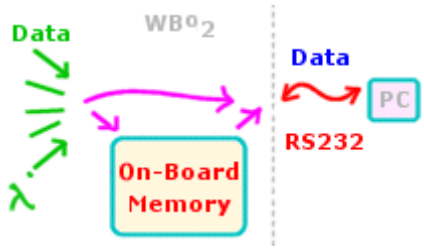
WBo2 Logging Specifications

Back | Buy |



All Tech Edge's Wideband (WBo2) units include some form of logging capability. Some have **on-board** logging memory (2A0, 2B0), and others don't (2C0, 2D0, 2E0) but they all generate an RS232 **data stream** that can be sent to a collection device (a PC, etc.) for later analysis.

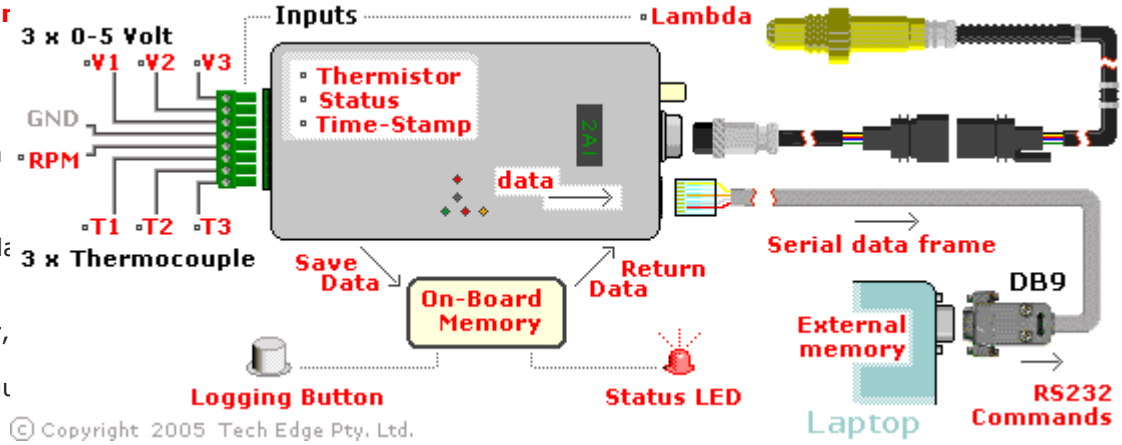
This page describes WBo2's **integrated logging capability** including specifications for the RS232 **data frame** used. Go here for newer info on the command for the 1 Mbyte logging module (applies to 2A1 and rev 3 units with on-board logging memory)

The **on-board logging** functionality (which has been standard since early 2004) is also described. From September 2005 a number of on-board memory sizes are available.

Note: Some of the following paragraph refer to the 2A0 unit, but they can equally refer to other units with on-board (eg. 2B*n*) and serial logging capability (eg. 2E0). For specific hardware information about your particular model.

Overview - On-Board Memory & Serial Logging

Serial Loggir
The 2A0 WBo2 unit has 8 sources of external data it can log.



- Lambda from the sensor,
- 3 analogu 0-5 Volt channels,
- an RPM channel,
- and 3 thermocouple channels.

The green connector at one end of the unit provides the physical connections to 7 of those external data inputs. A time-stamp, units status and the unit's internal temperature are also available. The collected data is formed into a serial data frame and sent out the RS232 port (the RJ45 connector). A number of data frame types may be produced, but the default frame is interpreted by either TEWBlog software or one of the intelligent displays (either LA1 or LD02). Any time the unit is powered, it generates these data frames. The power-up frame type, and the frame repeat rate, may be changed, but remember the displays only interprets the default frame type.

On-Board Logging : Although data is always pumped out the serial port, that data can also be stored in memory *on-board* the WBo2 unit. The saving of on-board logging data is controlled by the logging button (shown at left for the 2A0 unit), and under external software control, returned at a later time for display/analysis.

