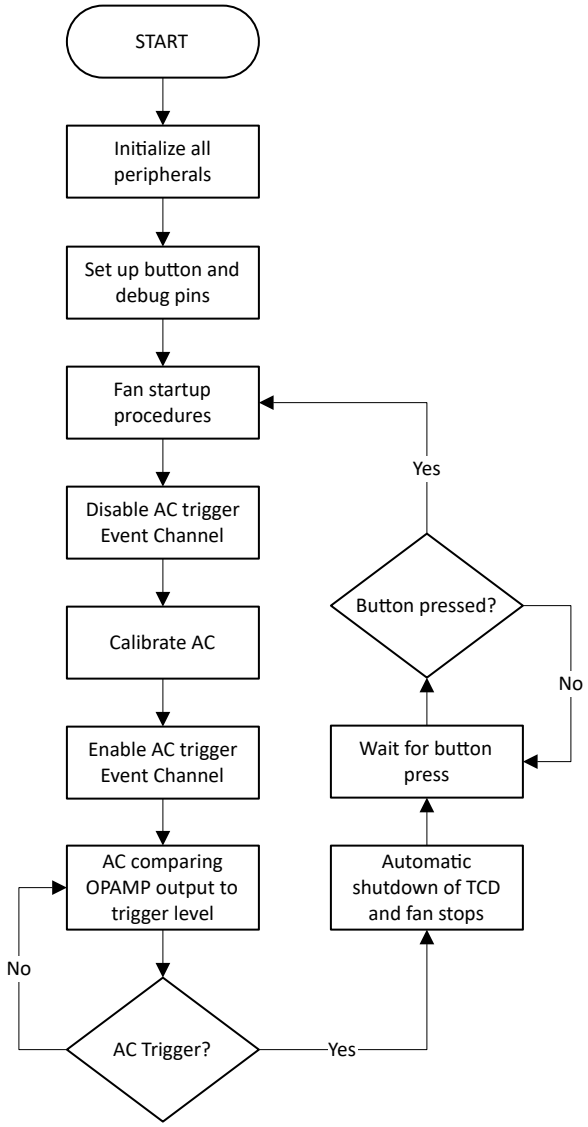
Figure 1-1. AVR® DB Family Overview



2. Overview

The application creates a current spike detector that continuously monitors the motor current independently of the CPU. On start-up, the device measures the maximum voltage across the current shunt resistor. To prevent false alarms, and to provide some error margins, this current limit is increased by 50 mV. If a spike is detected, the TCD will automatically shut down the motor, and the code will wait for a button press to turn the motor on again.

Figure 2-1. Code Flow Diagram



3. Hardware Configuration

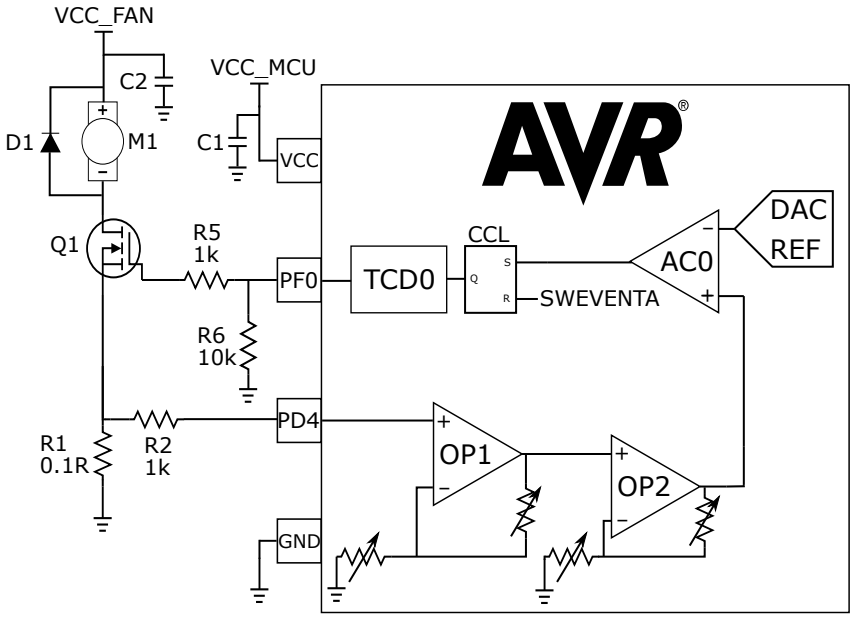
This application note assumes the use of the AVR128DB48 Curiosity Nano (EV35L43A). To set up this circuit, a breadboard or a stripboard PCB can be used with the AVR128DB48 Curiosity Nano and the following components are needed:

