ALL Power Labs

GCU Manual

Hardware version 3.01

Draft revision 0.6

G. Homsy

6 December, 2009

Table of Contents

1 Introduction

The Allpower Labs GCU is a standalone industrial process control kit, based on an Atmel ATmega 1280 microcontroller. To the core controller are attached numerous I/O peripherals, chosen specifically to be useful for Gasifier control and research.

The GCU is supplied with a basic standalone application preinstalled. This application, called the "Basic App" in this document, supports basic measurement and manual control, using the built-in keypad buttons and the built-in display; and data logging, via the USB serial port. The Basic App, and all its supporting libraries, are Open-Source, and are distributed under the GPL (GNU General Public License). The source code is available from Allpower's "gekgasifier" web site, and on SourceForge.net.

The microcontroller is also preprogrammed with the Arduino boot loader. This allows any user with basic microcontroller skills to write and modify their own gasifier control programs, using the Open-Source Arduino development kit, available for free download from the Arduino site.

The GCU is available as a board-level product only, with two different feature sets: The "basic" and the "full". The "basic" may be upgraded to the "full" by a user experienced in surface mount soldering and electronics debugging techniques.

1.1 Features in brief

- ATmega 1280 processor core
- Sixteen K-type thermocouple inputs
- Six differential or gauge pressure/vacuum inputs
- Eight PWM FET outputs
- Four auxiliary analog inputs
- Frequency counter input
- Three R/C hobby servo outputs
- Display and four button keypad
- USB serial host interface
- SD-card slot
- CANbus interface
- Auxiliary RS-232 interface
- User prototyping / expansion area

1.2 Getting started

- Plug the power supply in
- (optional) Plug a K-type thermocouple into the connector labeled "T0"
- (optional) Attach a piece of tubing to one of the pressure ports on the pressure sensor labeled "PS0"
- (optional) Attach a hobby servo to the connector labeled "SRV0". The *black* wire from the servo should be connected to PIN 1 on the connector (labeled with a dot)

- (optional) Attach a *small* (less than 2 amp) twelve-volt DC motor to pins 2 and 6 of the connector labeled "FET0 – FET3"
- Now, plug the GCU into the power supply
- The power LED should light, and the heartbeat LED should start flickering
- The menu of the "Basic App" should immediately appear on the display screen
- Practice navigating through the various display and control screens. You should be able to measure temperature and pressure, and control your hobby servo and your DC motor
- If it works, congratulations! If it doesn't, refer to the "troubleshooting" section

2 Features in detail

2.1 Power input

The power input jack accepts 7 to 30 volts DC, center positive, on a 5 x 2.5 mm barrel connector. The input has a reverse blocking diode to prevent damage in case of reverse polarity connections.

Input power is applied directly to the FET output circuitry if jumpers JP1 and/or JP2 are installed. *The reverse blocking diode will not protect the FET circuitry!*

In parallel, the input power is fused by a resettable PTC fuse, F8, and then fed to a buck-mode switching regulator, to efficiently derive a 5 volt supply for the R/C hobby servos, and for the rest of the onboard circuitry.

2.2 CPU

The MCU used is the Atmel ATmega1280, in the LQFP100 package.

2.2.1 Feature set

The ATmega1280 features 128kB of in-circuit programmable flash memory, 4kB of EEPROM, and 8kB of on-chip SRAM.

The clock speed used in the GCU is 16MHz.

The MCU features four 8-bit PWM channels, and twelve programmable resolution PWM channels. Of these, only the programmable resolution PWM channels are used in the GCU. They are in groups of three, on four separate hardware timer/counters. Three groups are used for the FET PWM outputs, and one group is used for the R/C hobby servo outputs.

There are sixteen 10-bit analog inputs. Sadly, four are consumed by the JTAG interface. Six are used for the pressure and temperature sensors, four are auxiliary inputs, and two more are available for user expansion.

2.2.2 Arduino environment

The GCU is supplied pre-flashed with the Arduino Mega bootloader. If you install the Arduino development environment on your PC and attach a USB cable to the GCU, then you will be able to upload Arduino sketches to the GCU.

2.2.3 AVR-ISP

We encourage you to use the Arduino environment to develop enhancements to the GCU firmware. If you do not wish to use the Arduino environment, the MCU is programmable directly in-circuit using one of Atmel's AVR ISP dongles, of either the six pin or the ten pin flavor. Consult Atmel's web site for AVR development kits if you wish to develop your own firmware directly in C.

