Reading Binary

Student Activity

7 8 **9** **10** 11 12

TI-Nspire Innovator Investigation Student 90 min

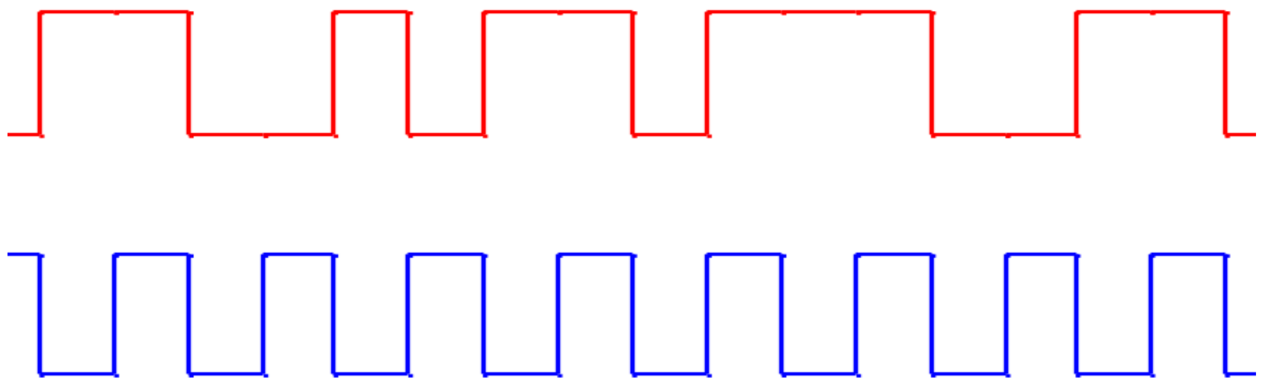
Synchronising Data

A binary signal such as: 1101 is easy to read as numbers but when this is sent as light or electrical signals how does the electrical circuit detect whether two 1's or just a single '1' were received at the start of the signal? The simple answer: "knowing the duration of a single bit". In practical terms this is not as easy as it sounds. A range of protocols are used to streamline digital communications.

In this activity a clock pulse will be sent at the 'same time' as the data[1]. In the case of the TI-Innovator, two LEDs will be used; one LED will be used as a clock pulse advising 'when' to read each bit; the second LED will be used to transmit the binary data. As the data will be read by 'humans' the transmission will be made slow enough to record the data 'by-hand', the speed can be changed as your skill level increases.

The sample data transmission below shows a clock pulse (bottom) and data (top). Look at the data signal, what data do you think is trying to be sent? This diagram illustrates some of the complexities pertaining to data transmission. If the data is read as the clock pulse goes 'high' (leading edge), the first data point to be read would be a '1'. The second time the clock pulse goes 'high' the data is low so a '0' would be read. What state is the data in when the third clock pulse goes 'high'? This example illustrates the importance of timing.

Now consider data transmission on the trailing edge of the clock pulse? Would the same signal be sent? Would either method result in the intended data transmission? In both these situations square wave forms have been used, real situations involve ramp up / ramp down times which mean the leading and trailing edge are not straight up and down making timing even more critical. With CPU clocks typically running at 1.8GHz the sequence below would take approximately 0.000 000 001 seconds to transmit.



These examples illustrate a very important point when considering the order in which information will be sent from the calculator to the TI-Innovator. Programs read statements sequentially, so it is important that the data be sent to the TI-Innovator before sending the signal to 'read'.

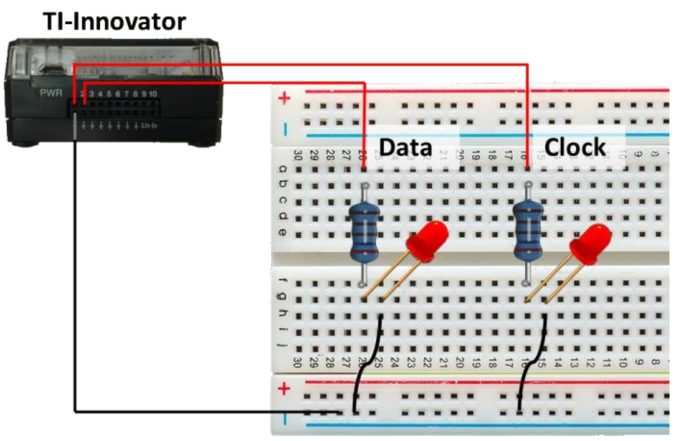
[1] This form of data transmission is more resource heavy than more complex asynchronous communications.

Author: P. Fox

Constructing the Circuit

In this activity two LED's will be connected to the TI-Innovator's power output ports. One LED will be used to display the data; the other will be used as a clock-pulse, a signal to the receiver to read the status of the data LED.

Construct the circuit shown below. Note that the LED connected to power output 1 (PWR 1) will be used as the clock. The LED connected to power output 2 (PWR 2) will be used to transmit data.