- AVR128DB48 Curiosity Nano Evaluation Kit
- A small 5V computer fan or similar (M1)
- 2x 1 kΩ resistor (R2, R5)
- 1x 10 kΩ resistor (R6)
- 1x 0.1 Ω current sense resistor (R1)
- 2x 100 nF capacitors (C1, C2)
- 1x N-Channel MOSFET transistor (Q1)
- 1x signal diode (D1)

Note: C1 is already present on the AVR128DB48 Curiosity Nano and as such does not need to be added when using the kit.

These components have to be connected as shown in the schematic below:

Figure 3-1. Overcurrent Protection Circuit Schematics

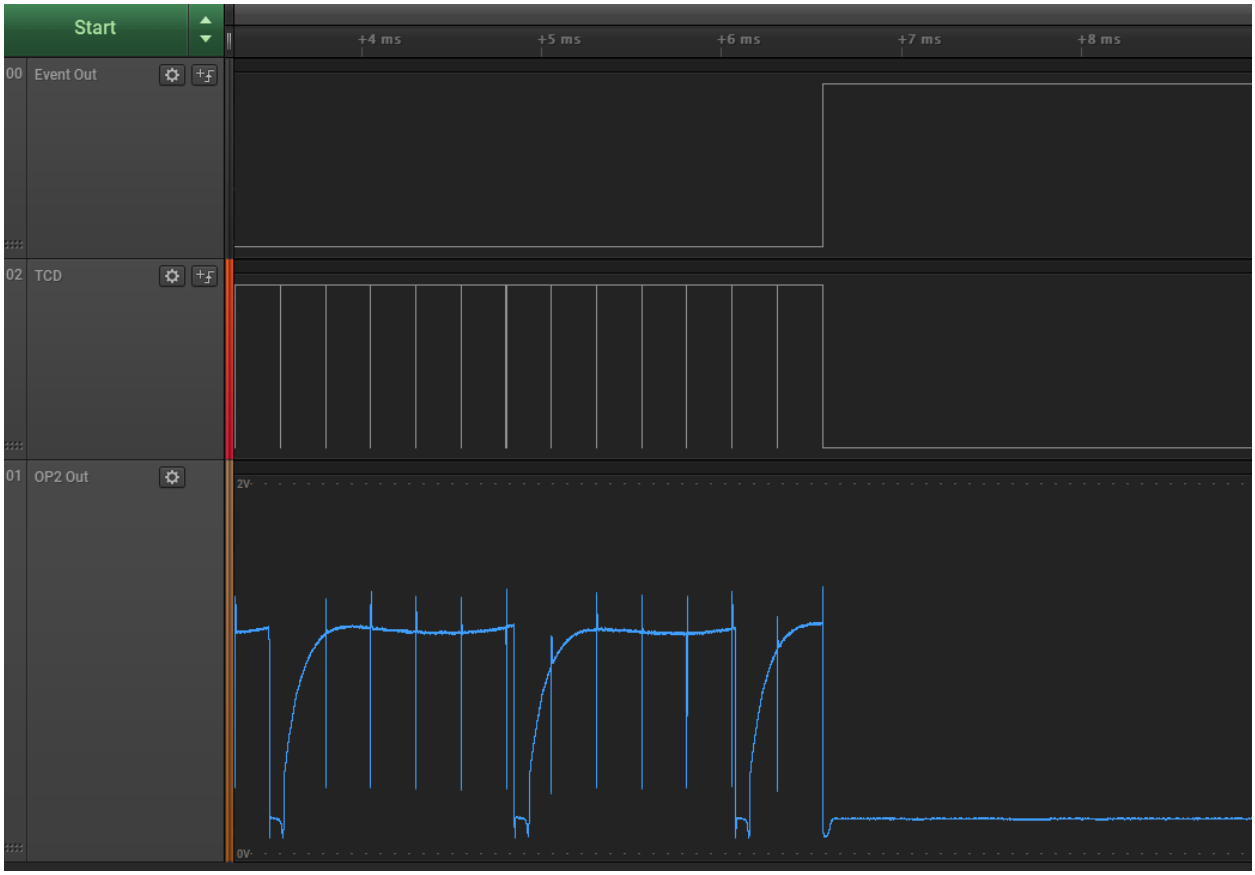


4. Current Spike Detection Code

4.1 Working Code Result

When the hardware is set up as described in the previous section, the resulting TCD output, and motor current draw should look like in Figure 4-1. Different motors might have a different current draw profile. The spikes seen in “OP2 Out” is due to noise from TCD toggling. In this example, TCD is run at the maximum duty cycle, which is one cycle low and the rest high.