2.2.4 JTAG

The GCU features a ten pin JTAG interface for in-circuit debugging. To use this feature, you must have a compatible JTAG dongle. Consult Atmel's web site for compatible JTAG dongles.

2.2.5 Compilers

The GCU firmware libraries are developed in C, and typically compiled in the Arduino environment, which uses the GCC / GNU Make toolchain. The hardware interfacing specification is open, therefore users may develop their own firmware if they wish. Other C compilers for the AVR series include:

- IAR systems
- Imagecraft
- Codevision AVR

The "Basic App" is also compiled and built in the Arduino environment.

2.3 Thermocouple inputs

There are sixteen thermocouple inputs, labeled T0 through T15, and ideally suited for K-type thermocouples. They use a low-cost chopper stabilized differential amplifier circuit specifically designed for this application. Input is via Omega type PCC-SMP receptacles. The mating connectors are listed in section 6.3. Measurement range is from room temperature up to about 1250 degrees C.

The inputs have been designed to work with either case-grounded or case-floating thermocouples. For fastest response time, lowest cost, and best noise immunity, Allpower Labs suggests using case-grounded type thermocouples. In this case, it is important that the frame ground of your gasifier be at the same voltage as supply ground to the GCU. If your application requires floating the gasifier itself away from GCU ground, please consult Allpower Labs for further ideas and instruction.

Cold junction compensation is done in software, *assuming the cold junction temperature is 25C.* Therefore, the measurements will be in error by the amount by which the ambient air temperature differs from 25C. This was done to minimize costs, assuming most gasifier applications could live with +/- a few degrees accuracy. For more accuracy, provision has been made for a Dallas DS1821 semiconductor temperature sensor to be fitted at U307, to provide cold junction compensation based on the actual cold-junction temperature.

The thermocouple channels may be configured for other types of thermocouples via user firmware, by writing calibration routines specific to the type of thermocouple used. Alternatively, these inputs may be used to measure any low voltage differential signal, as long as the common mode voltage is between -0.3 and +0.7 volts. A good example might be current sensing using a shunt resistor.

2.4 Pressure inputs

There are six pressure sensor positions, labeled PS0 through PS5. The board land patterns are compatible with Freescale pressure sensor case types 482-01, 482A-01, 1369-01, 1351-01, and 1735-01.

Any Freescale pressure transducer with internal temperature compensation and on chip signal amplification and conditioning that fits the land pattern on the GCU may be used. Examples are MPXx7002, MPXx5010, MPXx5004, MPXx4006, etc. Consult the Freescale web site for a selection guide.

The "basic" GCU kit is supplied with one MPXV7007DP sensor (at PS0), and one MPXV7002DP sensor (at PS3). These are dual-ported sensors which read differential pressure/vacuum bidirectionally. The 7007 reads from +28 inches to -28 inches of water (+/- 7kPa). The 7002 is more sensitive (but also more fragile). It reads from +8 inches to -8 inches of water (+/- 2kPa).

The "full" GCU kit is supplied with three MPXV7007DP sensors (at PS0, PS1, and PS2), and three MPXV7002DP sensors (at PS3, PS4, and PS5).

2.5 Auxiliary analog inputs

Four auxiliary analog inputs, ANA0 through ANA3, are provided via connector ANA. They have input voltage dividers whose gain is settable via user-supplied resistors.

R603 and R604 set the input gain of ANA0. Gain is R604/(R603+R604).
R605 and R607 set the input gain of ANA1. Gain is R607/(R605+R607).
R608 and R609 set the input gain of ANA2. Gain is R609/(R608+R609).
R610 and R611 set the input gain of ANA3. Gain is R611/(R610+R611).

The divided voltages are clamped to the range of 0..5 volts, via a zener shunt protection circuit. If you wish to overdrive the voltage on the analog inputs, please be sure to select input resistors with high enough impedance to limit the input current to 30 mA or less.

2.6 FET outputs

Eight sink-to-ground type switched FET outputs are provided, at the connectors labeled FET0-3 and FET4-7. The device used is the Vishay Si4322DY, with integral Schottky diode. Vds is 30V, and Id is 18 amps, though thermal considerations will prevent sinking 18 amps continuous. Allpower Labs suggests not sinking more than 5 amps continuous per channel.

