

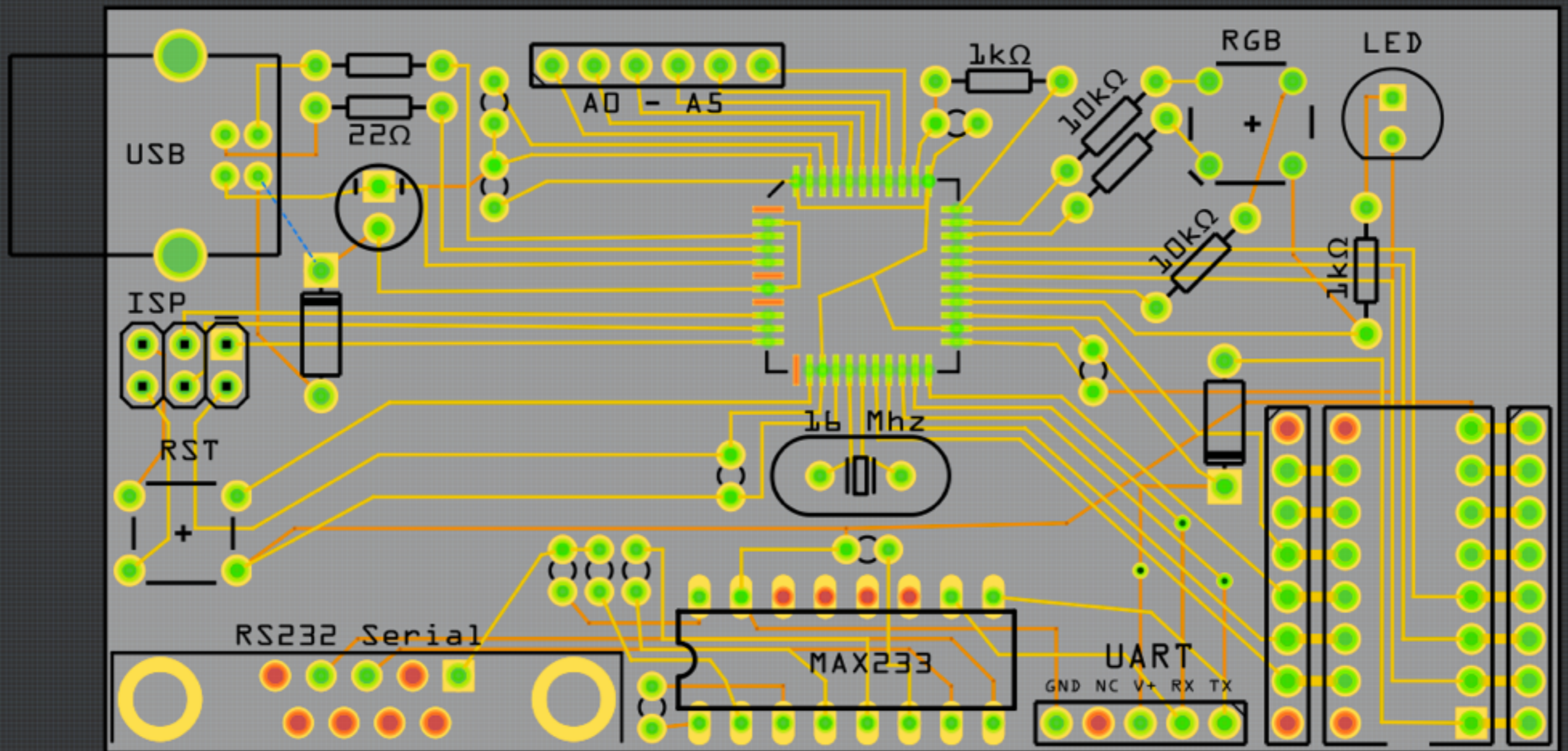
GPIO

GP GPIO

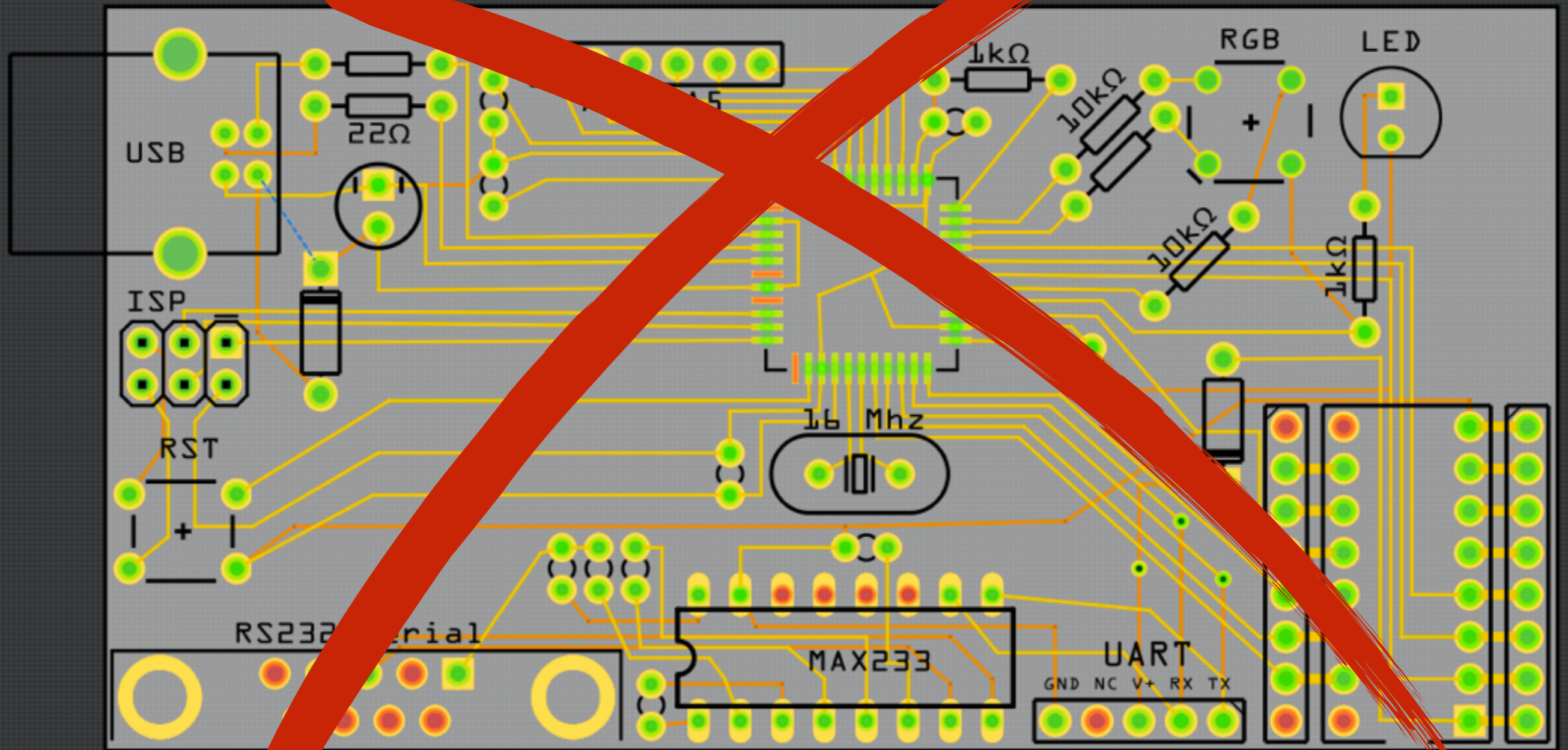
(GP)²IO

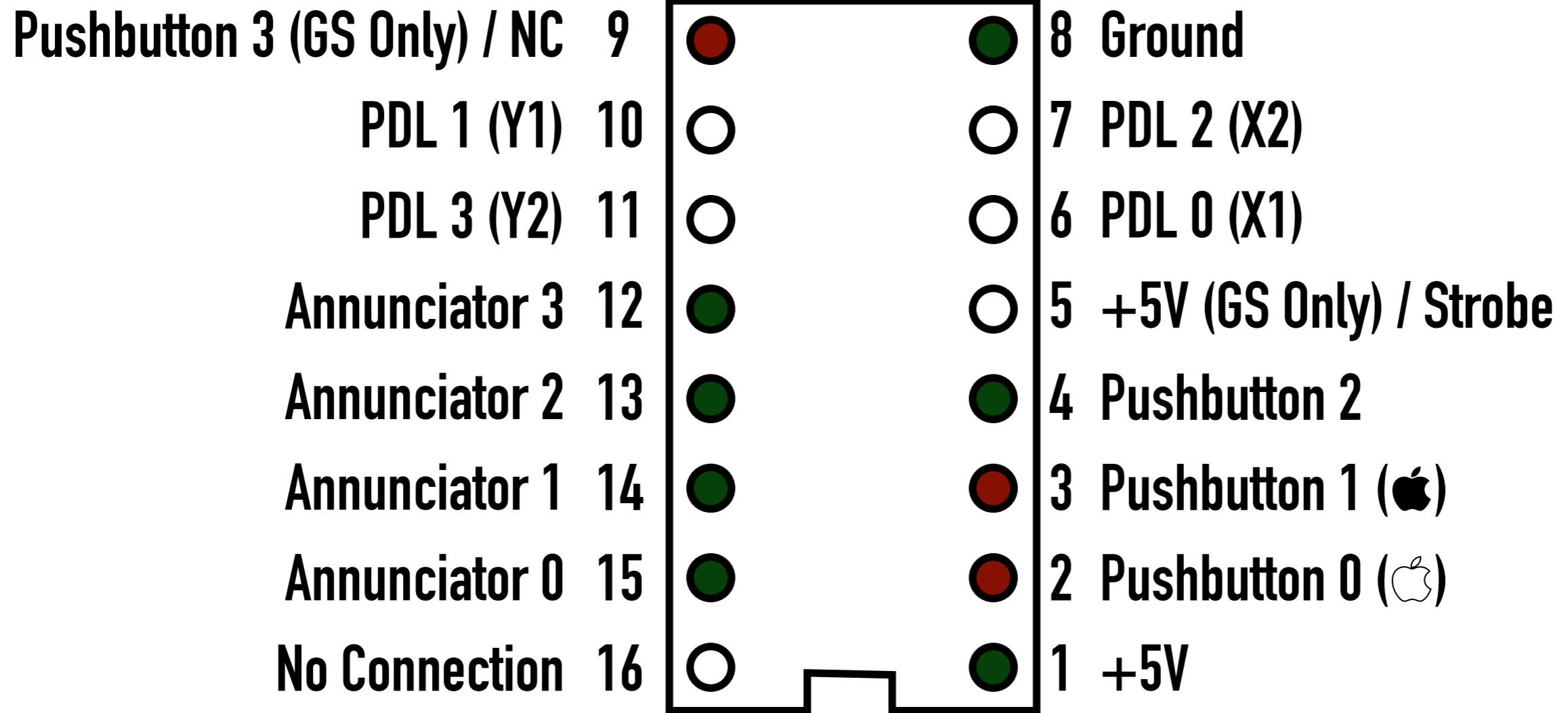
General Purpose Game Port I/O

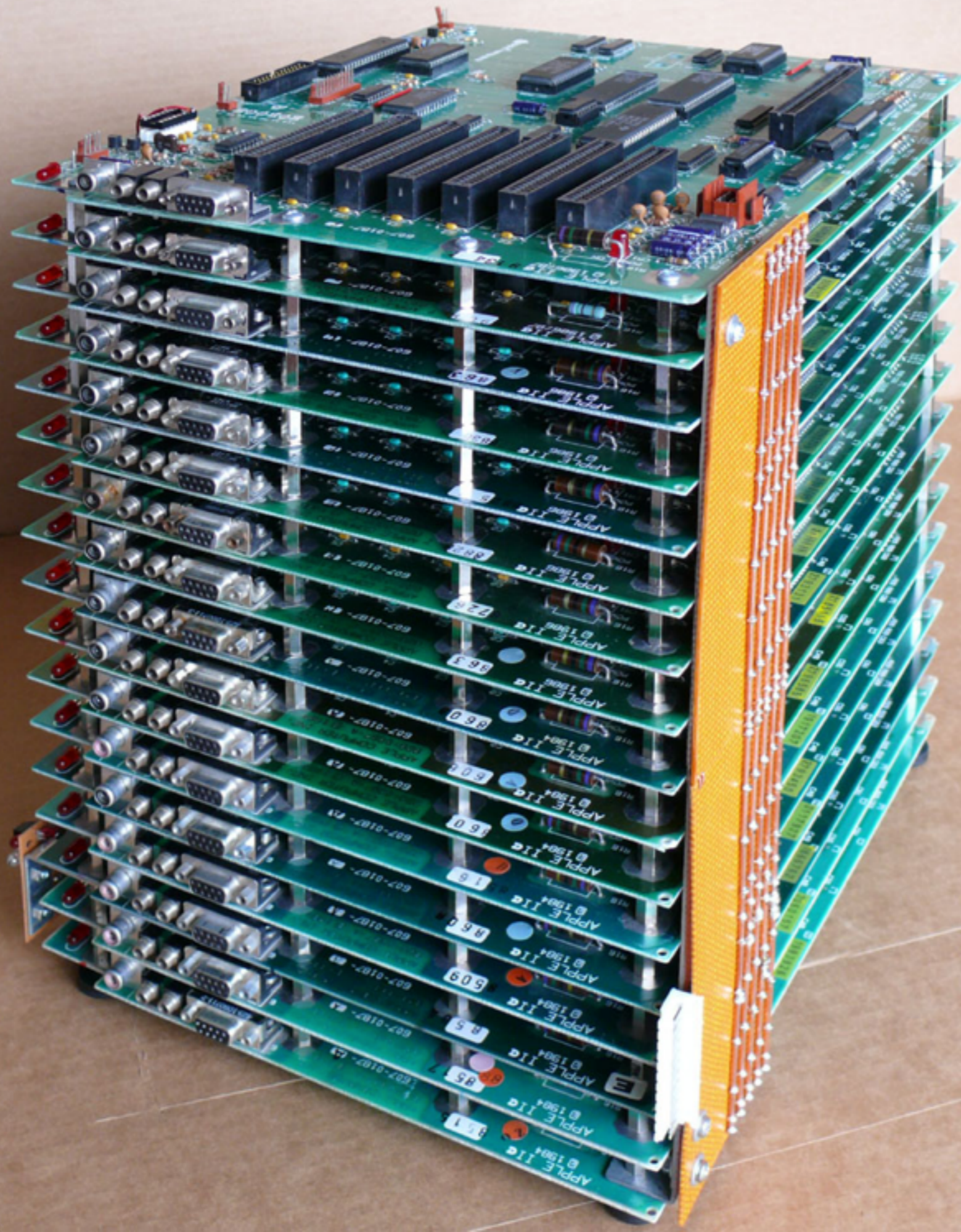
β hardware



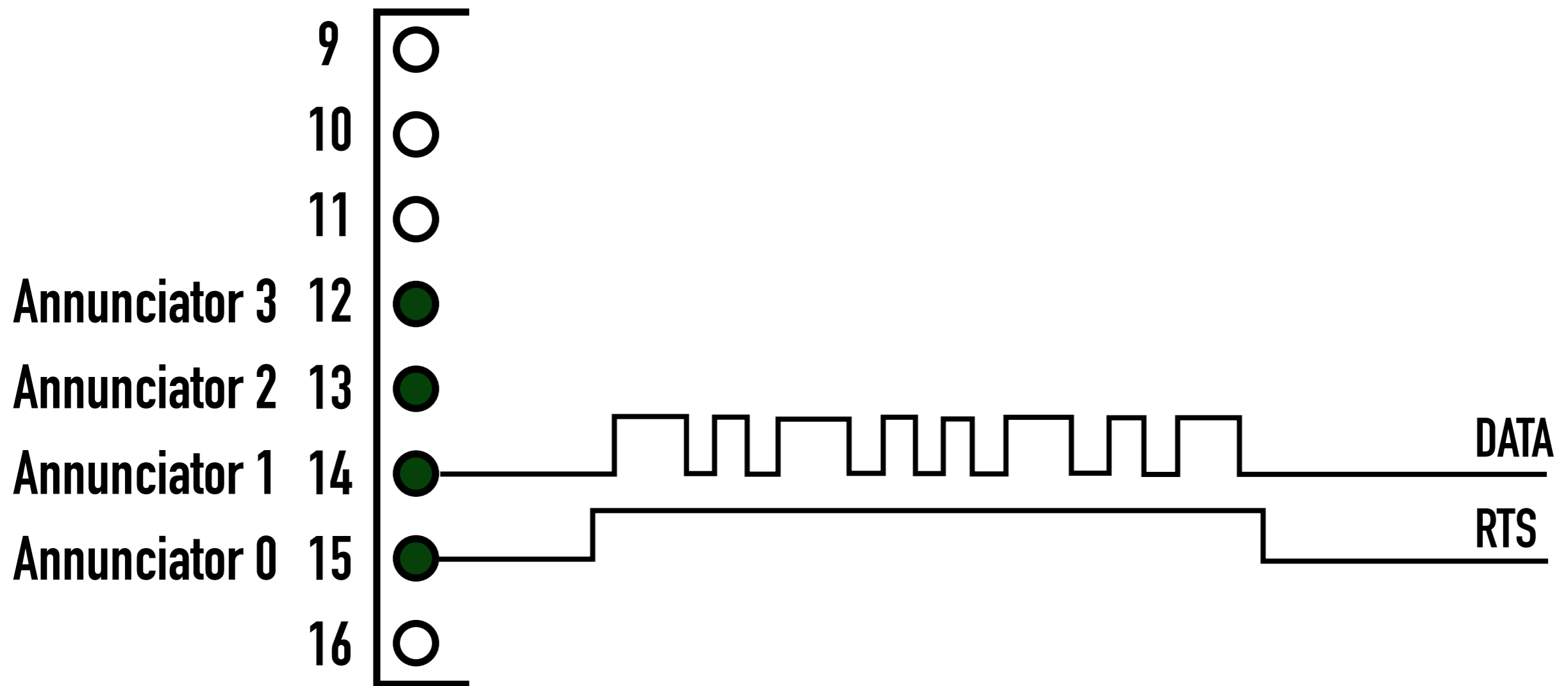
β hardware







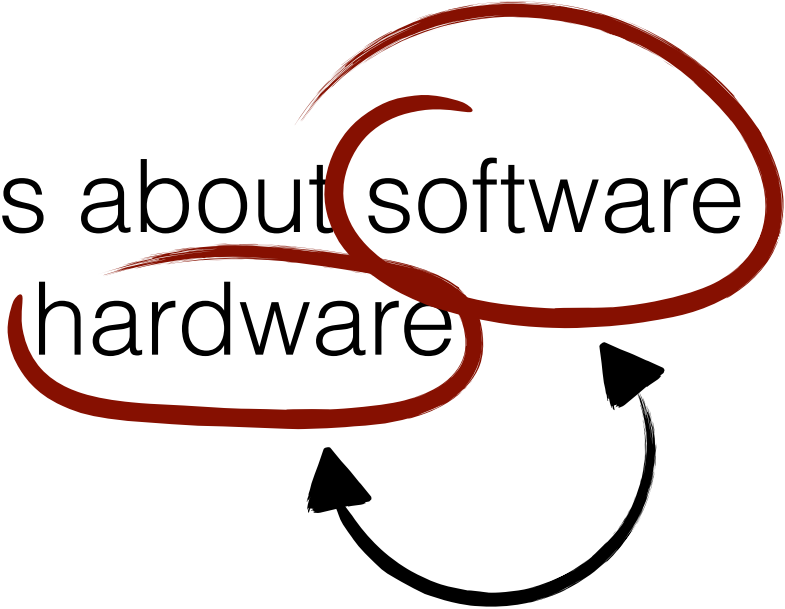
Apple II \Leftrightarrow (GP)²IO



People who are really serious about software