Figure 4-1. Motor Failure Event, Sampled at 50 MHz



4.2 Initialization

First, initialize the peripherals to be used: TCD, CCL, AC and OPAMP.

Start with setting up the OPAMP as two cascaded noninverting Programmable Gain Amplifiers (PGA), with a total gain of 60. This gain is chosen based on the input signal produced by the current draw of the fan motor used during development. Depending on the current draw, a higher or lower gain can be chosen to fit your application. To lower noise, the first op amp in the cascade is set to maximum, that is 16x, while the second op amp can be adjusted to application needs, preferably to a gain that results in a 1-2V range output for ease of detection. If a different setup is used, please refer to the AVR128DB48 Data Sheet for gain calculation. The following table shows possible gain settings for OPAMP.OP2RESMUX.

Table 4-1. Gain Settings for OPAMP.OP2RESMUX

Group Configuration	Resulting Total Gain
OPAMP_OP2RESMUX_MUXWIP_WIP0_gc	~17
OPAMP_OP2RESMUX_MUXWIP_WIP1_gc	~18
OPAMP_OP2RESMUX_MUXWIP_WIP2_gc	~21
OPAMP_OP2RESMUX_MUXWIP_WIP3_gc	~32
OPAMP_OP2RESMUX_MUXWIP_WIP4_gc	~43
OPAMP_OP2RESMUX_MUXWIP_WIP5_gc	~64
OPAMP_OP2RESMUX_MUXWIP_WIP6_gc	~128
OPAMP_OP2RESMUX_MUXWIP_WIP7_gc	~256

```
void opamp_init(void)
{
    /*Disable input on op amp output pin*/
    PORTD.PIN5CTRL = PORT_ISC_INPUT_DISABLE_gc;
       PORTE.PIN1CTRL = PORT_ISC_INPUT_DISABLE_gc;

    /*Set up op amp*/
    OPAMP.CTRLA = OPAMP_ENABLE_bm;
    OPAMP.TIMEBASE = (uint8_t) ceil(CLK_PER*0.000001)-1; /*Number of peripheral clock cycles
that amounts to 1us*/

    //OP2 setup
    OPAMP.OP2CTRLA = OPAMP_RUNSTBY_bm | OPAMP_ALWAYSON_bm | OPAMP_OP2CTRLA_OUTMODE_NORMAL_gc;
    OPAMP.OP2SETTLE = OPAMP_MAX_SETTLE_TIME; //As the settle time is unknown, the maximum
should be set
    OPAMP.OP2INMUX = OPAMP_OP2INMUX_MUXNEG_WIP_gc | OPAMP_OP2INMUX_MUXPOS_LINKOUT_gc;
    OPAMP.OP2RESMUX = OPAMP_OP2RESMUX_MUXWIP_WIP5_gc | OPAMP_OP2RESMUX_MUXBOT_GND_gc|
OPAMP_OP2RESMUX_MUXTOP_OUT_gc;

    //OP1 setup
    OPAMP.OP1CTRLA = OPAMP_RUNSTBY_bm | OPAMP_ALWAYSON_bm | OPAMP_OP1CTRLA_OUTMODE_NORMAL_gc;
    OPAMP.OP1SETTLE = OPAMP_MAX_SETTLE_TIME; //As the settle time is unknown, the maximums
should be set
    OPAMP.OP1INMUX = OPAMP_OP1INMUX_MUXNEG_WIP_gc | OPAMP_OP1INMUX_MUXPOS_INP_gc;
    OPAMP.OP1RESMUX = OPAMP_OP1RESMUX_MUXWIP_WIP7_gc | OPAMP_OP1RESMUX_MUXBOT_INN_gc |
OPAMP_OP1RESMUX_MUXTOP_OUT_gc;
}
```

