Yamaha Electronic Musical Instruments

E-bus internal communications bus

E-bus based on I2C (inter-Integrated Circuit) Bus
Technical Overview

1. Introduction

The methods used to interconnect the internal circuit boards of user-interfaces such as the keyboard PCB and switch/LED panel PCBs to the rest of the circuitry, are different in each product.

Figure 1 shows the method used in older keyboards, where each key and panel switch/LED was scanned as part of a matrix that required many connections back to the scanning Main or Sub-CPU.

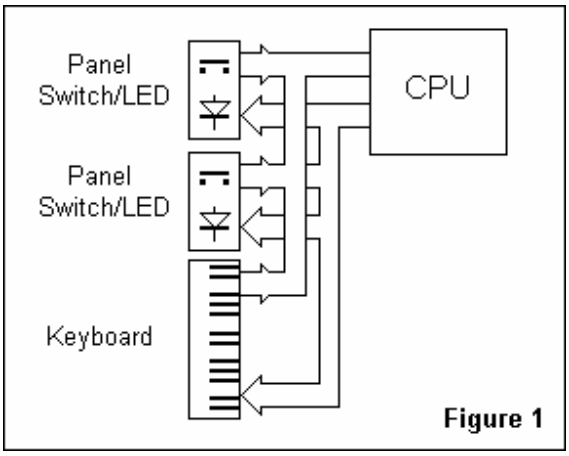

Figure 1

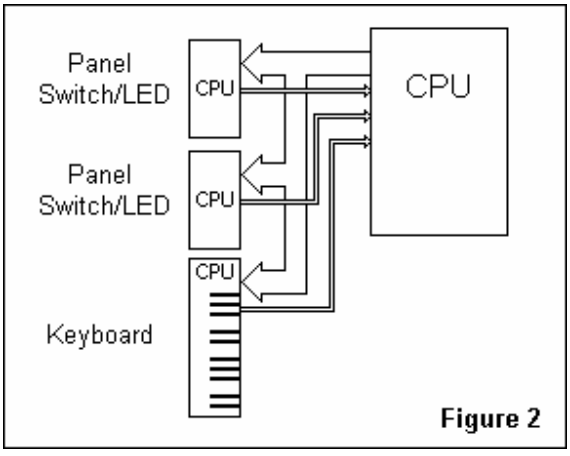

Figure 2

For example, an 88-note velocity-sensitive keyboard was usually scanned as a 12-note by 8-octave by 2-make/break-contact matrix: a total of twenty-two connections just for the keyboard alone.

This method evolved into that represented by Figure 2, in which individual scanning CPUs were located on each panel and keyboard PCB. The number of outgoing connections was considerably reduced compared to the previous method because the key/panel data was transmitted serially back to the main CPU. However, to identify its function in the system, each PCB still required several additional discrete lines. The Electone EL90 Organ and later models used this method.

Both of the methods just described required different hardware to accommodate even minor differences between the specifications of each model-level in a given product range.

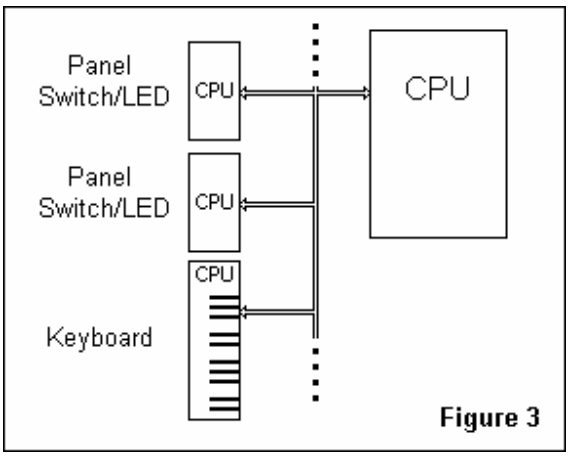

Figure 3

Figure 3 shows a system implemented using E-bus, which is a true bus system in which a practically unlimited number of devices (integrated circuits) share common connections. E-bus is a serial bus standard in musical instruments based on the I2C standard. It features:

- High-speed serial communication
- Few signal lines
- Interactive communication
- Communication specifying the device

Using E-bus, the designer can unite both hardware and software aspects of each interface to provide commonality of PCB/assemblies for different models.

2. About the I2C bus

The I2C (pronounced "I *squared* C", not "I *two* C") bus standard was developed by Philips Semiconductors in the early 1980s. Its original purpose was to provide an easy way to connect a CPU to peripheral chips in a TV-set.