All but the very earliest 2A0 units had a **32 k byte** on-board memory (those early units can be upgraded to 32 k or even the newer 1 M size). The 32 k on-board memory can store over *1,150 frames* of *28 bytes* each (the size of the default data frame). That's about 115 seconds, or just less than 2 minutes, at the default logging rate of 10 frames/sec. The Logging rate can be varied up to around 50 frames/sec (23 seconds of logging) or down to less than a frame every 2 seconds (about 40 minutes). Logging software can retrieve logged data and send it out the unit's serial RS232 port (at 19,200 bits/sec). It typically takes less than

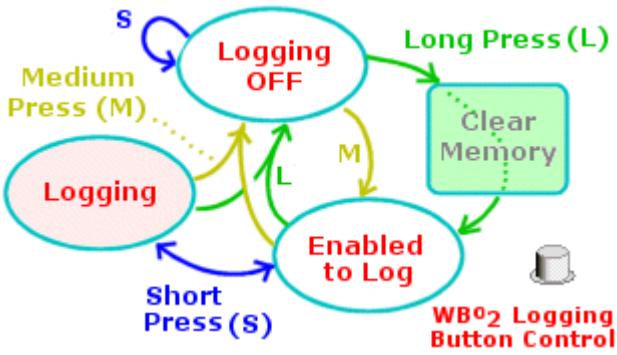
20 seconds to download the maximum data that can be stored.

Logging Control - Button & Serial Commands

Two ways to control on-board logging are possible :

- **The logging button** controls start, stop, and clearing of on-board memory. One of the buttons on intelligent displays also operates in a similar manner. (Refer to the display's user guide for more info - LA1, LD02)
- **Serial (RS232) commands** can also do the same, but importantly, can also command the downloading of saved data. Note that intelligent displays use RS232 commands to perform the logging functions. Serial command, are described fully below.

Because there is only one button, and because we want to preserve existing data that may be stored in memory, a combination of **short**, **medium**, and **long** button presses commands the WBo2 on-board logger to enter one of three states - **Logging OFF**, **Logging Enabled** and **Logging**. Refer to diagram at right



- A **Short** press is from **0.2** to **1.5** seconds (*blue arrows*).
- A **Medium** press is from **1.5** to **4** seconds (*yellow arrows*).
- A **Long** press is greater than **4** seconds (*green arrows*).

To understand what the diagram means, here are the simple steps to using on-board logging. The normal power-up state for WBo2 is the *logging OFF* state :

1. From the *logging OFF* a **short** press is ignored to prevent accidently filling memory.
2. A **medium** press puts the unit into the *enabled to log* state.
3. **short** presses **start** *logging* - the Red LED does a double flash (see below).
4. Another **short** press **stops** *logging* - multiple **sessions** are possible.
5. a **medium** (or **long**) press returns to the *stopped* state.
6. a **long** press from the *stopped* state enters the *enabled* state **AND clears all recorded data** (use before a new recording session).

LED Status During On-board Logging

On WBo2 units with on-board logging memory, the **red** LED, which normally shows wideband status, flashes according to a different pattern when on-board logging is enabled.

Enabled to log

- **Logging OFF** - red LED is **steady** (LED shows normal WB operation).
- **Enabled to Log** - red LED **slow even flash**.
- **Logging** - red LED **faster double flash**.
- **Clearing Memory** - red LED **fast triple flash**.

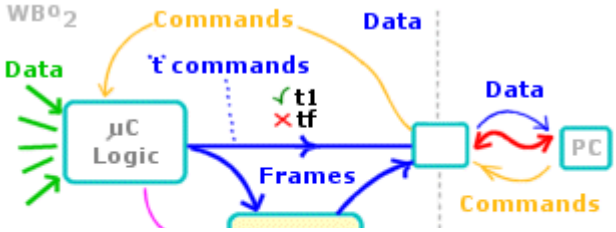
Logging to Memory

The status LED will flash this way even if RS232 commands (eg. from a display) enabled on-board logging.

Software Control of Logged Data

It is important to differentiate between two classes of commands that control the data saved & generated by WBo2 :

- **Trace Commands** : control the format of **data frames** produced by the wideband unit (and optionally saved in on-board memory).



They are described in the next section. Once saved, the format of that data cannot then be changed.

- **Logging Commands** : control the actual saving and restoring of data to/from the wideband unit's on-board memory. These commands (x, y & z) are described in the section following.

Serial Data Control (Trace Commands)

Even if the WBo2 unit does not have on-board logging memory, RS232 logging commands control the operation of the serial **data frame** allowing it to be stopped, started and its repeat rate changed. The data frame produced can be selected from among a number of preset types including a version 1.5 frame that is compatible with older WBo2 units. An option to produce an ASCII/Hex version of each frame, but without the CRC and frame header information, is available for debugging and human readable logging using software like *HyperTerminal*.