Pin 1 (labeled with a white stripe) of each terminal strip is ground. Pins 2 through 5 are the FET outputs. Pin 6 is the positive supply for flyback protection.

Flyback protection is provided by D501 through D508. Flyback reference supply should either be presented at pin 6 of the terminal strip, or JP501 (resp. JP502) should be installed to use the supply input to the board as the flyback reference. If JP501 (resp.

JP502) is installed, then the positive supply input to the board will be available at pin 6 of the terminal strip.

Overcurrent protection is provided by F501 through F508, resettable PTC fuses. As supplied, these are of type Littelfuse 30R300UU. Hold current is 3 amps, trip current is 6 amps, and the trip time is 10.8 seconds.

All FET outputs may be modulated via hardware PWM outputs of the MCU.

2.7 Display and keypad

A 4x20 alphanumeric backlit LCD display is provided, along with four softswitches below it (PB0 through PB3). This facilitates stand-alone operation, without the use of a host computer.

If more softswitches are desired for your application, a 4 by 4 matrix keypad may be connected to the connector labeled "keypad". See section 6.3 for details.

2.8 USB Host interface

The USB host interface is the primary means for the GCU to communicate with a host computer. It behaves as a USB serial port.

The host interface uses an FTDI FT232RL chip. The drivers for this chip are already included with modern versions of the Windows, MacOS, and Linux operating systems. You should not have to download or install any drivers – the GCU should appear as a serial port when you plug it in via a USB cable. Examples might be COM7 or COM8 in windows, or /dev/ttyUSB0 in Linux.

You may talk directly to the GCU using a terminal program of your choice. Set your serial communication parameters to 115200 baud, 8 data bits, no parity, 1 stop bit, no hardware flow control.

The Basic App outputs data from all the sensors in CSV format. You may log this data by setting your terminal program to save the session to a file. Or in Linux, you may log data directly from the command line by using, for instance, "cat /dev/ttyUSB0 > logfile.csv". Alternatively, you may alter the GCU Basic App, and write a host-side program to your own specifications to interface with the GCU.

2.9 Frequency counter input

A single frequency counter input is provided via the connector labeled "TIMER". This facilitates, for example, measurement of the RPM of an attached internal combustion engine (or a turbine, for that matter).

The input has a voltage divider whose gain is settable via user-supplied resistors. R601 and R602 set the input gain of TIMER. Gain is R602/(R601+R602).

The divided voltage is clamped to the range of 0..5 volts, via a zener shunt protection circuit. If you wish to overdrive the voltage on the timer input, please be sure to select input resistors with high enough impedance to limit the input current to 30 mA or less.

2.10 PWM servo outputs

Three R/C hobby style PWM servo outputs are provided, with standard pinouts. The connectors are labeled SRV0, SRV1, and SRV2. The BLACK (ground) wire of the servo should be aligned with pin 1 of the connector (labeled on the board with a white stripe).

Power for the servos is 5 volts DC, derived from the input power to the board via the main switching regulator. The drive current per servo should not exceed 1 amp continuous.

Pulse frequency is nominally 62.5 Hz (i.e. 16 ms period). Pulse width is variable from approximately 500 us, to approximately 1500 us.

2.11 CANbus

A CANbus interface is provided at the terminal strip labeled “CAN”. Presently no firmware support is provided. This might be used, for example, to interface with the motor management system on a modern automobile.

The hardware is an MCP2515 CAN controller chip, interfaced to the MCU via the SPI port.

2.12 Auxiliary RS-232 port

An auxiliary RS-232 port is provided, with a DB-9M connector. The GCU functions as DTE. Presently no firmware support is provided.

2.13 SD-card slot

An SD card slot is provided. This is intended for data logging. No firmware support will be included in the early firmware versions.

The slot is interfaced to the MCU via SPI, so code may NOT be directly executed from the SD card, and data may NOT be fetched directly from the SD card directly through the data address space of the MCU. In the future, low level I/O firmware routines will be provided for interfacing to the SD card.

2.14 Prototyping and modification area

Underneath the display (or on the back side of the board, if you wish to think of it that way), a generous breadboard area is provided. This area has approximately 800 plated-through holes, on 0.100” centers, to accommodate user circuitry.