Then, set up the TCD to drive the PF0 pin, and to shut down on a failure event. Currently, the TCD is not enabled. However, it will be started later on.

```
void tcd_init(void)
{
    PORTMUX.TCDROUTEA = PORTMUX_TCD0_ALT2_gc;
    PORTF.DIRSET=PIN0_bm;

    TCD0.CTRLB = TCD_WGMODE_TWORAMP_gc;
    TCD0.CMPASET = 0; /* Compare A Set: 0 */
    TCD0.CMPACLR = 1001; /* Compare A Clear: 1001 */
    TCD0.CMPBSET = 0; /* Compare B Set: 0 */
    TCD0.CMPBCLR = 1000; /* Compare B Clear: 1000 */

    ccp_write_io((void*)&(TCD0.FAULTCTRL),1 << TCD_CMPAEN_bp /* Compare A enable: enabled */
          | 0 << TCD_CMPA_bp /* Compare A value: disabled */
          | 0 << TCD_CMPB_bp /* Compare B value: disabled */
          | 0 << TCD_CMPBEN_bp /* Compare B enable: disabled */
          | 0 << TCD_CMPC_bp /* Compare C value: disabled */
          | 0 << TCD_CMPCEN_bp /* Compare C enable: disabled */
          | 0 << TCD_CMPD_bp /* Compare D vaule: disabled */
          | 0 << TCD_CMPDEN_bp /* Compare D enable: disabled */);

    TCD0.EVCTRLA = TCD_CFG_ASYNC_gc     /* Neither Filter nor Asynchronous Event is enabled */
```

```
    | TCD_ACTION_FAULT_gc  /* Event trigger a fault */
    | TCD_EDGE_RISE_HIGH_gc /* The falling edge or low level of event generates retrigger or
fault action */
    | 1 << TCD_TRIGEI_bp;  /* Trigger event enable: enabled */

    TCD0.INPUTCTRLA=TCD_INPUTMODE_WAITSW_gc; //Wait for a reset command
    TCD0.INPUTCTRLB=TCD_INPUTMODE_WAITSW_gc; //Wait for a reset command

    while ((TCD0.STATUS & TCD_ENRDY_bm) == 0); // Wait for Enable Ready to be high.

    TCD0.CTRLA = 0 << TCD_ENABLE_bp      /* Enable: disabled */
    | TCD_CLKSEL_OSCHF_gc   /*  */
    | TCD_CNTPRES_DIV1_gc   /* Sync clock divided by 1 */
    | TCD_SYNCPRES_DIV1_gc;


}
```

The AC needs to be set up to trigger at a suitable level. First, set it to trigger at some low level, here 1.1V. This level will later be automatically calibrated to suit the motor at hand.

```
void ac_init(void)
{
    PORTA.DIRSET=PIN7_bm;
    VREF.ACREF = VREF_REFSEL_4V096_gc;

    AC0.MUXCTRL = 0 << AC_INVERT_bp     /* Invert AC Output: disabled */
    | AC_MUXNEG_DACREF_gc /* DAC Reference */
    | AC_MUXPOS_AINP2_gc; /* Positive Pin 0 */


    AC0.DACREF = ac_calculate_trigger_voltage(AC_TRIGGER_VOLTAGE_MV_INIT); /* DAC Voltage
Reference: 0x64 */

    AC0.CTRLA = 1 << AC_ENABLE_bp /* Enable: enabled */
    | AC_HYSMODE_NONE_gc   /* No hysteresis */
    | AC_POWER_PROFILE0_gc /* Power profile 0, lowest consumption and highest response time.
*/
    | 1 << AC_OUTEN_bp     /* Output Buffer Enable: enabled */
    | 0 << AC_RUNSTDBY_bp; /* Run in Standby Mode: disabled */
}
```

The helper function `ac_calculate_trigger_voltage(uint16_t mV)` is used to set the AC trigger. It calculates the DACREF value corresponding to a trigger level in millivolt.

```
uint8_t ac_calculate_trigger_voltage(uint16_t mV)
{
    uint8_t triggerVoltage = (((uint32_t)mV*256)/VREF_AC_MV);
    return triggerVoltage;
}
```

To keep the overcurrent signal active until reset, the CCL LUT0 and LUT1 form an RS-latch with the AC0 output as S input and Software Event A as R input as shown in Figure 3-1.

```
void ccl_init(void)
{
    CCL.SEQCTRL0 = CCL_SEQSEL_RS_gc; // Create a RS latch
    /*Set up LUT0*/
    CCL.LUT0CTRLB = CCL_INSEL0_AC0_gc; // AC0 and mask all other bits
    CCL.TRUTH0 = 0x02; // When AC is hight hight output, otherwise low output
    CCL.LUT0CTRLA = CCL_ENABLE_bm | CCL_OUTEN_bm;
    PORTA.DIRSET = PIN3_bm;

    /*Set up LUT1*/
    EVSYS.USERCCLLUT1A = EVSYS_USER_CHANNEL1_gc;
    CCL.LUT1CTRLB = CCL_INSEL0_EVENTA_gc; // Event A as input and mask all other bits
    CCL.TRUTH1 = 0x02; // When Event A is high, we have a high output, otherwise low output
    CCL.LUT1CTRLA = CCL_ENABLE_bm;

    CCL.CTRLA = CCL_ENABLE_bm;
}
```