Command	Action
t*n*	Trace command - stop/start/one-shot logging to serial port (and to on-board memory if on-board logging enabled). n = **f**, (ie. **tf**) stops logging - the Auto Frame generation bit is reset to **0** - see below. n = **1**, (ie. **t1**) start continuous logging - Auto Frame generation bit is set to **1**. n = **0**, (ie. **t0**) log just one single frame - Works only if frames are not being generated already.
s03*xx*	Set logging control byte to xx (hex) where xx bits are defined (default = **bold**) : b**7** - Frame generation, **1** = **Auto** (always produced), **0** = Manual. b**6** - Frame type, **1** = **Binary**, **0** = hex/ASCII. b**5** - USR inputs, **1** = scale to **1.5 format** (scaled x8 = 13 bits), **0** = raw. b**4** - Frame mode, **1** = debug mode (ASCII/Hex), (*See* **** note*** *below*) **0** = **normal** (binary). b**3-0** Frame type (*See* **** note*** *below*) **0** : V1.5 compatible mode. **1** : **V2.0 mode**.
s04*xx*	Set the logging delay between frames to xx (hex) in units of 10 mSec. Minimum value is 1, maximum is 100 (0x64).

**** note :*** Combinations of the **s03** command's **xx** bits produce the following **modes**

s03**e0**	1v5 Binary mode, used for 1.5 logging.
s03**e1**	2v0 Binary Mode, (default mode, used by intelligent displays).
s03**80**	1v5 ASCII HEX Mode, as above but ASCII readable.
s03**81**	2v0 ASCII HEX Mode (Human readable version of above **E1**).
s03**82**	CAL ASCII HEX Mode, produces calibration information.
s03**84**	Status ASCII HEX Mode, PID and error status information.

**** note :*** In ASCII/Hex mode the frame header (2 bytes) and frame sequence counter byte are not transmitted, neither is the CRC (checksum). Data is transmitted as groups of two bytes, ie. as 4 Hex characters. This results in a clean human readable display for most frames as most displayed data is 16 bits long.

Refer to the WBo2 software page for more information on available utilities.

Note that the ***sxxyy*** commands (such as s03e0) are generic commands to set configuration byte **xx** to the value **yy**. They are often followed by the **w** command used to permanently save configuration memory.

On-Board Memory Control (Logging Commands)

On-board logging can be stopped and started via the WBo2 unit's press button. RS232 commands also control logging to the on-board memory as well as later retrieval of that stored data by, say, a PC.

The 32 k byte EEPROM is read or written as **512 pages** of **64 bytes (size=$40)** but is **addressed**, using a **logging pointer**, in a linear fashion from $**0040** to $**7FFF**. Note that as the first 64 byte page (address 00 to $003F) in the EEPROM is reserved for working pointers, the smallest value for the logging pointer is therefore $0040.

Data is saved in frame sized chunks (the **green rectangles** in the diagram). Each new **log session** simply appends new data to the previously recorded data. The logged data frame's time stamp will help show you where a log session's data starts and stops.

Logging occurs at the speed, and mode, specified in WBo2's configuration RAM locations 3 and 4 (see the previous "Software Control of Logging" section). Note: no compression is used for storing data and multiple formats may be saved. This also means the storage time can be easily calculated.

Command	Action
x*n*	Stop/Start logging of data to on-board memory. n = **0**, stop logging (ie. **x0**) - same as pressing the **logging button** to STOP logging. n = **1**, start logging (ie. **x1**) - same as pressing the **logging button** to START logging. n = **2** (or more) returns the current logging pointer value as a 4 digit hex value and is useful to determine how much logging memory remains.
y*wwww*	Set logging pointer to *wwww* (eg. minimum y is $0040, maximum $7FFF). Normally the **y0040** command is used to clear logged data memory in preparation for a new recording session and is equivalent to pressing the on-board logging button (from the logging OFF state) for more than 4 seconds.
z	Dumps all logged data. This is from y=$0040 to the logging pointer's current value. The logging pointer is not reset by this command.

From September 2005 some units are available with larger memory capacity, such as the 2A1 with 1 Mbyte of on-board memory. Refer to the 2A1 technical information for the additional commands required.

