



SW-UCM MANUAL

Universal Control Module

V-1.01 November 14, 2009

www.extreme-fire.com

Introduction

The Extreme-Fire SW-UCM is a universal computer control module that can be easily programmed to carry out various functions.

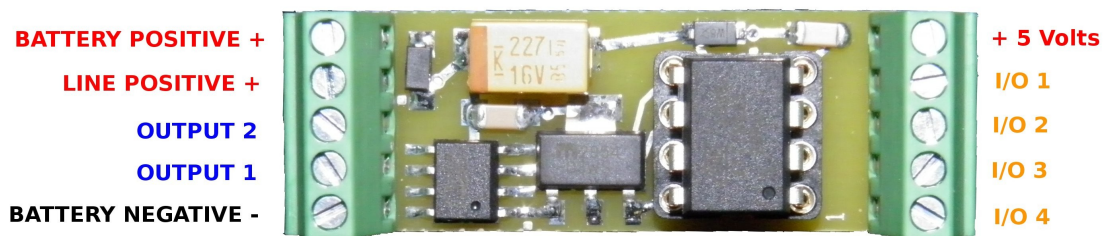
Features

- 4 Inputs (A/D or digital) and two high current Outputs.
- 8MHz ATMEL ATTINY85 RISC processor.
- Plenty of program and variable space left open.
- All major hardware functions are already programmed. A short C code can change the functions dramatically.
- 100uS clock for accurate timing.
- Hardened to easily withstand the AEG Environment.
- Screw clamp terminal connections eliminate many connectors and make installation far easier.
- Open source, Public Domain, design and software.

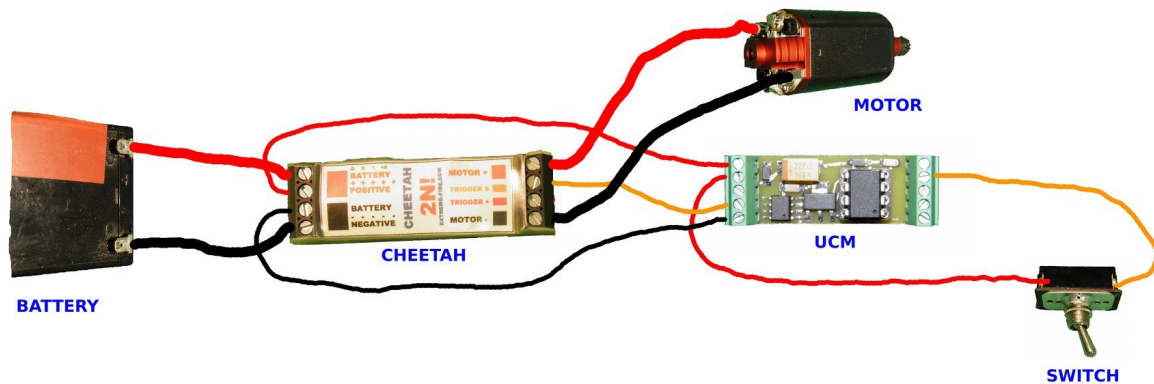
Installation

The SW-UCM normally has the following connections available:

- 3 A/D inputs with 20k ohms of resistance that read from 0 to 20.00 volts.
- 2 100mA buffered outputs from 0 to Battery voltage.
- 1 100mA +5.00 volt supply.
- 1 Filtered and fused Battery line voltage output.
- 1 Output that normally drives an LED.
- 6-18V Battery input.