should make their own hardware.

— Alan Kay

People who are really serious about software
should make their own hardware

The text is annotated with two overlapping red circles. The upper circle encloses the word 'software', and the lower circle encloses the word 'hardware'. A black curved arrow starts from the bottom of the 'software' circle and points to the top of the 'hardware' circle, indicating a relationship or flow between the two concepts.

— Alan Kay

0303-	A0	08		LDY #\$08
0305-	85	EF		STA \$EF
0307-	8D	59	C0	STA \$C059
030A-	26	EF		ROL \$EF
030C-	90	03		BCC \$0311
030E-	A2	14		LDX #\$14
0310-	2C	A2	0A	BIT \$0AA2 / LDX #\$0A
0313-	8D	5B	C0	STA \$C05B
0316-	CA			DEX
0317-	D0	FD		BNE \$0316
0319-	8D	5A	C0	STA \$C05A
031C-	A2	05		LDX #\$05
031E-	CA			DEX
031F-	D0	FD		BNE \$031E
0321-	88			DEY
0322-	D0	E6		BNE \$030A
0324-	8D	58	C0	STA \$C058
0327-	60			RTS

0303-	A0	08		LDY #\$08
0305-	85	EF		STA \$EF
0307-	8D	59	C0	STA \$C059
030A-	26	EF		ROL \$EF
030C-	90	03		BCC \$0311
030E-	A2	14		LDX #\$14
0310-	2C	A2	0A	BIT \$0AA2 / LDX #\$0A
0313-	8D	5B	C0	STA \$C05B
0316-	CA			DEX
0317-	D0	FD		BNE \$0316
0319-	8D	5A	C0	STA \$C05A
031C-	A2	05		LDX #\$05
031E-	CA			DEX
031F-	D0	FD		BNE \$031E
0321-	88			DEY
0322-	D0	E6		BNE \$030A
0324-	8D	58	C0	STA \$C058
0327-	60			RTS

8 bits: 10011001 (0x99)

0303-	A0	08		LDY #\$08
0305-	85	EF		STA \$EF
0307-	8D	59	C0	STA \$C059
030A-	26	EF		ROL \$EF
030C-	90	03		BCC \$0311
030E-	A2	14		LDX #\$14
0310-	2C	A2	0A	BIT \$0AA2 / LDX #\$0A
0313-	8D	5B	C0	STA \$C05B
0316-	CA			DEX
0317-	D0	FD		BNE \$0316
0319-	8D	5A	C0	STA \$C05A
031C-	A2	05		LDX #\$05
031E-	CA			DEX
031F-	D0	FD		BNE \$031E
0321-	88			DEY
0322-	D0	E6		BNE \$030A
0324-	8D	58	C0	STA \$C058
0327-	60			RTS