Serial Logger Frame Format - 1.5 & 2.0 Modes

The onboard microprocessor produces a serial RS232 data stream at **19,200** baud (8 bit, no parity). A version **1.5 mode frame** format is produced for compatibility with version 1.5 software. The version **2.0 mode frame** format is slightly different (in particular the wideband value) and adds the thermocouple information not available on the 1.5 unit. Other enhancements include error status, and of course there are now a number of different frame formats that may be sent.

The bytes 4 & 5, 6 & 7 and 8 & 9 are really high and low byte pairs (ie. Motorola format) of four 16 bit values. Each of these two bytes is really a 13 bit value in the range 0 to 8191 representing a voltage between 0 and 5.00 Volts. The actual measurement is made at an accuracy of 10 bits and is averaged over 8 samples for each of the 4 inputs (ie 32 interleaved samples).

The first 3 (bytes 1, 2 & 3) and last (CRC byte 12) bytes allow the receiving computer to correctly synchronise with data and to check that data has not been corrupted.

1.5 Data Frame Format

Bytes **1** & **2** are the fixed **frame signature** bytes of 0x5A, 0xA5 (ie. hex values). These value were chooses as they have alternating 0->1 bit patterns.

byte **3** is the **frame sequence counter** and simply counts from 0 to 255 and then repeats, and allows the receiver to keep track of possible lost data frames. All these values are included in the CRC calculation.

Bytes **4** and **5**, the **SVout ADC** value, is compatible with the version 1.5 unit (and the 5301/LD01 displays). SVout is described in the DIY-WB 1.0/1.5 section as the **Vout** signal. The SVout value (ie. Vout) varies from a minimum rich value of 0x068E (=1678 decimal, so SVout is *5*1678/8192=1.02 Volts*) to maximum lean (ie. free air) value of 0x199A (=6554 decimal, so SVout is

*5*6554/8192=4.00 Volts*). The Min and Max SVout values are clamped by the firmware.

Bytes **6** through **9** are the two 0 to 5.00 Volt **User Inputs** U1, U2 (U3 is not returned) On the 1.5 unit these inputs are really a 10 bit value that is sampled rapidly 8 times and summed. On the 2.0 unit there is just one 10 bit sample for each channel, but the result is shifted 3 places to be compatible with the 1.5 data. (ie. there are 1024 steps - 0, 8, 16 .. 8176, 8184) for version 1.5 compatibility.

Bytes **10** and **11**, the **RPM count** measure how many 5 microsecond time periods are counted between successive positive pulse edges on the COIL or RPM Pulse input pins. As the frequency of the pulses increases (RPM goes up), the count goes down. For a four cylinder 4 stroke engine (that produces two spark events per revolution), at 6,000 RPM there are 100 revolutions per second and 200 spark events and therefore the COIL input will measure 5,000 microseconds between sparks, or an RPM count of 1,000 for bytes 10 and 11. *Note* that if the pulse rate is lower than 20 per second then the unit will "time out" and return an invalid count.

1 - Frame **Header** byte 1 (0x5A)
2 - Frame Header byte 2 (0xA5)
3 - Frame **Sequence** counter
4 - **SVout** ADC high
5 - SVout ADC low
6 - **User 1** ADC high byte
7 - User 1 ADC low byte
8 - **User 2** ADC high byte
9 - User 2 ADC low byte
10 - **RPM** count high byte
11 - RPM count low bye
12 - **CRC** (1's comp. sum of above)

Final **CRC byte 12**, when added to all the other bytes in the frame, and truncated to 8 bits, should result in the value 0xFF. If not then at least one bit in the frame has been zapped and the frame's data should not be trusted. As the sequence counter constantly changes, then the CRC value will also change.

The native mode for WBo2 (ie. version **2A** units) is the **2.0 mode frame** format as described below. Note carefully the differences between the 2.0 frame's **Wideband ADC** and the 1.5 frame's **SVout** values, and observe that 2.0's **Ipx** value is a closer match to the 1.5's **SVout** value.