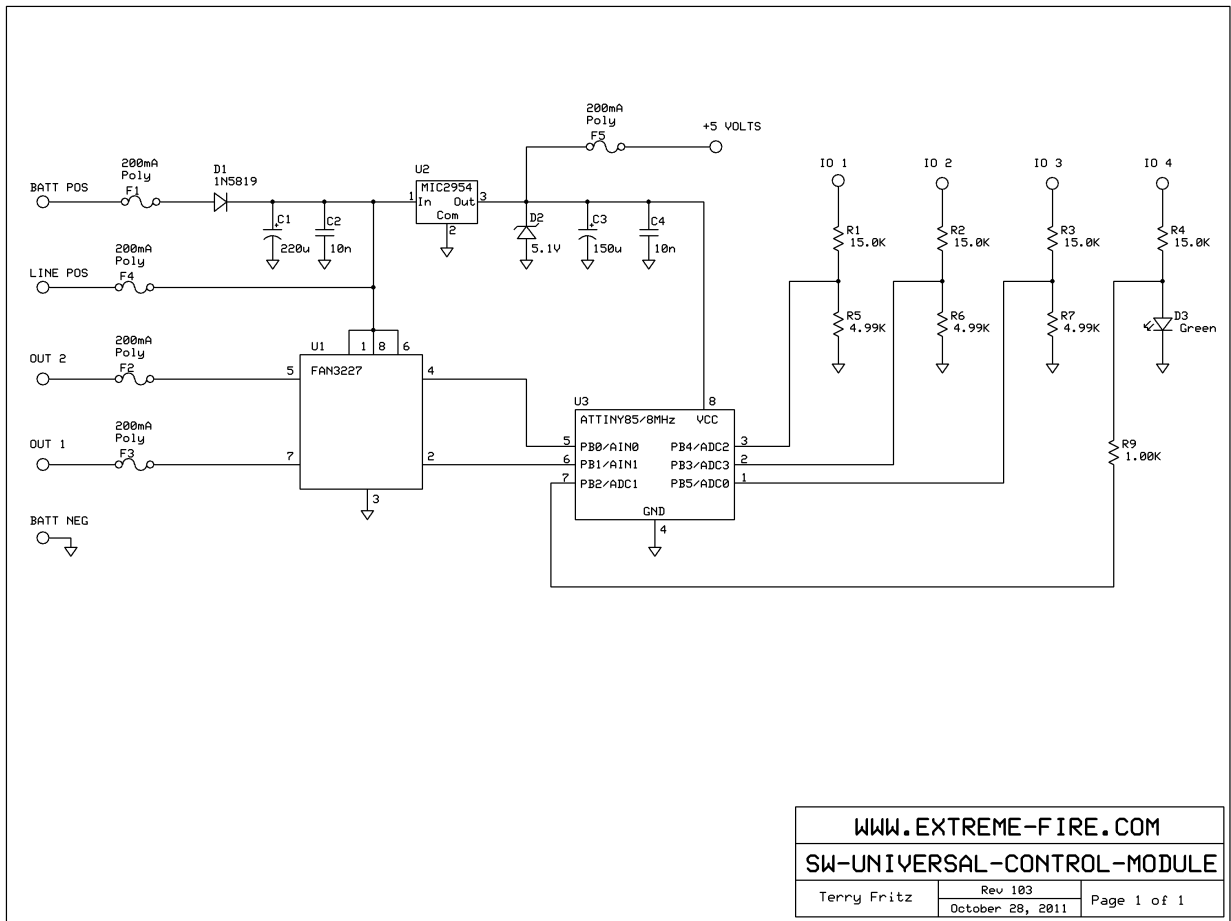
Connections



Typical wiring

Specifications:

Battery Voltage	6 to 18 VDC
Current	~15 mA base unit.
Size	1.6 x 0.58 x 0.46 inches (41 x 15 x 12 mm)
Weight	0.26 ounces (7.3 gram)
I/O Input Resistance	20 kOhm
Drive Current	100mA



