

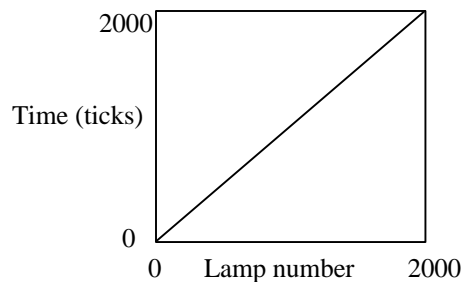
A Call for Algorithms

Calling all geeks, nerds, and mathheads,

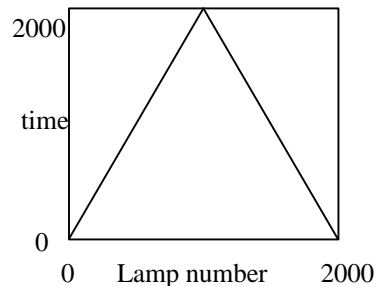
We of the L2K project are gathering methods for creating patterns for our ring of lights. Bring us your small, your elegant, your beautiful (did I say small) algorithms for repeating and evolving one dimensional patterns.

Our display device is a ring of 2000 lights, expressed on both a small and large scale. What can you do with this simple display? I already have clock wheels, colliders, chase sequences, and block flashers. What can you think of?

All high-speed patterns have a fixed period of 2000 clock states (ticks). With the number of lights fixed at 2000, this means that all repeating patterns can be expressed as a 2D grid with time as one axes and lamp number index as the other.



This linear graph would be a single dot of light circling once per second.



This graph would display as a pair of dots colliding at lamp 1000

From this example you can see that many patterns can be created by sets of linear equations, perhaps being driven by a table of values.

It would be very cool to get a graphic editor written that would allow a freehand bitmap to be entered and edited. This bitmap could then be processed into a set of pattern codes and sent to the ring.

All Algorithms will be expressed as 'C' code running on the small embedded computer called the Pattern Buffer. This is an industrial 18 Mhz Z180 with 512K Sram and 128K Flash. (It's rugged enough to survive the desert, so that's where the code goes).

The goal is to create sub-routines that will take parameters and create variations on different themes. Fixed-point math is preferred (Floating point is available, but very slow).

The ideal expression of this sub-routine will take seed parameters in one function and set up static variables for another function that will emit a progressive value each time it's called. Patterns that display shifting patterns can return a single bit per call. Burst patterns can fill an entire buffer at a time, as a 10 bit values in an Int.

I will take care of the details of connecting your algorithm into the PB parser and buffer feeder calls.

Here is the system hardware:

There is a master control/timing generator called the "Pattern Buffer"

- The PB controller has a RS232 comm port for remote control and a NTSC video output.

 - The comm port is fixed at 9600,N,8,1

- There is a control program written in 'C' with timing control provided by custom hardware.

 - The program has a real time event list manager

 - Events consist of text strings that are managed by a parser

 - Events may be loaded for certain date/times

 - Events may be grouped into macros

 - The event parser will create variable lists from text strings and call specified functions

There are two rings of 2000 lights, addressable as 10 bit words on 400 interface boards.

- Each interface board is addressable, Address zero being a broadcast (all call) mode

- There is an "indoor" loop from 1 to 200 and an outdoor loop, repeating addresses from 1 to 200

- All communications go through the indoor loop, then the outdoor loop.

- The most significant bit of the 10-bit word is rightmost on the display

 - I.E. The MSB of address 1 displays next to the LSB of address 2

- The indoor loop has buttons under each light, these buttons have different modes:

 - Flash the light when pressed, but have no effect on the current packet

 - Read a button down as a one bit into the current packet, clear the bit when released.

 - Toggle the display bits on button down and change the data in the current packet.

The communications rate and frame counter rate is 2000 command packets/events per second,

- Each packet is 25 bits, defined as:

 - 1 start bit,

 - 3 command bits,

 - 11 address/count bits,

 - 10 pattern/mode bits

- Each packet can do one thing,

 - Set the function mode bits inside the board (set addresses, reset counter, etc..)

 - Display a 10-bit pattern on the lamps

 - Store an 11-bit frame count and a 10-bit pattern in an event list

 - Store an ending mode for the event list (Blank, Repeat, Hold, or FastLoop)

 - Shift a single bit into the entire ring. (use the ring like a 2000 bit shift register)

Fast patterns (up to 2000 pps) can stored as one-second loops,

- Loops are loaded into the event lists of each board.