8 bits: 10011001 (0x99)

safe keeping in zero page

0303-	A0	08		LDY #\$08
0305-	85	EF		STA \$EF
0307-	8D	59	C0	STA \$C059
030A-	26	EF		ROL \$EF
030C-	90	03		BCC \$0311
030E-	A2	14		LDX #\$14
0310-	2C	A2	0A	BIT \$0AA2 / LDX #\$0A
0313-	8D	5B	C0	STA \$C05B
0316-	CA			DEX
0317-	D0	FD		BNE \$0316
0319-	8D	5A	C0	STA \$C05A
031C-	A2	05		LDX #\$05
031E-	CA			DEX
031F-	D0	FD		BNE \$031E
0321-	88			DEY
0322-	D0	E6		BNE \$030A
0324-	8D	58	C0	STA \$C058
0327-	60			RTS

8 bits: 10011001 (0x99)
safe keeping in zero page
ready to send (RTS HIGH)

0303-	A0	08		LDY #\$08
0305-	85	EF		STA \$EF
0307-	8D	59	C0	STA \$C059
030A-	26	EF		ROL \$EF
030C-	90	03		BCC \$0311
030E-	A2	14		LDX #\$14
0310-	2C	A2	0A	BIT \$0AA2 / LDX #\$0A
0313-	8D	5B	C0	STA \$C05B
0316-	CA			DEX
0317-	D0	FD		BNE \$0316
0319-	8D	5A	C0	STA \$C05A
031C-	A2	05		LDX #\$05
031E-	CA			DEX
031F-	D0	FD		BNE \$031E
0321-	88			DEY
0322-	D0	E6		BNE \$030A
0324-	8D	58	C0	STA \$C058
0327-	60			RTS

8 bits: 10011001 (0x99)
safe keeping in zero page
ready to send (RTS HIGH)
high bit to Carry

0303-	A0	08		LDY #\$08
0305-	85	EF		STA \$EF
0307-	8D	59	C0	STA \$C059
030A-	26	EF		ROL \$EF
030C-	90	03		BCC \$0311
030E-	A2	14		LDX #\$14
0310-	2C	A2	0A	BIT \$0AA2 / LDX #\$0A
0313-	8D	5B	C0	STA \$C05B
0316-	CA			DEX
0317-	D0	FD		BNE \$0316
0319-	8D	5A	C0	STA \$C05A
031C-	A2	05		LDX #\$05
031E-	CA			DEX
031F-	D0	FD		BNE \$031E
0321-	88			DEY
0322-	D0	E6		BNE \$030A
0324-	8D	58	C0	STA \$C058
0327-	60			RTS

8 bits: 10011001 (0x99)
safe keeping in zero page
ready to send (RTS HIGH)
high bit to Carry
bit is 1, long loop

0303-	A0	08		LDY #\$08
0305-	85	EF		STA \$EF
0307-	8D	59	C0	STA \$C059
030A-	26	EF		ROL \$EF
030C-	90	03		BCC \$0311
030E-	A2	14		LDX #\$14
0310-	2C	A2	0A	BIT \$0AA2 / LDX #\$0A
0313-	8D	5B	C0	STA \$C05B
0316-	CA			DEX
0317-	D0	FD		BNE \$0316
0319-	8D	5A	C0	STA \$C05A
031C-	A2	05		LDX #\$05
031E-	CA			DEX
031F-	D0	FD		BNE \$031E
0321-	88			DEY
0322-	D0	E6		BNE \$030A
0324-	8D	58	C0	STA \$C058
0327-	60			RTS

8 bits: 10011001 (0x99)
safe keeping in zero page
ready to send (RTS HIGH)
high bit to Carry
bit is 1, long loop
long loop: 20 iterations

0303-	A0	08		LDY #\$08
0305-	85	EF		STA \$EF
0307-	8D	59	C0	STA \$C059
030A-	26	EF		ROL \$EF
030C-	90	03		BCC \$0311
030E-	A2	14		LDX #\$14
0310-	2C	A2	0A	BIT \$0AA2 / LDX #\$0A
0313-	8D	5B	C0	STA \$C05B
0316-	CA			DEX
0317-	D0	FD		BNE \$0316
0319-	8D	5A	C0	STA \$C05A
031C-	A2	05		LDX #\$05
031E-	CA			DEX
031F-	D0	FD		BNE \$031E
0321-	88			DEY
0322-	D0	E6		BNE \$030A
0324-	8D	58	C0	STA \$C058
0327-	60			RTS