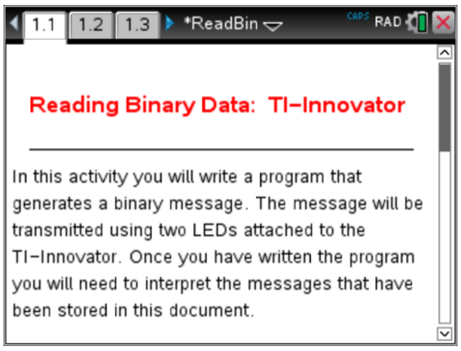


Coding the Algorithm

Connect the TI-Innovator hub to your calculator using the Calculator to Calculator cable. The "B" end connects to the Innovator and the "A" end to the calculator.

Open the file called: ReadBinary

The file includes the start of a program that needs to be completed and lots of 'data' that will be used in this activity.



Reading Binary Data: TI–Innovator

In this activity you will write a program that generates a binary message. The message will be transmitted using two LEDs attached to the TI–Innovator. Once you have written the program you will need to interpret the messages that have been stored in this document.

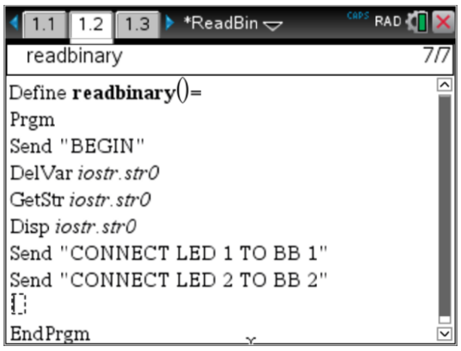
The two LED's need to be set up on TI-Innovator. From the menu select: **Hub** > **Send "Connect – Output** > **LED**

Label the first LED '1' and attach it "**TO**" breadboard location 1:

Hub > **Ports** > **BB 1**

The entire command should: Send "CONNECT LED 1 TO BB 1"

Repeat the above process to set up LED 2 on BB 2.



```
Define readbinary()=
Prgm
Send "BEGIN"
DelVar iostr.str0
GetStr iostr.str0
Disp iostr.str0
Send "CONNECT LED 1 TO BB 1"
Send "CONNECT LED 2 TO BB 2"

EndPrgm
```

Notes:

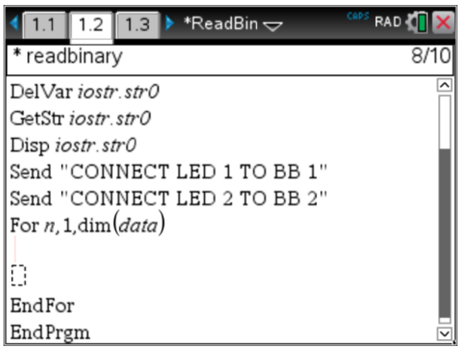
- This program will read binary data stored in a list called "Data". The length of the list will be determined automatically by the program, however the list must only contain 1's and 0's.
- Wait times will be set to 2 seconds. With practice you should be able to read the transmitted information much quicker, too fast however will may lead to errors.

Author: P.Fox

A loop needs to be created to continually and sequentially send the data. The number of times the loop is repeated depends on the amount of data. (Number of bits)

Create a FOR loop that starts at 1 and ends at **DIM**(Data) and uses *n* as the automatic loop counter.

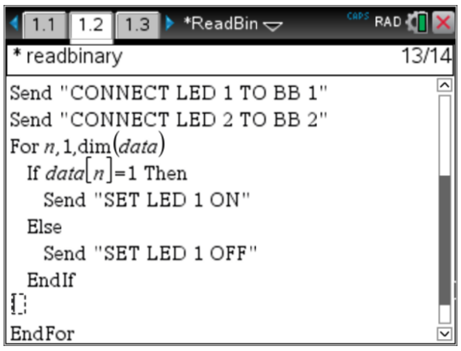
The dimension command can be found in the catalogue or typed directly.



```
* readbinary                    8/10
DelVar iostr.str0
GetStr iostr.str0
Disp iostr.str0
Send "CONNECT LED 1 TO BB 1"
Send "CONNECT LED 2 TO BB 2"
For n,1,dim(data)

EndFor
EndPrgm
```

The next step is to check if the data is a '1' or a '0'. If it is a '1' then LED 1 needs to be switched 'ON', otherwise it must be a zero and LED 1 is switched 'OFF'. The screen opposite shows the set of commands to achieve this result.

The "**If … Then … Else … EndIf**" command can be found in the Control menu.

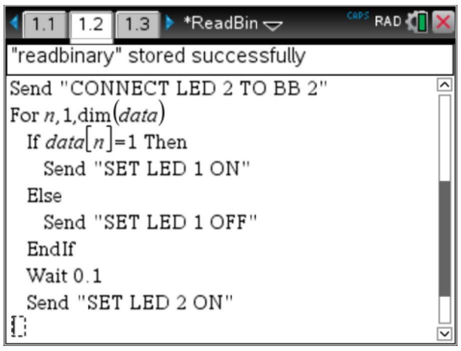


```
* readbinary                    13/14
Send "CONNECT LED 1 TO BB 1"
Send "CONNECT LED 2 TO BB 2"
For n,1,dim(data)
  If data[n]=1 Then
    Send "SET LED 1 ON"
  Else
    Send "SET LED 1 OFF"
  EndIf

EndFor
```

At this point the program is slowed down slightly simulating a small 'ramp up' time. Insert a **WAIT** command immediately after the **EndIf** and have the program wait 0.1 seconds before sending the command to switch the clock pulse on. (LED 2)

Set the clock pulse (LED 2) to come on, shown opposite.

