The H-ITT Transmitter Signal

- When you click the H-ITT transmitter button:
  - A-E: LED indicator comes on, and at same time, TWO bursts of infrared light come out: LED stays on even after transmission stops, until button is released
  - * button: on release of button, LED flashes and two infrared bursts are sent

- bursts last 53 milliseconds, are 9 ms apart, and have a bit-period of about 0.5 ms (about 2000 bits per second)

- Let’s look at it on scope...

H-ITT Transmitter Protocol

Transmitter 55573 sends an “A”

Transmitter 55573 sends a “B”

Comparison of A & B first packets

Differences are minor, showing up only near beginning & end

We will represent “high” states (light on) as 1’s, and lows (off) as 0’s

Notice standard widths: choices are single- or double-width (both for the zeros and the ones)
Decoding the A signal

Sequence starts out: 01101001001101001001001001...
Notice the 01 delimiters: 01101001001101001001001001...
This gives the signal its choppy appearance (never see 3 1's or 0's in a row)
Actual data appears between delimiters: 1's look fat, 0's look skinny

Resulting bit-sequence for A signal (both packets) is:

```
1 0 0 1 0 0 0 0 0 0 1 1 0 1 0 0 1 0 0 1 0 0 1 0 0 1
1 0 0 1 0 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 0 1 1 1
```

The different buttons: first four bits

- **A**: 1001 → 001 → 1
  - first bit always 1
- **B**: 1010 → 010 → 2
- **C**: 1011 → 011 → 3
- **D**: 1100 → 100 → 4
- **E**: 1101 → 101 → 5

<<

```
1110 → 110 → 6
```

What's with the Checksum?

Break data into chunks of 8 bits (bytes) and add up:

```
1001
00000000
11011001
00010101
11101111
```

Checksums provide a "sanity check" on the data integrity

The Transmitter ID bytes

- Transmitter number is binary-coded in the usual sense:
  - 32768 + 16384 + 4096 + 2048 + 256 + 16 + 4 + 1 = 55573
  - this exactly the number pasted behind the battery
- Second packet inverts all the bits to ensure data integrity
A look at the H-ITT Serial Datastream

E-button on H-ITT (first of two packets):

- Serial datastream looks a lot different
  - this one allows many zeros or ones in a row
  - delimiters (called start bit and stop bit) bracket 8-bit data (1 byte)
  - in this case, 0's are positive voltage, 1's are negative (backwards)
  - happens much faster than IR: in this case 19,200 bits/sec (baud)

Packet breakdown:
- first packet: button number (5 -> E), with LSB first: 101000
- next three packets are ID, also LSB first within each
- last packet is checksum type of verification

Stereo Remote Control

- Similar to H-ITT transmitters in principle:
  - bursts of infrared light carrying digital information
  - punctuated by delimiters so no long sequences of 1’s or 0’s

- Key differences:
  - signal initiated by a WAKE UP! constant-on burst
  - pattern that follows is repeated indefinitely until button is released
    - I can never get fewer than three packets...
  - packet is variable in length depending on button

Sample patterns for data packet

<table>
<thead>
<tr>
<th>POWER</th>
<th>VOL +</th>
<th>VOL -</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000000</td>
<td>10000000</td>
<td>01000000</td>
<td>100000</td>
<td>010000</td>
<td>11000100</td>
<td>00100100</td>
<td>101001000</td>
<td>011001000</td>
<td>111001000</td>
</tr>
</tbody>
</table>

A Different Code...

- The radio remote uses a different scheme:
  - does not use the 01 delimiters like H-ITT did
  - instead, uses 10 to represent zero, and 1000 to represent 1
  - sequence for the 5 button is:
    - 100010001000100010001000100010001000100010...

  ID part
  
  data part

  - in data part, least significant bit (LSB) is first
  - so the number part of “5” is 101001000 → 1010
  - least significant digit is first, so reverse order for more familiar form: 0101 = 5
Magstripe Idea

- On magnetic stripe, N-S junctions eat their own magnetic flux lines, but N-N or S-S present external flux lines of opposite direction
  - pattern of N-N and S-S creates + and - transitions
  - zero represented by long period
  - one represented by short period
  - zeros look fat; ones thin (sign irrelevant)
- two streams are produced from this:
  - a data stream
  - a clock
- data valid when clock high

Magstripe Geometry

- There are up to three tracks of data
  - Tracks 1 and 3 typically higher-density (7-bit) alpha-numeric data
  - Track 2 typically lower-density (5-bit) numeric data
  - Track 2 used on almost every card; track 1 often, track 3 seldom