I2C is an acronym for Inter-Integrated Circuit, which also explains its purpose: providing a means for ICs within a product to communicate with each other. It is a serial interface that transmits information between devices via two signal lines: SCL (Serial CLock), and SDA (Serial DAta)

3. Hardware of E-bus

The E-bus hardware physically consists of seven conductors or lines:

- Four for power supply, typically +5V and 0V
- Two for the bi-directional SDA and SCL signals
- One for the active-low Initial-Clear signal

Each device connected to the bus may function purely as a receiver, such as an LCD or LED driver, as a transmitter that provides information about key or panel-switch presses, or it may provide both functions. Regardless, every device has the ability to affect the logic level on both the SDA and SCL signal lines. The logic-level transitions are the result of serial-data "messages" being placed on the bus by the devices.

Incidentally, the correct term for this hardware structure is "buss", but Yamaha uses "bus" when referring to E-bus, so it will also be used in this document.

E-bus allows only Fast mode (Max. 400kbps), although the I2C standard allows three speeds: Standard mode, Fast mode, and High-Speed mode.

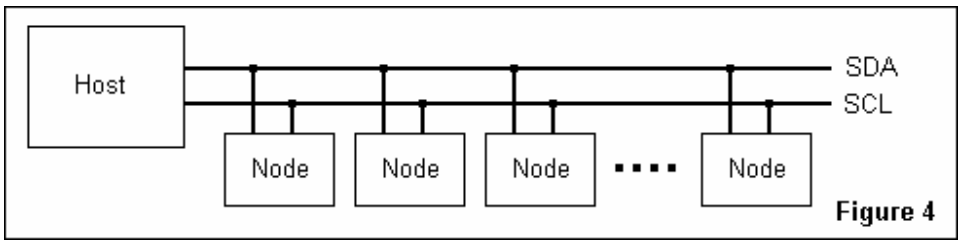

Figure 4 shows the basic structure of an E-bus system. It consists of a Host and one or more Nodes. The Host is either a microprocessor that acts as an interface between the E-bus and Main CPU systems, or it can be the Main CPU itself. The Nodes are peripheral devices that access the bus. If figure 4 represented a keyboard for example, then the Nodes could be individual CPUs that handle keyboard or panel-switch scanning on physically separate PCBs.

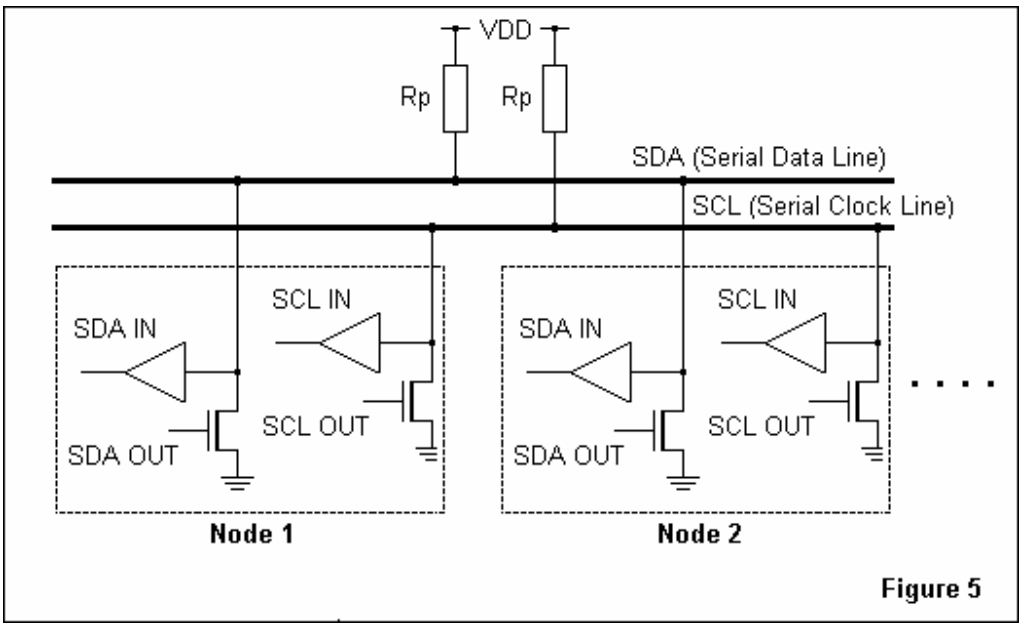

Figure 5 shows the circuitry within each device that “looks” into the E-bus. When turned on, an open-drain MOSFET or open-collector bipolar transistor drives its bus line to 0V (logic level 0). When off however, the line’s logic level rises to the supply voltage via the pullup resistor Rp connected to that line.