Switching on LED 2 represents the 'leading edge' of the clock pulse. When the program is running, this is the prompt to read the status of LED 1 and record a '1' or a '0'.



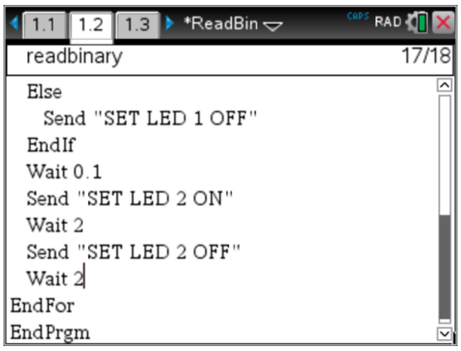
```
"readbinary" stored successfully
Send "CONNECT LED 2 TO BB 2"
For n,1,dim(data)
  If data[n]=1 Then
    Send "SET LED 1 ON"
  Else
    Send "SET LED 1 OFF"
  EndIf
  Wait 0.1
  Send "SET LED 2 ON"

```

Another **WAIT** command is required to allow time to record the status of LED 1, remember, in this activity the 'reading' is being done by humans!

Insert a **WAIT** time of **2** seconds. (This can be changed later.)

Immediately after the **WAIT** command, switch the clock pulse (LED 2) **OFF**, followed by another WAIT of 2 seconds representing the cyclic ON / OFF of the clock pulse.

Make sure the "EndFor" command is in place and the program is finished.

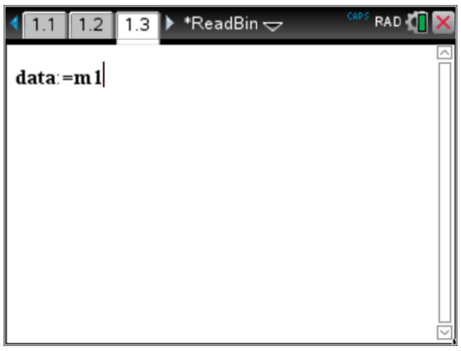


```
readbinary                      17/18
  Else
    Send "SET LED 1 OFF"
  EndIf
  Wait 0.1
  Send "SET LED 2 ON"
  Wait 2
  Send "SET LED 2 OFF"
  Wait 2
EndFor
EndPrgm
```

Navigate to page 1.3. Data has already been stored in this file. To shift the data into the data list type:

data:=m1

Use the VAR key to access the **readbinary** program and run it. Watch how LED 2 switches ON / OFF regularly and LED 1 switches ON / OFF according to the data.



```
data:=m1
```

For each of the following questions, work with a friend or two to read, record and translate the messages stored on the calculator.

Question: 1.

Use a stop watch to record how long it takes your team to record, convert and read the binary message. (the entire process below)

- Record binary output from the first message (M1),
- Convert the message binary into decimal
- Convert the decimal into text using ASCII table and hence write out the text message.

Question: 2.

Edit the program and change the wait times from 2 seconds to 1 second for the on and off signals immediately after each clock pulse for LED 2. (Clock) The message will be sent twice as fast. Once again, use a stop watch to record how long it takes your team to record, convert and read the binary message. Before you start, put second message into the data list: data:=m2 .

Question: 3.

The messages decoded in Question 1 and 2 are different lengths. Determine the best way to compare the conversion speed for both messages and compare the two conversion times.

Question: 4.

Adjust the wait times in the program once again. Store the third message M3 into the data list and start the conversion timing the total conversion time. Record all results as before and the wait time used to convert the message.

Question: 5.

There is a mistake in message 4. Determine the message and corresponding mistake.
You may also like to practice your timing.

Question: 6.

Would a parity bit check have identified the error?

ASCII Table – (Letters only)

A = 65	B = 66	C = 67	D = 68	E = 69	F = 70	G = 71	H = 72	I = 73	J = 74
K = 75	L = 76	M = 77	N = 78	O = 79	P = 80	Q = 81	R = 82	S = 83	T = 84
U = 85	V = 86	W = 87	X = 88	Y = 89	Z = 90				
a = 97	b = 98	c = 99	d =100	e = 101	f = 102	g = 103	h = 104	i = 105	j = 106
k = 107	l = 108	m = 109	n =110	o = 111	p = 112	q = 113	r = 114	s = 115	t = 116
u = 117	v = 118	w = 119	x = 120	y = 121	z = 122				

ASCII Table – (Some Special Characters)

Space = 32	Exclamation = 33	Double Quotes = 34	Dollar Sign = 36	Ampersand = 38
Single Quote = 39	Comma = 44	Hyphen = 45	Period = 46	Forward Slash = 47

Author: P.Fox