Lastly, the button and debug signals showing the AC event level are set up.

```
void button_init(void)
{
    PORTB.DIRCLR=PIN2_bm;
    PORTB.PIN2CTRL=PORT_ISC_FALLING_gc | PORT_PULLUPEN_bm;
    sei();
}
```

```
    button_init();
    PORTC.DIRSET = PIN4_bm | PIN5_bm;
    PORTC.OUTCLR = PIN4_bm | PIN5_bm;
    PORTD.DIRSET = PIN2_bm;
```

4.3 Motor Start-Up and AC Calibration

To start the fan motor, utilize the function `fan_start()` given below. This function invokes a series of other functions which will be described chronologically.

```
void fan_start(void)
{
    ac_trig_event_disable();
    tcd_enable();
    ac_calibration();
    ac_trig_event_enable();

}
```

`ac_trig_event_disable()` disconnects the AC trigger event from the CCL RS-latch, disabling the current spike detector.

```
void ac_trig_event_disable(void)
{
    AC0.INTCTRL &= ~AC_CMP_bm;
    EVSYS.CHANNEL0=EVSYS_CHANNEL0_OFF_gc;
    EVSYS.USERTCD0INPUTA = EVSYS_USER_OFF_gc;
    EVSYS.USEREVSYSEVOUTD = EVSYS_USER_OFF_gc;
}
```

`tcd_enable()` then turns on the fan motor by enabling the TCD peripheral.

```
void tcd_enable(void)
{
    TCD0.CTRLA = TCD_ENABLE_bm;
    TCD0.CTRLE = TCD_RESTART_bm;
}
```

`ac_calibration()` increases the triggering point of the AC in 50 mV increments until no trigger happens within 100 ms. This is done on every start-up to ensure optimal detection conditions and to avoid false positives. The current draw of some motors is affected by temperature and other physical conditions.

```
void ac_calibration(void)
{
    PORTB.DIRSET = PIN3_bm;
    PORTB.OUTCLR = PIN3_bm;
    uint8_t calibrating = 1;
    uint16_t ac_trigger_voltage_mv=AC_TRIGGER_VOLTAGE_MV_INIT;
    AC0.INTCTRL = AC_INTMODE_NORMAL_POSEDGE_gc;
    while (calibrating)
    {
        ac_trigger_voltage_mv += 50;
        AC0.DACREF = ac_calculate_trigger_voltage(ac_trigger_voltage_mv);
        _delay_ms(100);//Allow some time for the flag to be raised
        if (!(AC0.STATUS & AC_CMPIF_bm))
        {
            calibrating=0;
            PORTB.OUTSET =PIN3_bm;
```

```
        }
        AC0.STATUS=AC_CMPIF_bm;
    }

}
```

`ac_trig_event_enable()` reconnects the AC trigger event to the CCL, making the system ready to detect a current spike event.

```
void ac_trig_event_enable(void)
{
    AC0.INTCTRL = AC_INTMODE_NORMAL_POSEDGE_gc | AC_CMP_bm;
    EVSYS.SWEVENTA = EVSYS_SWEVENTA_CH1_gc;
    EVSYS.CHANNEL0=EVSYS_CHANNEL0_CCL_LUT0_gc; //Output from the RS latch as trigger
    EVSYS.USERTCD0INPUTA=EVSYS_USER_CHANNEL0_gc;
    EVSYS.USEREVSYSEVOUTD = EVSYS_USER_CHANNEL0_gc;
}
```

4.4 Detection and Restart

On a detected current spike, the hardware stops the motor without any CPU involvement. Pressing button SW0 on the AVR128DB48 Curiosity Nano will trigger the Interrupt Service Routine (ISR) `ISR(PORTB_PORT_vect)`, which will invoke the `fan_start()` routine to restart the fan motor.

```
ISR(PORTB_PORT_vect)
{
    fan_start();
    PORTB.INTFLAGS=0xff;
}
```

5. MPLAB® Mindi™

To help the development and adaptation of this application to different environments and fans, an MPLAB Mindi simulation of the application was created. The simulation consists of two parts. The first part is a model of a DC motor representing the fan. The second part is all the components needed to replicate the functionality of the AVR DB. Only the op amp components are true representations of the circuits in the AVR DB. The rest of the components are generic circuit elements needed to mimic the behavior of the AVR DB.