All unused pins on the MCU are padded out to vias, to facilitate user interfacing to the MCU. Sadly, these vias are somewhat haphazardly placed, due to space restrictions on

the board. Contact APL for details on via locations, or look at the board yourself with a loupe in the vicinity of the MCU.

3 Operation in detail

XXX TBD

4 Firmware and support

The "Basic App" supplied with the GCU is written in C, and compiled in the Arduino environment.

The decision to embed in the Arduino environment was made to make it very easy for you to modify the firmware to suit your application. This chapter explains how to get started with your custom modifications.

We hope (indeed, the GPL license *requires*) that if you make generally useful modifications and/or enhancements, that you will upload your enhanced source code to the GEK wiki GCU user's community, so that others may also benefit.

4.1 Getting started

4.1.1 Obtaining and installing the Arduino environment

Go to www.arduino.cc, and download the latest version of the Arduino development environment. As of this writing, version 0.15 is current. Unzip the environment onto your hard disk.

Go to the folder where you unzipped the environment, and double click Arduino.exe. You should now have the Arduino environment up and running.

4.1.2 Obtaining and installing the source code

Go to the GEK Gasifier wiki, go to the GCU page, and find the link for the Kitchen Sink libraries zip file, and the Basic App zip file. Download these both to your hard disk.

Unzip the libraries into <arduino-install-dir>/hardware/libraries/KS

Unzip the Basic App into <My Documents>/Arduino/KS

4.1.3 Recompiling

Now, *terminate and restart the Arduino environment.* This is necessary to get it to recognize that new libraries have been installed.

In your "sketchbook", you should find a new sketch called "KS". Load it. You will see the C source for the Basic App in the editor window.

Type Ctrl-R, or select Sketch→Verify/Compile.

If all is well, after some time, the diagnostic window will display "Binary sketch size: <xxx> bytes (of a 126976 byte maximum)". If this message appears, you have successfully recompiled the Basic App.

If there are problems, scroll upward in the Arduino diagnostic window for hints as to what may have gone wrong. If all else fails, consult the GEK wiki user's community, or contact Allpower Labs.

4.1.4 Uploading to the GCU

Now, plug the GCU into a USB port on your computer. If all is well, the GCU should present itself as a serial RS-232 port. If you don't know what port number it is, try looking in "Control panel/System/Hardware/Device manager" (on Windows), or do "dmesg" (on Linux).

Once you have found the correct port name or number, tell Arduino what board you're using and where it is:

Tools→Board→Arduino Mega <<this tells Arduino what board you're using>>
Tools→Serial Port→<your port name> <<this tells Arduino what serial port it's on>>

Now, the *moment de la verité*: Click the "Upload to I/O Board" button. After a few seconds, the heartbeat LED on your GCU should go out. After a few more moments, it should start again, the diagnostic window will display a happy message, and the Basic App menu will again appear in the display.

If this all goes well, then congratulations – you have successfully replicated the entire GCU firmware development environment!

Again, if all does not go well, scroll upward in the Arduino diagnostic window for hints. If all else fails, bang your head on a wall for a few moments, then go to the GEK wiki and start asking questions.

4.2 Modifying the firmware

Modification of the firmware is accomplished by modifying the KS GCU support libraries, and/or the Basic App sketch. Please refer to section 5.2 below for details.

To review: The firmware is licensed to you under the GNU General Public License. This means, in short, that if you modify the firmware you must provide the modified source to Allpower Labs and make it available to others free of charge. This is to foster a spirit of cooperation rather than competition, and to allow the community at large to benefit collectively from the works of its members.

This cooperation is most easily accomplished by uploading your changes or enhancements to the GEK Gasifier Wiki, or to the SourceForge repository.

5 Internals

5.1 ATmega1280 I/O pin assignments

The pin assignments of the MCU are listed below, by port name. The Arduino pin assignments given are valid if you compile your Arduino sketch against the "Arduino Mega" software core. The pins listed as "N/A" are not directly accessible via the Arduino Mega core using the PinMode(), DigitalRead() and DigitalWrite() functions. However, they are still accessible using your own C code. See the heartbeat LED in the Basic App sketch for an example.