2.0 Data Frame Format

1 - Frame **Header** byte 1 (0x5A)
2 - Frame Header byte 2 (0xA5)
3 - Frame **Sequence** counter
4 - **Tick** high byte (1 tick = 1/100 Second)
5 - Tick low byte
6 - **Wideband** ADC high byte
7 - Wideband ADC low byte
8 - **Ipx** high byte (8192=F/A, 4096=stoich)
9 - Ipx low byte
10 - **User 1** ADC high byte (**V1** input)
11 - User 1 ADC low byte

The **first 3** bytes (**frame ID**, & **frame sequence counter**) and last byte (**CRC**) are as described above.

Bytes **4 & 5** count relative time in **Ticks** - measured in 1/100 of a second (ie. 10 mSec/tick). The count goes from 0 to 65535 (unsigned) and will overflow in just under 11 minutes (10 minutes, 55 seconds & 35 mSec) and should be sufficient to act as a timebase for most logging tasks.

Bytes **6** & **7** - **Raw Wideband ADC** value. This is the 12 bit value (0 to 4095) that is applied to the DAC (Digital to Analogue converter) and represent the voltage in mVolts produced by the DAC. As the DAC is followed by a amplifier with a gain of (68+15)/68, a count of 1 represents 1.22059 mV and 4096 represent 5.000 Volts. Note that the WBlin value is looked up in the WB unit's EE PROM tables using the pump current (Ipx) as index.

Bytes **8** & **9** **Normalised Pump Current** (Ipx) - After a calibration factor has been applied to the raw pump current, the resulting normalised pump current Ipx has a free-air value of 8192 and a stoich value of 4096. A value of 0 represent the richest condition that can be sensed by the WB unit (less than AFR=10). Ipx is used to calculate WBlin (bytes 6 & 7) and other values too. The Ipx to Lambda table lsu_ipx_LP as a text file (or, as an eXcel file) shows how Ipx relates to Lambda (or AFR) and also % oxygen.

12 - **User 2** ADC high byte (**V2** input)
13 - User 2 ADC low byte
14 - **User 3** ADC high byte (**V3** input)
15 - User 3 ADC low byte
16 - **Thermocouple 1** ADC high (**T1** Input)
17 - Thermocouple 1 ADC low
18 - **Thermocouple 2** ADC high (**T2** Input)
19 - Thermocouple 2 ADC low
20 - **Thermocouple 3** ADC high (**T3** Input)
21 - Thermocouple 3 ADC low
22 - **Thermistor** ADC high (internal sensor)
23 - Thermistor ADC low
24 - **RPM** count high byte
25 - RPM count low bye
26 - **Status**/Error high byte
27 - Status/Error low byte
28 - **CRC** (1's comp. sum of above)

Bytes **0** through **15** are the three 0 to 5.00 Volt **user inputs U1, U2** & **U3** (there's an additional user input available on version 2.0). Although these inputs are sampled just once, they are shifted to be in the range 0 to 8184 (ie. 1024 steps - 0, 8, 16 .. 8176, 8184) for version 1.5 compatibility.

Bytes **16** through **21** are the three **thermocouple inputs T1, T2** & **T3**. The raw thermocouple voltage is amplified by a factor of 101 before sampling by the ADC at 10 bits of resolution, giving a value from 0 to 1023 (1024=5.00 Volts). For a K-type thermocouple, a sampled count of 512 corresponds to a thermocouple voltage of (512/1024)*5.00/101 = 24.75 mV and, after consulting tables, this is an uncompensated thermocouple temp of about 596 °C. A count of 1023 gives 49.46 mV or a temp of about 1217 °C

Byte **22** & **23** is the **10 bit thermistor count** and can be used for cold junction compensation for calculating the thermocouple's absolute temperature. Refer to eXcel spreadsheet thermist.xls for further information.

Bytes **24** & **25** The **RPM count** has the same format as the 1.5 frame (ie. count 1 = 5 µSec) but with the additional feature of averaging two successive triggering events to cater for odd fire engines that may be triggered from the one coil.

Bytes **26** & **27** **Status/Error bytes** see Error Codes section below.

Error Codes and PID Operational Status

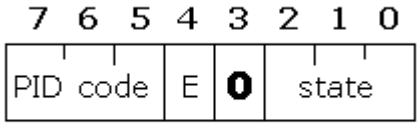
WBo2's hardware and software contains two **PID**s (Proportional, Integral & Derivative) feedback controllers for both **Lambda sensing** and also precise **heater temperature** control. Each PID samples a **signal** from the Lambda sensor and in turn provides an **output** that in turn controls that signal :

- Lambda measurement - **Vs** (Nernst cell sensing) for **Ip** (pump current) control.
- Heater temperature - **Ri** (internal sensor resistance) for heater **PWM** (Pulse Width Modulation) control.