5.1 Modeling Using MPLAB® Mindi™

The classical way of modeling a DC motor is as two differential equations, one for the mechanical and one for the electrical dynamics as shown in below:

$$J\ddot{\theta} + b\dot{\theta} = Ki$$

$$L\dot{i} + Ri = V - K\dot{\theta}$$

MPLAB Mindi is only capable of simulating circuit elements, therefore the mechanical equation is transformed into an equivalent circuit representation. This is achieved by saying that $R_{los} = b/K^2$, $L_I = J/K^2$ and $i_{emf} = \dot{\theta}K$. This gives the differential equation:

$$L_I\dot{i_{emf}} + R_{los}i_{emf} = i$$

$$L\dot{i} + Ri = V - i_{emf}$$

The model of the DC motor consists of three parts; Motor_Electrical, Motor_Mechanical and Motor_Load.

5.1.1 Motor Electrical

The Motor Electrical part implements the electrical differential equation explained above. It consists of an arbitrary voltage supply acting as the power supply for the DC motor being controlled by a PWM signal. There is also an arbitrary voltage supply used to transform the current from the mechanical part into voltage, representing the EMF of the motor.

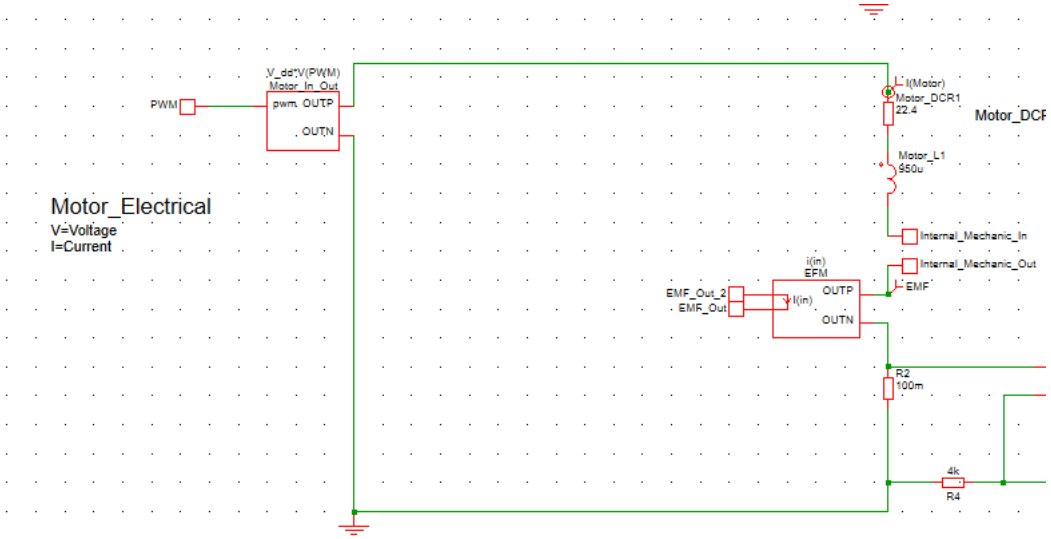
In Motor Electrical two parameters determine the behavior of the motor, R (denoted Motor_DCR1) and L (denoted Motor_L1). They can be calculated using the following formula:

$$R = \frac{V_{in}}{i_{stall}}$$

$$L = R\tau_e$$

Where V_{in} is the input voltage to the DC motor, i_{stall} is the stall current and τ_e is the time constant of the electrical system.

Figure 5-1. Motor Electrical



5.1.2 Motor Mechanical

The Motor Mechanical part implements the transformed mechanical differential equation explained above. It consists of an arbitrary voltage supply to transform the current from the electrical part to voltage and an arbitrary voltage supply to represented the losses due to the load.