8 bits: 10011001 (0x99)

safe keeping in zero page

ready to send (RTS HIGH)

high bit to Carry

bit is 1, long loop

long loop: 20 iterations

Annunciator 1 (DATA HIGH)

0303-	A0	08		LDY #\$08
0305-	85	EF		STA \$EF
0307-	8D	59	C0	STA \$C059
030A-	26	EF		ROL \$EF
030C-	90	03		BCC \$0311
030E-	A2	14		LDX #\$14
0310-	2C	A2	0A	BIT \$0AA2 / LDX #\$0A
0313-	8D	5B	C0	STA \$C05B
0316-	CA			DEX
0317-	D0	FD		BNE \$0316
0319-	8D	5A	C0	STA \$C05A
031C-	A2	05		LDX #\$05
031E-	CA			DEX
031F-	D0	FD		BNE \$031E
0321-	88			DEY
0322-	D0	E6		BNE \$030A
0324-	8D	58	C0	STA \$C058
0327-	60			RTS

8 bits: 10011001 (0x99)

safe keeping in zero page

ready to send (RTS HIGH)

high bit to Carry

bit is 1, long loop

long loop: 20 iterations

Annunciator 1 (DATA HIGH)

delay loop counter

repeat while X != 0

0303-	A0	08		LDY #\$08	8 bits: 10011001 (0x99)
0305-	85	EF		STA \$EF	safe keeping in zero page
0307-	8D	59	C0	STA \$C059	ready to send (RTS HIGH)
030A-	26	EF		ROL \$EF	high bit to Carry
030C-	90	03		BCC \$0311	bit is 1, long loop
030E-	A2	14		LDX #\$14	long loop: 20 iterations
0310-	2C	A2	0A	BIT \$0AA2 / LDX #\$0A	
0313-	8D	5B	C0	STA \$C05B	Annunciator 1 (DATA HIGH)
0316-	CA			DEX	delay loop counter
0317-	D0	FD		BNE \$0316	repeat while X != 0
0319-	8D	5A	C0	STA \$C05A	Annunciator 1 (DATA LOW)
031C-	A2	05		LDX #\$05	
031E-	CA			DEX	
031F-	D0	FD		BNE \$031E	
0321-	88			DEY	
0322-	D0	E6		BNE \$030A	
0324-	8D	58	C0	STA \$C058	
0327-	60			RTS	

0303-	A0	08		LDY #\$08
0305-	85	EF		STA \$EF
0307-	8D	59	C0	STA \$C059
030A-	26	EF		ROL \$EF
030C-	90	03		BCC \$0311
030E-	A2	14		LDX #\$14
0310-	2C	A2	0A	BIT \$0AA2 / LDX #\$0A
0313-	8D	5B	C0	STA \$C05B
0316-	CA			DEX
0317-	D0	FD		BNE \$0316
0319-	8D	5A	C0	STA \$C05A
031C-	A2	05		LDX #\$05
031E-	CA			DEX
031F-	D0	FD		BNE \$031E
0321-	88			DEY
0322-	D0	E6		BNE \$030A
0324-	8D	58	C0	STA \$C058
0327-	60			RTS

8 bits: 10011001 (0x99)

safe keeping in zero page

ready to send (RTS HIGH)

high bit to Carry

bit is 1, long loop

long loop: 20 iterations

Annunciator 1 (DATA HIGH)

delay loop counter

repeat while X != 0

Annunciator 1 (DATA LOW)

reset delay loop

0303-	A0	08		LDY #\$08	8 bits: 10011001 (0x99)
0305-	85	EF		STA \$EF	safe keeping in zero page
0307-	8D	59	C0	STA \$C059	ready to send (RTS HIGH)
030A-	26	EF		ROL \$EF	high bit to Carry
030C-	90	03		BCC \$0311	bit is 1, long loop
030E-	A2	14		LDX #\$14	long loop: 20 iterations
0310-	2C	A2	0A	BIT \$0AA2 / LDX #\$0A	
0313-	8D	5B	C0	STA \$C05B	Annunciator 1 (DATA HIGH)
0316-	CA			DEX	delay loop counter
0317-	D0	FD		BNE \$0316	repeat while X != 0
0319-	8D	5A	C0	STA \$C05A	Annunciator 1 (DATA LOW)
031C-	A2	05		LDX #\$05	reset delay loop
031E-	CA			DEX	
031F-	D0	FD		BNE \$031E	
0321-	88			DEY	count down 8 bits
0322-	D0	E6		BNE \$030A	
0324-	8D	58	C0	STA \$C058	
0327-	60			RTS	

