Cassette input port

Unlike some microcomputer systems available at present which allow the transfer of data to and from a cassette recorder, the Apple II and ITT 2020 depend virtually entirely on software for analysing the output from the cassette when reading data, and generating a suitable signal for it when writing data. This means that the data is available to the processor in virtually the same form as it is recorded on the tape, with only a Schmitt input between it and a "clean" output buffer, so that it is not buffered off from it by a P.I.I. or frequency comparator. This means that one can perform various functions on the signal fed into the data input (which of course, doesn't necessarily have to come from a cassette recorder. This is, however, probably the best way to start as it will already be hooked up).

Period measurement

By deriving a means to measure the frequency or period at this point, the value obtained can be used by other programs for control or effect...or anything else.

In this case the result determines the colour of a pattern displayed on the screen, and any number of experiments to be performed; while the result of other applications of the technique are only limited by the user's imagination.

Whistling is chosen as the best input (when using a microphone) as it gives a purer waveform than, for instance, voice input. Hence results are more reliable (see later, however).

Software

The technique produces a number proportional to the period of the incoming signal by starting a counter at a 1 to 0 transition and stopping it as soon as a 1 to 1 transition is encountered.

One sample of the period measurement would probably do on its own if the signal to be measured were clean (a pure sine or squarewave, for example), but there are bound to be various discontinuities or extraneous noises involved too and these will almost inevitably cause errors.

To get round these problems not just one but sixteen samples are taken and the results stored in memory.

Averaging

In the next stage all the sixteen samples are added together, the result being stored in memory (two bytes). This value is then divided by sixteen to yield the average value of all the samples. This has a far greater chance of being representative of the actual period than the value obtained from a single measurement.

Initially this part of the program was done in BASIC with only the period measurements in machine code; but the BASIC section proved too slow for most applications and so the averaging is now included in the machine code section.

The sixteen samples are stored in memory locations 0300-030F. Locations 0310-0325 contain the sub-routine which actually carries out the period measurement, whilst locations 0350-037F is the averaging program. The control program (0330-0348) calls

WHISTLE UPA COLOUR ON YOUR APPLE II/ITT 2020

Using only the cassette data input port of the Apple II (plus a microphone/pre-amp) the software described here provides a technique to generate effects or alter program direction as a result of the user whistling at different frequencies. R. W. Lawrence BSc explains... up the above two subroutines, and also determines the number of samples to be taken and where they are stored.

Locations 0326 and 0329 hold the low and high order bytes of the sum of the samples, and the final average value is stored in location 038F. This can be examined when in BASIC by PEEKing (911).

Source code

The programs are fairly self-explanatory. The division by sixteen is performed by a right shift operation done four times, from location 0329 through the accumulator. Since none of the period samples taken could be greater than 225; it follows that the average must be less than this too — hence the result must fit into the accumulator.

Basic program

This program puts the colour display up on the screen, and after calling up the machine code program, varies the colour of the display depending on the

<table>
<thead>
<tr>
<th>Listing</th>
<th>Storage locations</th>
</tr>
</thead>
<tbody>
<tr>
<td>0300</td>
<td>for period measurements</td>
</tr>
<tr>
<td>01</td>
<td>Initialise flags and counter</td>
</tr>
<tr>
<td>02</td>
<td>Zero yet? No, test again</td>
</tr>
<tr>
<td>03</td>
<td>Yes: Increment X</td>
</tr>
<tr>
<td>04</td>
<td>One yet? No? Go back and inc. X</td>
</tr>
<tr>
<td>1</td>
<td>Yes: Go on</td>
</tr>
<tr>
<td>OF</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
</tr>
<tr>
<td>0310</td>
<td>Transfer X to A and store final count in relevant location. Return.</td>
</tr>
<tr>
<td>A9 01</td>
<td>Set up pointer</td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
</tr>
<tr>
<td>1A</td>
<td></td>
</tr>
<tr>
<td>2C 60 C0</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td></td>
</tr>
<tr>
<td>0330</td>
<td></td>
</tr>
<tr>
<td>A9 00</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td></td>
</tr>
</tbody>
</table>
frequency of the incoming signal.
The system assumes the signal has a frequency range from about 400Hz to
about 2KHz, this being considered a reasonable range to whistle over. Any
measurement yielding values that are outside this preset range are “clipped”
into it by lines 220 and 230. Line 240 normalises the measurement into an
integer between 1 and 15: ready to set
the colour of the pattern to one of the
fifteen possible on the system.
Line 242 makes the display slightly
more interesting by randomly selecting
the “tunnel in” (line 245), or the
“tunnel out” (line 243) subroutine. These
two subroutines change the colour
of about ten concentric squares, the
only difference between them being the
STEP value which selects motion inwards
if $-2$, and outwards is $+2$.

**Looping**

As it stands, the processor remains in a
loop in the machine code program
awaiting any input to the cassette input
port. Whilst in this loop it cannot accept
data entered from the keyboard – or
anywhere else – only, in fact, from the
cassette input. Control can be returned
to the keyboard via the usual RESET:
control C. If desired this can be remedied
by putting in a “look at keyboard to see
if the key has been depressed” type of
programme between lines 0314 and
0317, and jumping out of the loop if it
has. This was omitted to keep the num-
ber of machine cycles within the loop to
a minimum and thus allow the highest
possible frequencies to be measured.

**Other applications**
The technique has been used success-
fully in other applications (the one
described here was really by way of an
example) such as in a software fre-
cency of the incoming signal

A reciprocal machine code program
could be added. This would form a
generator (using the internal speaker)
producing tones of the same frequency
as those measured by the above arrange-
ment when presented with the same
data. Thus the Apple could “whistle”
along with you – albeit rather roughly
as it would have to stop every so often
to see what you’re whistling.

A suitable machine code generator
program is also given. It’s a modified
form of the one listed in the Apple II
Reference manual, except that the dura-
tion is preset (byte in location 0391)
and also a “tuning” factor is added to
make sure it “whistles” what it “hears”
(using the terms euphemistically). Lines
03A2 – 03A9 perform the latter. In-
crease the value at 03A4 to flatten the
generated frequency for a given input,
and reduce it to sharpen the generated
frequency.

**Linking up**

The generator program can be linked to
the analyser program as follows: Line
0348 changes from 60 (RTS) to: 0348 4C 90 09 (JMP 0390).

P.S: If you can’t whistle, you’ll just
have to learn to hum sinusoidally.