Port A: Keyscan matrix, Display. Pinned out to connector "KEYPAD"

PIN NAME	MCU PIN NUMBER	ARDUINO PIN NUMBER	GCU SIGNAL NAME	USE
PA0	78	22	SCAN0	Keyscan matrix / Display
PA1	77	23	SCAN1	Keyscan matrix / Display
PA2	76	24	SCAN2	Keyscan matrix / Display
PA3	75	25	SCAN3	Keyscan matrix / Display
PA4	74	26	KEY0	Keyscan matrix
PA5	73	27	KEY1	Keyscan matrix
PA6	72	28	KEY2	Keyscan matrix
PA7	71	29	KEY3	Keyscan matrix

Port B: Used for R/C servo outputs, AVR ISP, and expansion

PIN NAME	MCU PIN NUMBER	ARDUINO PIN NUMBER	GCU SIGNAL NAME	USE
PB0	19	53	TP19	Expansion
PB1	20	52	SPI_SCK	AVR ISP
PB2	21	51	SPI_MOSI	AVR ISP
PB3	22	50	SPI_MISO	AVR ISP
PB4	23	10	TP18	Expansion
PB5	24	11	SERVO0	SRV0 pwm output
PB6	25	12	SERVO1	SRV1 pwm output
PB7	26	13	SERVO2	SRV2 pwm output

Port C: User expansion area. Pinned out to testpads.

PIN NAME	MCU PIN NUMBER	ARDUINO PIN NUMBER	GCU SIGNAL NAME	USE
PC0	53	37	TP27	Expansion
PC1	54	36	TP26	Expansion
PC2	55	35	TP25	Expansion
PC3	56	34	TP24	Expansion
PC4	57	33	TP23	Expansion
PC5	58	32	TP22	Expansion
PC6	59	31	TP21	Expansion
PC7	60	30	TP20	Expansion

Port D: I2C, Serial, User expansion, Frequency counter input

PIN NAME	MCU PIN NUMBER	ARDUINO PIN NUMBER	GCU SIGNAL NAME	USE
PD0	43	21	SCL	I2C Expansion, TP29
PD1	44	20	SDA	I2C Expansion, TP28
PD2	45	19	TP30	Expansion
PD3	46	18	TP31	Expansion
PD4	47	N/A	TP15	Expansion
PD5	48	N/A	TP16	Expansion
PD6	49	N/A	TP17	Expansion
PD7	50	38	TIMER0	Frequency counter input

Port E: User expansion, FET PWM

PIN NAME	MCU PIN NUMBER	ARDUINO PIN NUMBER	GCU SIGNAL NAME	USE
PE0	2	0	HOST_RXD, TP11	Host USB interface
PE1	3	1	HOST_TXD, TP12	Host USB interface
PE2	4		TP13	Expansion
PE3	5	5	FET0	FET PWM output
PE4	6	2	FET1	FET PWM output
PE5	7	3	FET2	FET PWM output
PE6	8	N/A	TP14	Expansion
PE7	9	N/A	SYSCLK	System clock output

Port F: Auxiliary analog inputs, JTAG interface

PIN NAME	MCU PIN NUMBER	ARDUINO PIN NUMBER	GCU SIGNAL NAME	USE
PF0	97	54	ANA0	Aux analog input
PF1	96	55	ANA1	Aux analog input
PF2	95	56	ANA2	Aux analog input
PF3	94	57	ANA3	Aux analog input
PF4	93	58	JTAG_TCK	JTAG
PF5	92	59	JTAG_TMS	JTAG
PF6	91	60	JTAG_TDO	JTAG
PF7	90	61	JTAG_TDI	JTAG

Port G: Display control, frequency counter, user expansion

PIN NAME	MCU PIN NUMBER	ARDUINO PIN NUMBER	GCU SIGNAL NAME	USE
PG0	51	41	DISPSTB	Display
PG1	52	40	DISPRW	Display
PG2	70	39	DISPRS	Display
PG3	28	N/A	TP9	Expansion
PG4	29	N/A	TIMER0	Frequency counter input
PG5	1	4	TP10	Expansion

Port H: Aux RS232, CANbus, FET outputs, user expansion

PIN NAME	MCU PIN NUMBER	ARDUINO PIN NUMBER	GCU SIGNAL NAME	USE
PH0	12	17	SER_RXD	Aux RS232
PH1	13	16	SER_TXD	Aux RS232
PH2	14		CAN_CS	CANbus
PH3	15	6	FET3	FET PWM output
PH4	16	7	FET4	FET PWM output
PH5	17	8	FET5	FET PWM output
PH6	18	9	TP8	Expansion
PH7	27	N/A	TP7	Expansion