Because both Ri and Vs are meaningless until the sensor has reached a minimum operating temperature, the two PIDs also operate in a startup mode. During startup the sensor is run in *open loop* mode where heater current is closely controlled to ensure the fastest warmup while remaining within the sensors operating envelope.

Feedback control relies on a measured signal being slightly different from the **setpoint** or target value. One measure of the PID's current status is how far away the controlled value is from this setpoint. In the case of the Vs/Ip feedback loop, the Vs setpoint is **450 mVolts**, and for the Ri/PWM feedback loop the Ri setpoint is **80 ohms**.

Two PID status bytes (bytes 26 and 27 of the logged data frame) are available for possible debugging. Byte 26 is for the Vs/Ip (or wideband) PID, and byte 27 is for the Heater PID. They take the general form shown at right (numbers 7 to 0 are the bit positions within each byte).



Bits 7 to 5 represents the **PID code** which takes the following values:

Bit 765	PID Operation
000	operating normally

001 Integral accumulator at **lower** clamp value
010 Integral accumulator at **upper** clamp value
011 Output control value at **lower** clamp
100 Output control value at **upper** clamp

Bit 4 (**E**) is set when the IPID error band is exceeded (but this in itself is not an error).

Bit 3 (**0**) is always zero.

Bits 2 to 0 reflect the current **PID state** and is different for each byte/PID. They are described in the following section.

Value	Byte 26 -- Vs/Ip PID (wideband) State	Red and Amber LED action
0	**null/off** state, only seen at system start-up or during debugging	***slow, even*** *RED flash*, *AMBER* ***off***
1	**Sense** state, looking for presence of sensor.	***Even*** *RED flash* and ***very short*** *AMBER flashes*
2	**Cold** state, heating up sensor, PIDs not yet operational	***Short*** *RED flash*, *AMBER shows heater* ***on*** *time*
3	**Warm** state, PIDs operational	*RED* ***solid ON*** = PIDs OK or, *RED* ***blinking*** = PID(s) error band exceeded
4	**Config** state, used for system testing	***Double*** *RED flash.*
5	**unused**	*nil*

Note that the "blinking" LED in the operational state does not indicate that either of the PIDs are unlocked, per se, but that one of the PIDs is operating with its error term greater than a pre-programmed threshold value.

Value	Byte 27 -- Heater PID State	Red and Amber LED action
0	**Normal Operation** state	***slow, even*** *RED flash*, *AMBER shows heater* ***on*** *time*
1	**Vbatt high**	***Even*** *RED flash* and ***very short*** *AMBER flashes*
2	**Vbatt low**	***Even*** *RED flash* and ***very short*** *AMBER flashes*
3	**Heater short**	
4	**Heater open circuit** (sensor not present)	
5	**FET failure**	
6	**unused**	*nil*

Thermocouple & Thermistor Information

Thermocouples generate a voltage that depends on the difference between the "hot" junction where two dissimilar metals are joined, and the "cold" junction. Basic data for K-type thermocouples is available from NIST at srdata.nist.gov/its90/main/its90_main_page.html.

The onboard thermistor has a nominal resistance of 47 k ohm at 25°C. It can be configured to measure the actual temperature of the WB unit but its primary purpose is to measure the thermocouple "cold junction" which is roughly the temperature of the circuit board. This eXcel spreadsheet of thermistor data describes the sensor used and

Adding 2A0 (rev-2 PCB) On-Board Logging Hardware

The original (Rev-2 PCB) 2A0 DIY kit version came without on-board logging, this section describes how to update those units. This section has moved (Jan 26 2004). Please refer to the WBo2 Logging Hardware page for more information.

For older 2A0 units that did not come with any on-board logging facility, see the 32 k byte Logger Hardware section for DIY details. Both older and new 2A0 models, as of September 2005, can be

upgraded to 1 M byte of logging memory.

Page updated 17 Jan 2007

Page created 6 Feb 2004 | *Tell me about broken links*

Back | WBo2 | Contact | Feedback | Copyright © Tech Edge