In addition to being able to drive the line, each device has an input buffer that reads the line’s logic level. Since all device output-drivers that are connected to a line are effectively in parallel, it’s possible that a “low” on a line may be due to one, two or several devices simultaneously driving the line. By reading the bus, each device knows whether or not the “highs” and “lows” that it placed on the line actually appear there. The bus line is always “high”, and is changed to “low” only when a driver is turned on.

4. Communication procedure

Any device (the Host or a Node) can initiate a message transmission, in which case that device is called a Master. Every other device is then a Slave for the purpose of that particular message, although as we will see, not *every* Slave will necessarily be involved any further with that message transmission.

Start and Stop conditions

The device wishing to communicate with another device must first place a Start condition on the bus. In doing so, it becomes a Master. As Figure 6 shows, the Start condition exists when the Master pulls the SDA line low while SCL is high. This condition is an indication to the Slaves that something is about to be transmitted. When a message is completed, the opposite occurs: the SDA line is released high while SCL is high.

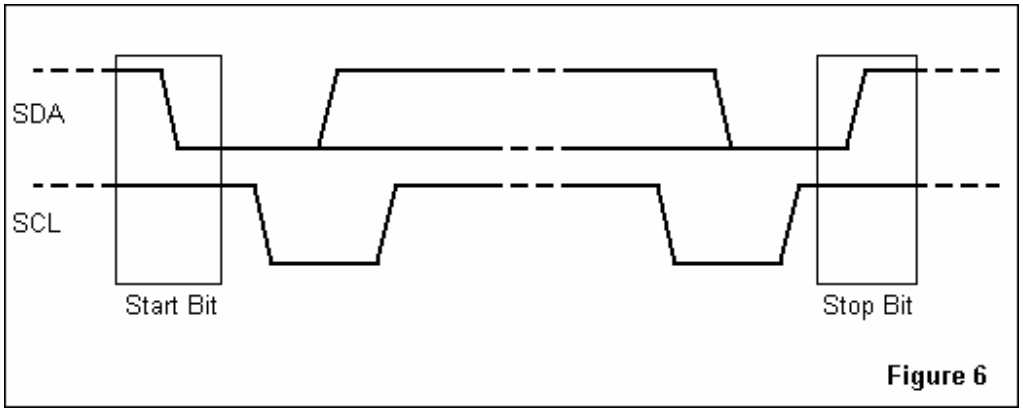

Figure 6

Bit Transmission

Figure 7 shows what happens on the bus when a data bit appears. The SDA line is allowed to change to the required state only when the SCL line is low. Conversely, when the SCL line is high, the logic level on the SDA line is considered to be valid.

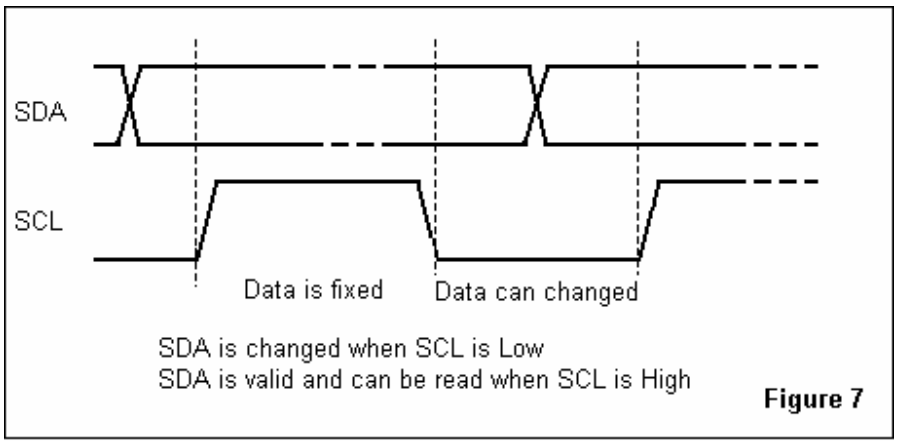

Figure 7

Addressing a Slave

After issuing the Start condition, the Master must then follow this by sending a byte containing the address of the Slave that it wants to communicate with. The relevant Slave, on seeing an address that matches its own, will acknowledge this by pulling the SDA line low after completion of the address byte. On seeing this, the Master knows that the Slave is actually out there, and communication can continue. The other Slaves ignore any data on the bus until a Stop message appears.

Since each node on an E-bus system generally contains an independently-running CPU, it's more than likely that two or more nodes may attempt to start communication simultaneously, since neither one "knows" of the other's existence.