- Each board can store 32 events.

Each event is an 11-bit frame count value and a 10-bit lamp pattern.
When the count value matches the current frame count, the 10-bit pattern is displayed
 The display can be a direct write or an XOR into the existing pattern. (mode bit)
The last event may be used as an end event modifier
 The pattern can restart at each full 2000-frame count (default)
 The pattern can one-shot (hold at end)
 The end event can reset the frame counter, breaking sync with the ring.
 This allows short repeating patterns for PWM effects.

Display patterns longer than one second can be created by direct loading of the patterns
 Event timing for direct load patterns must be provided by the Pattern Buffer Computer.
Direct loads can be absolute loads of a 10 bit pattern or a one bit shift-in.
 All shift-in commands are logical left shifts across ascending addresses.
 This causes the display to appear as a clockwise rotation.
 One bit is shifted across all 2000 lamp bits per command.

Slow and fast patterns can overlap.
 Fast patterns can play from the event list while direct or shift loads are being done at any rate.

Because the packet communications is synchronous and each controller must repeat each packet, the Ring controllers are limited in the amount of processing that can be done on a single packet. After the overhead of shifting the packet in and out, the total packet processing cannot exceed 200 uS or the next packet will be dropped.

The PICs used in L2K have a 1uS instruction cycle time, so packet processing is limited to 200 instructions in any single packet time. Code speedups include arranging the packet command values so a binary bit test tree can do most of the command decoding. One of the more interesting speedups is an event defer function. If a display packet is received at a cycle count that also trips an event, the event is deferred until a non-decode packet is received. This keeps the total per-packet processing load below the limit value.

Other limits allow only a single event to be deferred, this has a small side effect. If a block of direct display command, (streaming to a single address) blocks the event function until the cycle clock exceeds the trip point of the next event, the event list execution will hang for a full second. (Until the cycle clock rolls around to the missed event.) It is easy to avoid this by inserting some NOP commands into any dense direct write block. The percentage of NOPs depend on the dead time in a loaded event list. (A sparse list needs fewer NOPS).

A deferred event can overwrite a direct display command and this can have unintended results. Before starting a complex direct write sequence, you can suspend the event list for all boards by setting the mute bit in a broadcast command (code 0x000049).

Here is an example of a pattern I call "BOIL" This pattern displays a variable percentage of random flashing between a start and an end value. (The flashing is more frequent near the center of the area) This code is inside a timer driven task that fires at a predefined "tick" rate. It looks best at 15 to 35 ticks per second.

```

Cmd = 0;           //Cmd is the 10 lamp pattern
Addr = start / 10; //Addr is the board address
Lamp = start % 10; //Lamp is the number of the current lamp
Hwidth = (end - start) / 2; //one half the width of the arc

for(i = start; i < end; i++){
    //percent is a binary value from 0 to 255
    percent = 256 - (abs(center - i) << 8) / Hwidth;
    //get a random number and see if the percent is higher
    if(percent > (rand() & 0x1fff)){
        //set the current bit to a one
        Cmd |= (0x01 << CLamp);
    }
    //else the current bit is a zero, do nothing
    ++Lamp;           // move on to the next lamp
    if(Lamp > 10){   //if we have passed the end of this card,
        Lamp = 0; //reset the lamp pointer
        IPatLoader(DisplayNOW, Addr, Cmd); //send the pattern
        ++Addr;    //change to the next address
        Cmd = 0;  //start the lamps over at zero
    }
}

```

This is the 24 bit Command Word Format used between the PB controller and the ring boards

```

|byte 2||byte 1||byte 0|
765432107654321076543210
.....| | | | | | | | | | - Lamp pattern, one bit per lamp
...| | | | | | | | | | - Count value, 11 bits frame count or options (2047)
| | | - Command Code 3 command bits

```

000	Extended modes	address	Load, Snoop, nMute, Aset, nNop
001	Readback	address	returned data
010	shift display	address	new bit in LSB
011	Display now	address	lamp pattern

address = 0 is broadcast (all call)

100	Load first pattern	trip count	lamp pattern
101	Load next pattern	trip count	lamp pattern
110	Load end state event	trip count	hold, blank, restart
111	UNUSED		

Examples:

```

0x600000 = broadcast all lamps dark
0x600401 = turn on lamp one on board one
0x400400 = starting at board one, add a zero and shift all bits clockwise

```