Port J: SD, CANbus, Aux RS232, LED

PIN NAME	MCU PIN NUMBER	ARDUINO PIN NUMBER	GCU SIGNAL NAME	USE
PJ0	63	15	SD_CAN_MISO	SD/CANbus
PJ1	64	14	SD_CAN_MOSI	SD/CANbus
PJ2	65	N/A	SD_CAN_SCK	SD/CANbus
PJ3	66	N/A	SD_NSS	SD
PJ4	67	N/A	SER_CTS	Aux RS232
PJ5	68	N/A	SER_RTS	Aux RS232
PJ6	69	N/A	CAN_INT	CANbus
PJ7	79	N/A	LED	LED output

Port K: Temperature, Pressure, Aux analog

PIN NAME	MCU PIN NUMBER	ARDUINO PIN NUMBER	GCU SIGNAL NAME	USE
PK0	89	62	AT0	Temperature
PK1	88	63	AT1	Temperature
PK2	87	64	AT2	Temperature
PK3	86	65	AT3	Temperature
PK4	85	66	AP0	Pressure
PK5	84	67	AP1	Pressure
PK6	83	68	TP5	Analog expansion
PK7	82	69	TP4	Analog expansion

Port L: FET output, Dallas 1wire, Temperature, Pressure, User expansion

PIN NAME	MCU PIN NUMBER	ARDUINO PIN NUMBER	GCU SIGNAL NAME	USE
PL0	35	49	TP3	Expansion
PL1	36	48	TP2	Expansion
PL2	37	47	TP1	Expansion
PL3	38	46	FET6	FET PWM output
PL4	39	45	FET7	FET PWM output
PL5	40	44	DQ	Dallas 1wire interface
PL6	41	43	MUXA	Temperature / Pressure
PL7	42	42	MUXB	Temperature / Pressure

5.2 Firmware internals

5.2.1 Boot loader

The KS GCU is supplied with the "Arduino Mega" bootloader pre-loaded into flash memory. The Arduino bootloader is protected by the AVR's flash-protection fuse pattern, so the bootloader doesn't erase itself when you upload new code to the board.

If you somehow accidentally zorch the bootloader, you must upload a new one using an AVR-ISP programming dongle. Consult the Arduino documentation on how to do this.

5.2.2 Arduino core

As of this writing, the Basic App is compiled against the "Arduino Mega" core files. These are located in <<Arduino-install-dir>>/hardware/cores/arduino.

One drawback to this is that not all the ATMega's I/O pins are mapped into Arduino's "Digital I/O" pin space (see the pin assignments below for details, or consult the core file "pins-arduino.c"). Allpower Labs will eventually write a set of core files for the KS GCU, or you may do so yourself. If you do, please post them to the GCU Wiki page, or upload them to SourceForge.net/kitchensink.

If you wish to simply work around this problem, you can use the native AVR-style port macros to read and write pins that are not mapped into the Arduino's "Digital I/O" pin space. These macros are defined in <avr/io.h>, found at

<<Arduino-install-dir>>/hardware/tools/avr/avr/include/avr/io.h

See the Basic App source code for an example.

5.2.3 Libraries

The KS GCU Basic App uses a set of support libraries to allow I/O access to all the GCU's hardware functions. The Basic App was developed directly in C before it was ported to the Arduino environment; hence, these support libraries do not make use of the Arduino core files.

It is Allpower Labs intention to eventually modify the KS GCU libraries to make use of the Arduino core files, and of other Arduino libraries which others have already written (for instance, the LiquidCrystal.cpp display library, and the Servo PWM library).

Until this time, please feel encouraged to port portions of the Basic App and the supporting libraries to the Arduino core. As usual, if you do so, please upload your source code to the GCU Wiki page, or to SourceForge.net/kitchensink, so that others may benefit from your efforts.

The KS GCU support libraries are as follows. They are located where you installed them, *i.e.* in <<Arduino-install-dir>>/hardware/libraries/KS.