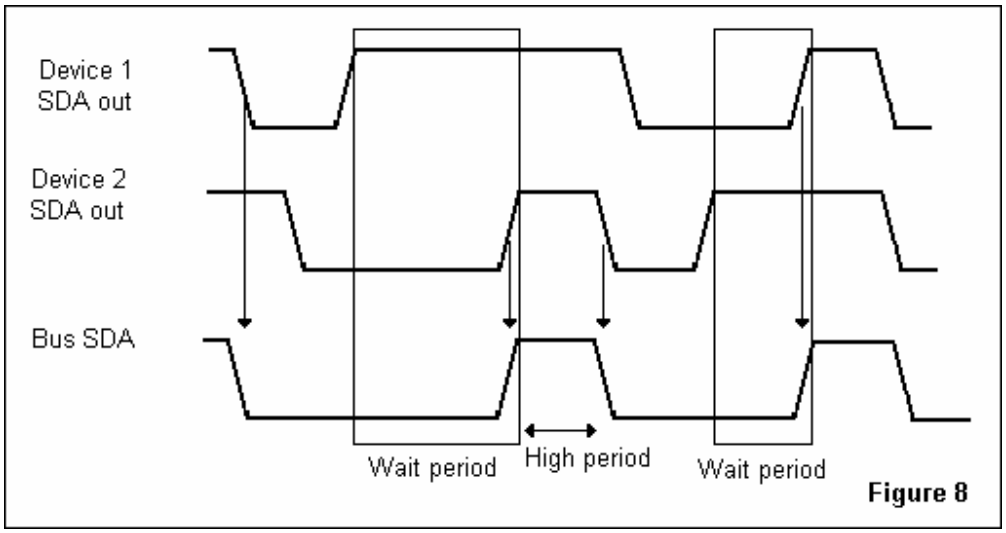

Figure 8

The device that outputs a low is always given priority because of the wired-OR bus arrangement. Figure 8 shows that a low bus level will always exist when either Device 1 OR Device 2 outputs a low. In order to prevent bus "collisions", each device must monitor the bus line before outputting a low. If a collision is detected, time-outs or wait-periods are initiated by the devices before the next bus drive attempt.

For example, in Figure 8 again, Device 1 and Device 2 have just issued their own Start conditions, shown here with one fractionally ahead of the other. On the bus, this appears as a normal Start. Each device then sends the first bit of an address that will identify the Slave it wants to communicate with. Device 1 wants to output a high, yet the bus is currently low because of Device 2. Since Device 1 sees a bus conflict, it waits (the Wait period) until the bus has returned to a high state for a certain amount of time (the High period). In doing so, Device 1 has "backed-off", allowing Device 2 to continue with the transmission.

5. Protocol

From a service and repair point of view - and since unlike MIDI, the E-bus is an internal system - there is little point in discussing in detail, the content of every possible message that may exist. However, an overview of the protocol may be helpful.

All E-bus messages are transmitted in "packets". A packet is all of the data that exists on the bus between the Start and Stop bits. Remember that once a Master device has initiated communication with its intended Slave by sending a Start bit and Slave address, no other device has any further interest in the bus data until a Stop bit is detected.

There are two types of packets: a 5-byte Standard Data packet, and a 17-byte Extension Data packet. The first two bytes of a packet are a destination address and a transmission address. The Slave device acknowledges its reception of each byte as it receives it. When the packet is completed, the Master will send the Stop bit. Figure 9 shows the contents of a message packet.