Track 2 Character Code

<table>
<thead>
<tr>
<th>Data Bits</th>
<th>Parity</th>
<th>Character</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 1</td>
<td>0</td>
<td>0 (0H)</td>
<td>Data</td>
</tr>
<tr>
<td>1 0 0 0 0</td>
<td>1</td>
<td>1 (1H)</td>
<td>&quot;</td>
</tr>
<tr>
<td>0 1 0 0 0</td>
<td>2</td>
<td>2 (2H)</td>
<td>&quot;</td>
</tr>
<tr>
<td>1 1 0 0 1</td>
<td>3</td>
<td>3 (3H)</td>
<td>&quot;</td>
</tr>
<tr>
<td>0 0 1 0 0</td>
<td>4</td>
<td>4 (4H)</td>
<td>&quot;</td>
</tr>
<tr>
<td>1 0 1 0 1</td>
<td>5</td>
<td>5 (5H)</td>
<td>&quot;</td>
</tr>
<tr>
<td>0 1 1 0 1</td>
<td>6</td>
<td>6 (6H)</td>
<td>&quot;</td>
</tr>
<tr>
<td>1 1 1 0 0</td>
<td>7</td>
<td>7 (7H)</td>
<td>&quot;</td>
</tr>
<tr>
<td>0 0 0 1 0</td>
<td>8</td>
<td>8 (8H)</td>
<td>&quot;</td>
</tr>
<tr>
<td>1 0 0 1 1</td>
<td>9</td>
<td>9 (9H)</td>
<td>&quot;</td>
</tr>
<tr>
<td>0 1 0 1 1</td>
<td>1</td>
<td>C (AH)</td>
<td>Control</td>
</tr>
<tr>
<td>1 1 0 1 0</td>
<td>2</td>
<td>D (BH)</td>
<td>Start Sentinel</td>
</tr>
<tr>
<td>0 0 1 1 1</td>
<td>3</td>
<td>E (CH)</td>
<td>Control</td>
</tr>
<tr>
<td>1 0 1 1 0</td>
<td>4</td>
<td>F (DH)</td>
<td>Field Separator</td>
</tr>
<tr>
<td>0 1 1 1 0</td>
<td>5</td>
<td>G (EH)</td>
<td>Control</td>
</tr>
<tr>
<td>1 1 1 1 1</td>
<td>6</td>
<td>H (FH)</td>
<td>End Sentinel</td>
</tr>
</tbody>
</table>

Mag-stripe Circuit Schematic
Unfamiliar bits

- You've seen many of the elements in the circuit by now
  - AND, NOR, NOT logic
- New pieces are down-counter, flip-flops (and latch version thereof)
- Also have a PIC microcontroller and an RS-232 level converter

The D-type flip-flop

- Rules:
  - whatever logic level sits at D input, it gets transferred to Q
    (and inverse logic to Q-bar) when the clock sees a positive edge
  - if R (reset) goes low, Q goes to zero (Q-bar to logic high)
  - if S (set) goes low, Q goes to logic high (Q-bar to zero)
  - holds Q, Q-bar values as long as S, R, high, no matter what
    happens to D input, provided no clock edge

74HC74: Dual D-type flop

<table>
<thead>
<tr>
<th>PIN NO.</th>
<th>SYMBOL</th>
<th>NAME AND FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 13</td>
<td>T_{RD}, R_{D}</td>
<td>asynchronous reset-direct input (active LOW)</td>
</tr>
<tr>
<td>2, 12</td>
<td>T_{D}, D/O</td>
<td>data inputs</td>
</tr>
<tr>
<td>3, 11</td>
<td>C \text{ or } Q</td>
<td>clock input (LOW to HIGH, edge-triggered)</td>
</tr>
<tr>
<td>4, 10</td>
<td>T_{SP}, Q_{D}</td>
<td>asynchronous set-direct input (active LOW)</td>
</tr>
<tr>
<td>5, 9</td>
<td>T_{Q}, Q/D</td>
<td>true flip-flop outputs</td>
</tr>
<tr>
<td>6, 8</td>
<td>Q_{D}</td>
<td>complement flip-flop outputs</td>
</tr>
<tr>
<td>7</td>
<td>GND</td>
<td>ground (0 V)</td>
</tr>
<tr>
<td>14</td>
<td>V_{CC}</td>
<td>positive supply voltage</td>
</tr>
</tbody>
</table>

74HC574 Octal D-flop

- Just 8 D-type flops with same clock, no set/reset
- Often called a "latch" preserves data input at clock
### 74HC193 Counter

<table>
<thead>
<tr>
<th>PIN NO.</th>
<th>SYMBOL</th>
<th>NAME AND FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 2, 7</td>
<td>G0 to G7</td>
<td>No-float outputs</td>
</tr>
<tr>
<td>4</td>
<td>CP0</td>
<td>Count down clock input</td>
</tr>
<tr>
<td>5</td>
<td>CP1</td>
<td>Count up clock input</td>
</tr>
<tr>
<td>8</td>
<td>GND</td>
<td>Ground (GND)</td>
</tr>
<tr>
<td>11</td>
<td>PC</td>
<td>Asynchronous parallel load input (active LOW)</td>
</tr>
<tr>
<td>12</td>
<td>D(CR)</td>
<td>Terminal count or carry output (active LOW)</td>
</tr>
<tr>
<td>13</td>
<td>IEN</td>
<td>Asynchronous master reset input (active HIGH)</td>
</tr>
<tr>
<td>14, 1-10, 9</td>
<td>D0 to D9</td>
<td>Data inputs</td>
</tr>
<tr>
<td>16</td>
<td>VCC</td>
<td>Positive supply voltage</td>
</tr>
</tbody>
</table>

- **Load input; count up or down; alert when at end**

### MAX232 RS-232 Level-shifter

- **Powered by 5V**
- **Uses charge-pump capacitors to generate ±12V for RS-232 levels**
- **Provides 2 channels of communication**
  - Transmit and receive for each

### Reading

- For magnetic stripe stuff, see:
  - [http://en.wikipedia.org/wiki/Magnetic_stripe_card](http://en.wikipedia.org/wiki/Magnetic_stripe_card)
  - [http://money.howstuffworks.com/question503.htm](http://money.howstuffworks.com/question503.htm)
  - [http://stripesnoop.sourceforge.net/faq.html](http://stripesnoop.sourceforge.net/faq.html)
  - [http://stripesnoop.sourceforge.net/devel/stripesnoop37.txt](http://stripesnoop.sourceforge.net/devel/stripesnoop37.txt)
- See data sheets linked via the Lab 9 Parts List