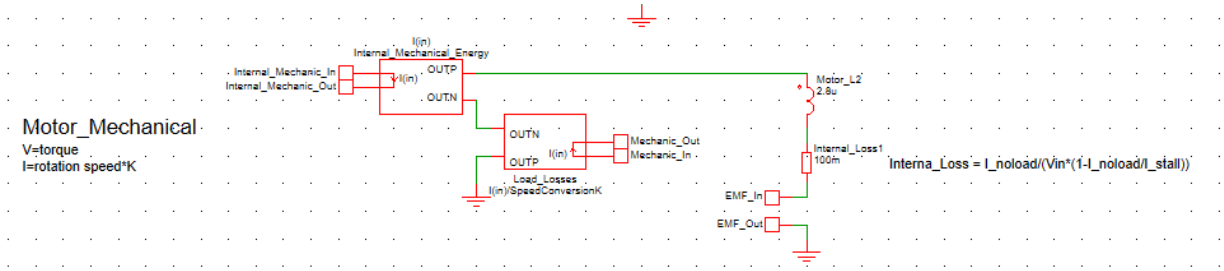
In Motor Mechanical, two parameters determine the behavior of the motor, R_{loss} (Interla_Loss1) and L_i (Motor_L2). They can be calculated using the following formula:

$$R_{loss} = \frac{i_{no\_load}}{V_{in}\left(1 - \frac{i_{no\_load}}{i_{stall}}\right)}$$

$$L = R_{loss}\tau_m$$

Where i_{no_load} is the current draw of the motor with no load, i_{stall} is the stall current and τ_m is the time constant of the mechanical system.

Figure 5-2. Motor Mechanical

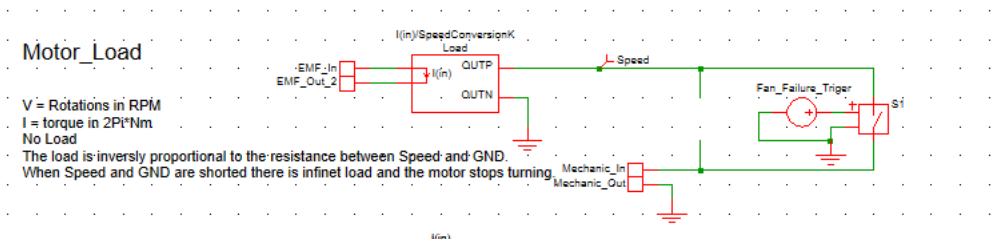


5.1.3 Motor Load

The Motor Load part represents the losses due to the load, it consists of an arbitrary voltage source that transforms the current from the mechanical part to speed in RPM. The losses due to the load are inversely proportional to the resistance between the voltage source and ground. The fan used in this example is permanently connected to the motor, so the losses due to the fan are already accounted for in the mechanical part. In normal operation, there is infinite resistance between the voltage source and ground (open circuit), representing no load.

To simulate a fan failure, such as something getting stuck in the fan, the model also has a switch that will short the voltage source to ground. This switch is triggered by a pulse coming from the Fan_Failure_Trigger voltage source.

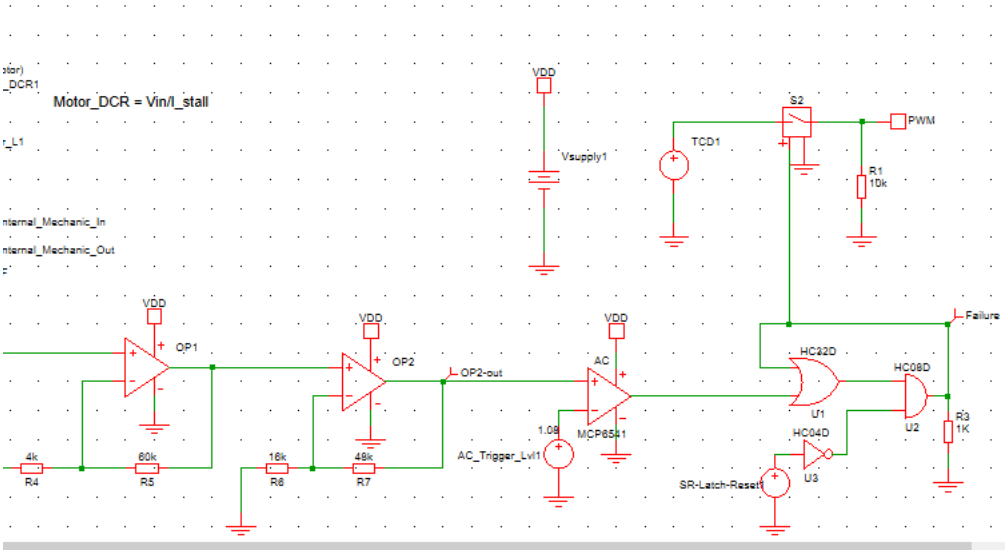
Figure 5-3. Motor Load



5.2 AVR® DB Components

To represent the AVR DB functionality, two of the AVR DB op amp models are configured as cascading non-inverting PGA as in software. The output from OP2 is connected to a generic Analog Comparator (AC) representing the AC in the AVR DB. The AC is also connected to a voltage source, AC_Trigger_lvl1, which sets the trigger level for the failure event. The output from the AC is connected to an RS-latch created by logic gates and inverters as in the actual hardware. The output from the RS-latch is connected to a switch which will break the connection between the PWM signal and the motor, which represents the TCD reacting to a fault event and turning off.