- Adc.c: functions to set the on-board four-to-one analog multiplexers, and to read the on-chip A/D converter
- Display.c: functions to drive the LCD display (Hitachi HD44780 compatible)
- Fet.c: functions to turn the FETs on and off (eventually, to do PWM control)
- Keypad.c: supports synchronous or asynchronous scanning of the keypad
- Pressure.c: supports the on-board pressure sensors
- Temp.c: supports the on-board temperature sensors
- Servo.c: not written yet. A stub for PWM servo control
- Timer.c: not written yet. A stub for the timer/counter RPM input routines
- Ui.c: a set of menu screens which support the menus found in the Basic App. In theory, these should be eventually moved out into the sketch directory, rather than being treated as a library. I simply have not figured out how to compile a sketch from multiple source files yet. If you figure it out, please let the GCU community know.

5.2.4 Basic App Arduino sketch

After all that, the Basic App is really quite simple. It is written using the Arduino core idiom, of a setup() function and a loop() function.

The setup() function should be self-explanatory.

The loop() function has basically five parts:

- Read all the various on-board sensors (and stash their values into statically allocated arrays defined by the various I/O modules)
- Display a User Interface screen on the built-in LCD display (menu state is stored statically internal to the ui.c module)
- Scan the keypad for a key, and handle the key if necessary (key handler is at the moment internal to the ui.c module – it manipulates the static UI menu state)
- Write all the various on-board actuators (using values from statically allocated arrays defined by the various I/O modules)
- Log data to the USB serial port (once per second)

This dispatch loop is executed as fast as possible, but with a 100 ms delay. Note the dispatch loop also toggles the state of the heartbeat LED each time it executes. This gives direct visual indication of computational progress.

The main loop() calls the data logging routine, not on every loop execution, but once per second. It does this by checking the return value from the Arduino real-time counter millis(), and calling the logging routine only if it's time to log another data point.

Modification style is basically up to you. Two simple and useful ideas are:

SD Data Logging: Modify the logging routine to write the on-board I/O data into a file on the on-board SD card.

Automated Measurement and Control: Write a control function to effect control based on measurements. Examples might be a thermostat, an automated fuel feed, an RPM governor, *etc.*

Insert your control function into the main dispatch loop(), after reading inputs, but before writing outputs. It can get the values of all the on-board inputs by referring to the input value arrays (*i.e.* Temp_Data[], Press_Data[], *etc.)*, and it can arrange for outputs to be effected by writing the output value arrays (*i.e.* Servo_Data[], Fet_Data[], *etc.*).

If you're going to keep the existing on-board menu system, be sure to think carefully about how your automated outputs will interact with it. You may wish to disable certain control items in the menu system, or establish some means of manual override, or you might choose to not worry about it at the software level.

Good luck, and Happy Hacking!

6 Appendices

6.1 Warranty

XXX TBD

6.2 Resources on the web

6.2.1 Allpower Labs

http://www.allpowerlabs.org

6.2.2 GEK

http://gekgasifier.pbworks.com

6.2.3 Kitchen Sink on Sourceforge

http://sourceforge.net/projects/kitchensink/

6.2.4 Arduino

http://www.arduino.cc

6.2.5 Atmel

http://www.atmel.com

6.2.6 GCC homepage

http://gcc.gnu.org

6.2.7 WinAVR homepage

http://winavr.sourceforge.net/

6.2.8 IAR systems

http://www.iar.com

6.2.9 Omega

http://www.omega.com

6.2.10 Freescale Semiconductor

http://www.freescale.com

6.2.11 FTDI

http://www.ftdichip.com

6.3 I/O connector pinouts

PWR

PIN IDENTIFIER	PIN NAME	DESCRIPTION
Center	+V_in	Positive voltage supply, 7 to 30VDC
Outside	GND	Ground

T0 through T15

PIN IDENTIFIER	PIN NAME	DESCRIPTION
Long slot	K	Thermocouple -
Short slot	A	Thermocouple +

PS0 through PS3

FITTING IDENTIFIER	FITTING NAME	DESCRIPTION
Bottom tube (nearest PC board)	"Vacuum" (P2) side	Positive pressure here relative to P1, gives a negative reading
Top tube (away from PC board)	"Pressure" (P1) side	Positive pressure here relative to P2, gives a positive reading

FET0-3

PIN NUMBER	PIN NAME	DESCRIPTION
1 (indicated by white stripe)	GND	FET0 – FET3 ground return
2	FET0	FET0 sink-to-ground output
3	FET1	FET1 sink-to-ground output
4	FET2	FET2 sink-to-ground output
5	FET3	FET3 sink-to-ground output
6	FET0-3_FLYBACK	Positive flyback reference voltage for FET0 – FET3. If JP501 is installed, this is taken from the onboard power jack. If JP501 is absent, this pin must be externally connected to the positive supply for the devices being switched by the FET0 – FET3 outputs.