Schematic Diagram

SW-UCM				
	220uF 16V Tan. Capacitor	DigiKey	399-5124-1-ND	1
	10nF 50V Ceramic Capacitor	DigiKey	490-1774-1-ND	2
	150uF 6V Tan. Capacitor	DigiKey	399-4732-1-ND	1
	1N5819 Diode	DigiKey	1N5819HW-FDICT-ND	1
	5.1V 500mA Zener Diode	DigiKey	BZT52C5V1-FDICT-ND	1
	Green 1206 LED	DigiKey	P11538CT-ND	1
	0.20A Poly Fuse 1206	DigiKey	NANOSMDC020FCT-ND	5
	4.99K Ohm 1/4W 1% Resistor	DigiKey	P4.99KFCT-ND	4
	15K0 1/4W 1% Resistor	DigiKey	P15.0KFCT-ND	4
	1.00K Ohm 1/4W 1% Resistor	DigiKey	P1.00KFCT-ND	1
	8-PIN IC Socket	DigiKey	ED90208-ND	1
	MOSFET Driver FAN3227T	DigiKey	FAN3227TMXCT-ND	1
	5V Regulator MIC2954-03WS	DigiKey	576-1154-1-ND	1
	ATTINY85 MPU	DigiKey	ATTINY85-20PU-ND	1
	5-Pos Terminal	DigiKey	277-1276-ND	2
	PC Board	Gandolf	Generic	1

Parts List

```

/* SW-Universal Control Module - Factory Control Program
WWW.EXTREME-FIRE.COM
Target controller = ATTINY85-20PU microcontroller from ATMEL.
Clock speed = 8 MHz internal.
Development platform = AVR Studio 4 Version 4.15.623 Service Pack 3 Build 623.
C compiler = WinAVR 20070525 Optimization = -Os
Programmer = AVR Dragon
Hardware Schematic = SW-UCM all versions.
Hardware Layout = SW-UCM all versions.

```

This program is fully released to the Public Domain by Terry Fritz 2007 / 2011.

History

```

001 - Initial release Oct. 26, 2011 Terry Fritz
002 - Removed Temperature and EEPROM. BOD to 1.8V. Faster turn on to set outputs. Fixed mode2 A/D
scaling.
003 - Removed extra A/D ranges. 250kHz ADC clock.
004 - Fixed Pseudo Clock and tested - Works good!! Clock is about +-2%.

```

Known issues:

To use pin 1 (AUX1) the Reset Disable Fuse has to be set.
(RSTDISBL = 1) They use pin 1 by default and will keep the pin high if not fixed.

Set BrownOut voltage to 1.8V in fuses.

Set Clock div/8 fuse off during programming

Future changes:

None

```
*/
```

```

/* Include libraries */
#include <avr/io.h> /* For all the odd ATTINY stuff */
#include <avr/interrupt.h> /* For PseudoClock */

/* Define ATTINY85 ports (PBx) to the I/O ports */
/* Note that these port numbers or NOT the physical pin numbers! */
#define I01 4 /* I01 to PB4 */
#define I02 3 /* I02 to PB3 */
#define I03 5 /* I03 to PB5 */
#define I04 2 /* I04 to PB2 */
#define OUT1 1 /* OUT1 to PB1 */
#define OUT2 0 /* OUT2 to PB0 */
#define Ground 10 /* Internal A/D ground used for calibration */

/* Function prototypes */
void EnergizePorts(void); /* Enables the FETs in a controlled way */
long GetVoltage(unsigned char Sensor); /* Gets the Voltage (in mV) from the chosen source */
void CalibrateAtoD(void); /* Calibrates A/D convertor */
void StartClock(void); /* Start Pseudo Clock */

/* Initialize variables */
/* long is a 4 byte integer */
long Voltage = 0; /* A/D convertor returned voltage in mV */
long VoltageOffset0 = 0; /* A/D 5.0V REF offset error */
volatile int long PseudoClock = 0; /* Pseudo Clock value. +-2% */

long Input1 = 0;
long Input2 = 0;
long Input3 = 0;