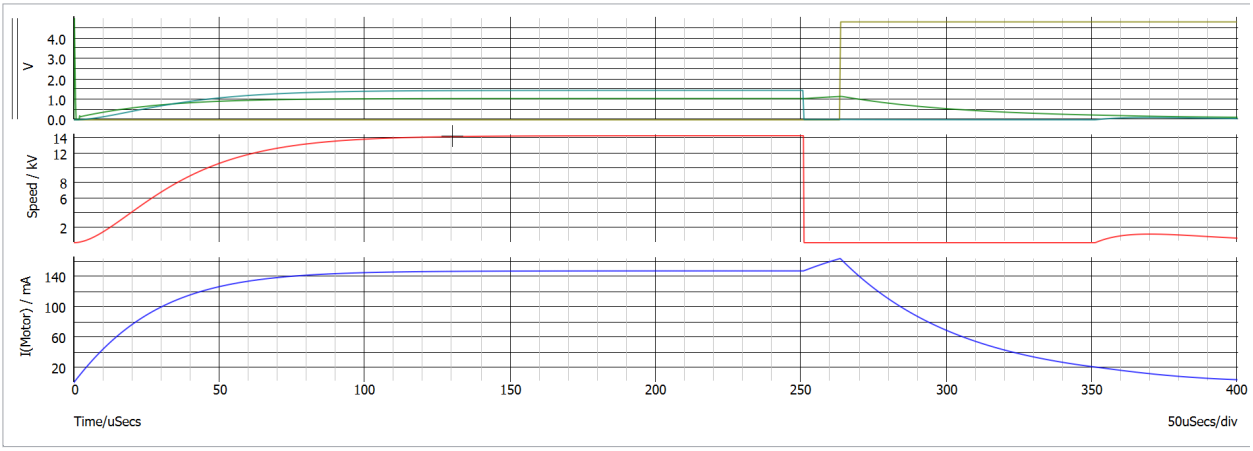
Figure 5-4. AVR® DB Components



5.3 Running the Simulation

Before running the simulation, change the parameters discussed in the above section to something that fits the intended DC motor. Simulate by selecting **Simulator** → **Run Schematic** or by pressing **Function Key F9**.

Figure 5-5. Simulation



The resulting output graph is shown above in Figure 5-5. The blue line is the current draw, red is the motor speed in RPM, green is the output from the op amp, turquoise is the EMF voltage and yellow is the failure detection. It can be seen that at about 250 µs something is obstructing the fan causing it to stall and the current to spike, the current spike triggers the fault event and the motor is turned off.

The code for Mindi models can be found on GitHub here:

View Code Example on GitHub
Click to browse repository

6. References

1. AVR128DB48 product page: www.microchip.com/wwwproducts/en/AVR128DB48.
2. AVR128DB48 Data Sheet: www.microchip.com/DS40002247.
3. Curiosity Nano AVR128DB48 product page: www.microchip.com/DevelopmentTools/ProductDetails/PartNO/EV35L43A.
4. AVR128DB48 Curiosity Nano User Guide: www.microchip.com/DS50003037.
5. Curiosity Nano AVR128DB48 Schematics ww1.microchip.com/downloads/en/DeviceDoc/AVR128DB48_Curiosity_Nano_Schematics.pdf.

7. Revision History

Doc. Rev.	Date	Comments
B	05/2021	New schematic and updated code
A	02/2021	Initial document release

The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods being used in attempts to breach the code protection features of the Microchip devices. We believe that these methods require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Attempts to breach these code protection features, most likely, cannot be accomplished without violating Microchip's intellectual property rights.
- Microchip is willing to work with any customer who is concerned about the integrity of its code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "unbreakable." Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Trademarks

ISBN: 978-1-5224-8313-7

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: www.microchip.com/support Web Address: www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **Dallas** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4485-5910 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-72400 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-72884388 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820