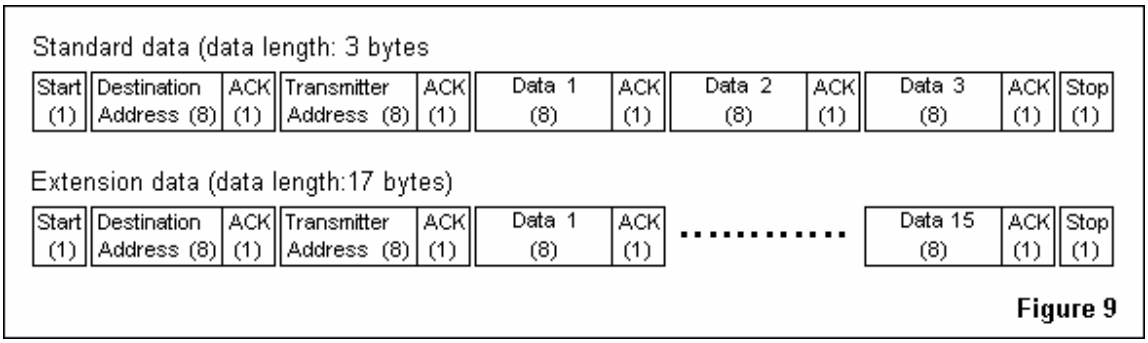

Figure 9

6. Circuit Implementation and fault diagnosis

Figure 10 shows an example of a typical circuit implementation. Depending on the product, there may be many "Node side" PCBs, each connected via its own loom to its respective E-bus connector on the DM PCB. Alternatively, the E-bus may be carried in and out of each Node PCB in a "daisy-chain" from a single DM PCB connector. Since all connectors have the same physical dimensions and pinouts, they are freely interchangeable.

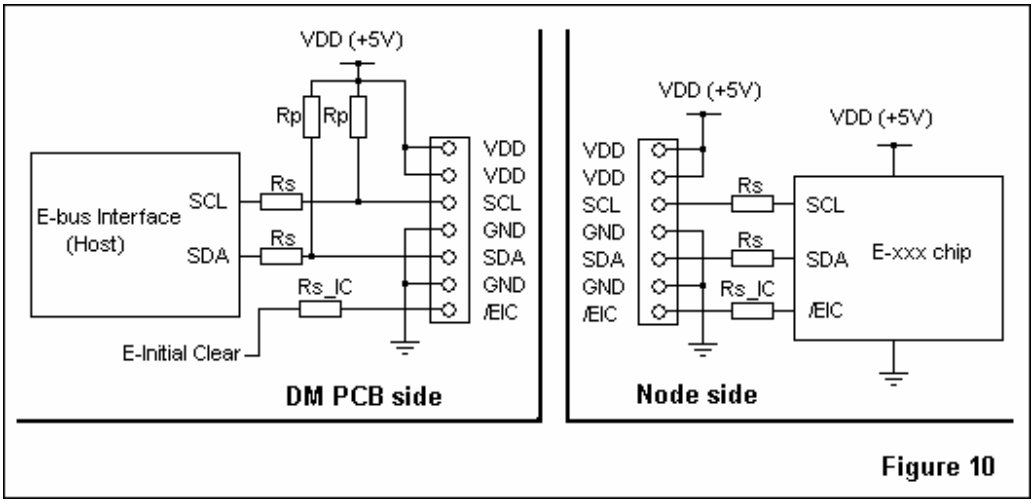

Figure 10

Resistors Rp pull up the SDA and SCL signal lines to +5V at the Host location only. Resistors Rs are placed between each device and the bus to protect against damage due to short-circuits to +5V. Shorts from either SDA or SCL to ground, while preventing communication, cannot cause damage since the device outputs are open-drains or open-collectors, and have no current-*sourcing* ability.

The node chips in any given system usually have the same part number, i.e. they are identical. To provide each chip with its own unique E-bus address, combinations of jumpers or resistors tie several port lines on each chip high or low.

Diagnosis of E-bus communication problems should present little difficulty, as the fault symptoms provide the clue. For example, a short-circuit between *any* line will disable all E-bus communication, typically resulting in the unit not progressing past its initial boot-up LCD display. Additionally, the unit won't respond to key or panel switch presses, nor will it illuminate any panel LEDs. In this case, switch off, remove all E-bus connectors at the DM PCB and re-apply

power. If the LCD subsequently progresses to a normal operating screen, then the fault is external to the DM PCB. If the short is located on a Node PCB, progressive reconnection of each Node PCB (after switch-off and switch-on, of course), and observation of the unit's subsequent operation, will eventually isolate the faulty section.

An open-circuit on any line between a node and the bus will disable only that node. If the E-bus connections are "daisy-chained" between Node PCBs, then those nodes that are "downstream" may also be affected. It's possible for example, to have the keyboard and all but one of the panel sections working. This immediately localises the problem to either the Panel PCB containing the E-bus node chip, or to a problem in the matrix of a panel section that is scanned by that particular chip. In the latter case, the problem is not so much a lack of E-bus communication but rather, the inability of the chip to detect the switch-press.

Finally, the E-bus Initial-Clear signal (/EIC), which resets all E-bus node devices, is usually generated by the Host device, and must be at a high logic-level to allow the nodes to operate. A short to ground will keep the nodes in reset, resulting in the same fault symptoms as those exhibited with open-circuit SDA or SCL lines. The voltage on an open-circuit /EIC line may float between a reset and operational logic state, possibly causing intermittent node operation.

Joseph Pantalleresco
Technical Support Co-ordinator
Professional and Music Products
Yamaha Music Australia

10th March 2006