FET4-7

PIN NUMBER	PIN NAME	DESCRIPTION
1 (indicated by white stripe)	GND	FET4 – FET7 ground return
2	FET4	FET4 sink-to-ground output
3	FET5	FET5 sink-to-ground output
4	FET6	FET6 sink-to-ground output
5	FET7	FET7 sink-to-ground output
6	FET4-7_FLYBACK	Positive flyback reference voltage for FET4 – FET7. If JP502 is installed, this is taken from the onboard power jack. If JP502 is absent, this pin must be externally connected to the positive supply for the devices being switched by the FET4 – FET7 outputs.

USB

PIN NUMBER	PIN NAME	DESCRIPTION
1	V_BUS	USB bus power
2	D-	USB inverted data
3	D+	USB positive sense data
4	GND	Signal ground

SRV0 through SRV2

PIN NUMBER	PIN NAME	DESCRIPTION
1 (indicated by white stripe)	GND	Servo ground return (black wire on hobby servoes)
2	+V	Servo power (+5VDC, up to 2A) (red wire on hobby servos)
3	PWM	Servo PWM data (0.5 to 1.5 ms pulse width) (white wire on hobby servos)

TIMER

PIN NUMBER	PIN NAME	DESCRIPTION
1 (indicated by white stripe)	GND	Ground
2	TIMER_IN	Timer input signal (input to user-programmable resistor divider network R601, R602)
3	GND	Ground

CAN

PIN NUMBER	PIN NAME	DESCRIPTION
1 (indicated by white stripe)	CAN_L	CAN inverted-sense data
2	GND	Ground
3	CAN_H	CAN positive-sense data

ANA

PIN NUMBER	PIN NAME	DESCRIPTION
1 (indicated by white stripe)	AGND	Analog ground reference
2	ANA0	Auxiliary analog input 0 (input to user-programmable resistor divider network R603, R604)
3	ANA1	Auxiliary analog input 1 (input to user-programmable resistor divider network R605, R607)
4	ANA2	Auxiliary analog input 2 (input to user-programmable resistor divider network R608, R609)
5	ANA3	Auxiliary analog input 3 (input to user-programmable resistor divider network R610, R611)
6	+5V	+5VDC utility output

ISP1

PIN NUMBER	PIN NAME	DESCRIPTION
1 (indicated by white dot)	MOSI	Master out, slave in
2	VTG	
3	GND	Ground
4	GND	Ground
5	RST	Processor reset
6	GND	Ground
7	SCK	SPI Serial clock
8	GND	Ground
9	MISO	Master in, slave out
10	GND	Ground

ISP2

PIN NUMBER	PIN NAME	DESCRIPTION
1 (indicated by white dot)	MISO	Master in, slave out
2	VTG	
3	SCK	SPI Serial clock
4	MOSI	Master out, slave in
5	RST	Processor reset
6	GND	Ground

JTAG

PIN NUMBER	PIN NAME	DESCRIPTION
1 (indicated by white dot)	TCK	
2	GND	
3	TDO	
4	VREF	
5	TMS	
6	SRST	
7	VCC	
8	TRST	
9	TDI	
10	GND	

KEYPAD

PIN NUMBER	PIN NAME	DESCRIPTION
1 (indicated by white dot)	KEY0	Keyscan matrix, column 0 (left)
2	KEY1	Keyscan matrix, column 1
3	KEY2	Keyscan matrix, column 2
4	KEY3	Keyscan matrix, column 3 (right)
5	SCAN0	Keyscan matrix, row 0 (onboard)
6	SCAN1	Keyscan matrix, row 1
7	SCAN2	Keyscan matrix, row 2
8	SCAN3	Keyscan matrix, row 3

RS232

PIN NUMBER	PIN NAME	DIRECTION	DESCRIPTION
1	N/C		
2	RXD	Input	Received data
3	TXD	Output	Transmitted data
4	N/C		
5	GND		Signal ground
6	N/C		
7	RTS	Output	Request to send
8	CTS	Input	Clear to send
9	N/C		

6.4 Firmware repository

The firmware source code repository will be located at sourceforge.net, under the project name “kitchensink”.