```

```
long Input4 = 0;
int Output1 = 0;
int Output2 = 0;
int LED = 0;

long LoopCount = 0;
long counter1 = 0;
long HighCount = 0;
long LowCount = 0;
volatile int long ClockStart = 0;
volatile int long LoopTime = 0;

int main(void)
{ /* ===== Start of main ===== */

/* MAIN POWER UP */

/* Energize drive circuits */
EnergizePorts(); /* Energize drive circuits */
StartClock(); /* Start Clock */
CalibrateAtoD(); /* Setup A/D parameters */

/* ##### MAIN SCANNING LOOP ##### */
MainLoop:

LoopCount++; // Note used here

ClockStart = PseudoClock; // Start time of cycle

HighCount = 0; LowCount = 0; // Clear counters

for(counter1 = 0; counter1 < 10; counter1++) // Test trigger voltage 10 times.
{
// Get port voltages ~160uS per test.
Input1 = GetVoltage(I01);
if (Input1 >= 5000) {HighCount++;} else {LowCount++;}
}

if (HighCount >= 3) {Output1 = 1;} else {Output1 = 0;} // Ouput on if trigger was on 3 or more tests
LED = Output1; // Turn on LED with output

// Just a delay so the cycle time is 2mS
LoopTime = PseudoClock - ClockStart;
do {LoopTime = PseudoClock - ClockStart;} while (LoopTime < 20);

// Actually set output port values
PORTB = LED * 4 + Output1 * 2 + Output2;

goto MainLoop;
/* ##### END MAIN SCANNING LOOP ##### */
} /* ===== End of main ===== */

/* FUNCTIONS */

void StartClock(void)
{
/* 100uS real time clock setup on 8 bit counter */
TCCR1 = 0b10000000; // clear timer1 when it matches the value in OCR1C
TIMSK = 0b01000000; // enable interrupt when OCR1A matches the timer value
sei(); // enable global interrupts
}
```

```

OCR1A = 100;           // set the match value for interrupt 8MHz clock / 800 = 100uS
OCR1C = 100;           // and the same match value to clear the timer
TCCR1 = 0b10000100;    // Divide by 8 prescaler (this starts the timer)
}

ISR(TIM1_COMPA_vect){PseudoClock++;} /* Real time clock update routine on 100uS interrupt */

void EnergizePorts(void)
{
  PORTB = 0x00; /* Set pullups off */
  DDRB = 0b00000111; /* Set port B pins 0, 1 and 2 for output */
  PORTB = 0x00; /* Set port B pin 0 to 0 and pin 1 to 0 */
}

long GetVoltage(unsigned char Sensor) // Takes about 160uS
{
  /* Select input Source */
  if (Sensor == I03) {ADMUX = 0x00;}
  if (Sensor == I04) {ADMUX = 0x01;}
  if (Sensor == I01) {ADMUX = 0x02;}
  if (Sensor == I02) {ADMUX = 0x03;}
  if (Sensor == Ground) {ADMUX = 0x00;} /* Internal ground for calibration */
  /* Start A/D conversion */
  ADCSRA = 0b11000101; /* Start A/D conversion (bit 6 is set) 250kHz ADC clock*/
  /* Test if done (bit 6 goes clear, but is set when A/D is complete) */
  do {} while (bit_is_set(ADCSRA,6)); /* This could be fixed now with the new clock! */
  /* Get voltage Vref = 5000mV Rdivider = 1/4 Full range is 20V */
  Voltage = ADCW; /* This word is the 10 bit A/D result */
  Voltage = Voltage - VoltageOffset0; /* A/D Offset error adjustment */
  if (Voltage < 0) {Voltage = 0;} /* If negative fix it */
  Voltage = ((Voltage * 20000) / 1023); /* Convert A/D full range (1023) to 19980mV or 19.98V */
  return Voltage;
}

void CalibrateAtoD(void)
/* 5.0V REF Range = 20V 1023 steps */
{
  VoltageOffset0 = 0; /* Clear old value */
  VoltageOffset0 = GetVoltage(Ground); /* Run the converter with grounded input */
  VoltageOffset0 = ADCW; /* Get the raw A/D converter output word */
}

```

Typical Program