0303-	A0	08		LDY #\$08
0305-	85	EF		STA \$EF
0307-	8D	59	C0	STA \$C059
030A-	26	EF		ROL \$EF
030C-	90	03		BCC \$0311
030E-	A2	14		LDX #\$14
0310-	2C	A2	0A	BIT \$0AA2 / LDX #\$0A
0313-	8D	5B	C0	STA \$C05B
0316-	CA			DEX
0317-	D0	FD		BNE \$0316
0319-	8D	5A	C0	STA \$C05A
031C-	A2	05		LDX #\$05
031E-	CA			DEX
031F-	D0	FD		BNE \$031E
0321-	88			DEY
0322-	D0	E6		BNE \$030A
0324-	8D	58	C0	STA \$C058
0327-	60			RTS

8 bits: 10011001 (0x99)
safe keeping in zero page
ready to send (RTS HIGH)
high bit to Carry
bit is 1, long loop
long loop: 20 iterations

Annunciator 1 (DATA HIGH)
delay loop counter
repeat while X != 0
Annunciator 1 (DATA LOW)
reset delay loop

count down 8 bits
loop until done with byte

0303-	A0	08		LDY #\$08	8 bits: 10011001 (0x99)
0305-	85	EF		STA \$EF	safe keeping in zero page
0307-	8D	59	C0	STA \$C059	ready to send (RTS HIGH)
030A-	26	EF		ROL \$EF	high bit to Carry
030C-	90	03		BCC \$0311	bit is 1, long loop
030E-	A2	14		LDX #\$14	long loop: 20 iterations
0310-	2C	A2	0A	BIT \$0AA2 / LDX #\$0A	
0313-	8D	5B	C0	STA \$C05B	Annunciator 1 (DATA HIGH)
0316-	CA			DEX	delay loop counter
0317-	D0	FD		BNE \$0316	repeat while X != 0
0319-	8D	5A	C0	STA \$C05A	Annunciator 1 (DATA LOW)
031C-	A2	05		LDX #\$05	reset delay loop
031E-	CA			DEX	
031F-	D0	FD		BNE \$031E	
0321-	88			DEY	count down 8 bits
0322-	D0	E6		BNE \$030A	loop until done with byte
0324-	8D	58	C0	STA \$C058	done sending (RTS LOW)
0327-	60			RTS	

0303-	A0	08		LDY #\$08	8 bits: 10011001 (0x99)
0305-	85	EF		STA \$EF	safe keeping in zero page
0307-	8D	59	C0	STA \$C059	ready to send (RTS HIGH)
030A-	26	EF		ROL \$EF	high bit to Carry
030C-	90	03		BCC \$0311	bit is 1, long loop
030E-	A2	14		LDX #\$14	long loop: 20 iterations
0310-	2C	A2	0A	BIT \$0AA2 / LDX #\$0A	
0313-	8D	5B	C0	STA \$C05B	Annunciator 1 (DATA HIGH)
0316-	CA			DEX	delay loop counter
0317-	D0	FD		BNE \$0316	repeat while X != 0
0319-	8D	5A	C0	STA \$C05A	Annunciator 1 (DATA LOW)
031C-	A2	05		LDX #\$05	reset delay loop
031E-	CA			DEX	
031F-	D0	FD		BNE \$031E	
0321-	88			DEY	count down 8 bits
0322-	D0	E6		BNE \$030A	loop until done with byte
0324-	8D	58	C0	STA \$C058	done sending (RTS LOW)
0327-	60			RTS	return to sender.

example: SENDKEY

```
20 1B FD JSR $FD1B
C9 9B CMP #$9B
F0 0B BEQ $0812
20 ED FD JSR $FDF0
09 80 ORA #$80
20 03 03 JSR $0303
4C 00 08 JMP $0800
60 RTS
```

example: SENDKEY

```
20 1B FD JSR $FD1B
```

keyboard input, byte now in Accumulator

```
C9 9B CMP #$9B
```

```
F0 0B BEQ $0812
```

```
20 ED FD JSR $FDF0
```

```
09 80 ORA #$80
```

```
20 03 03 JSR $0303
```

```
4C 00 08 JMP $0800
```

```
60 RTS
```

example: SENDKEY

20 1B FD JSR \$FD1B

keyboard input, byte now in Accumulator

C9 9B CMP #\$9B

check for ESC

F0 0B BEQ \$0812

20 ED FD JSR \$FDF0

09 80 ORA #\$80

20 03 03 JSR \$0303

4C 00 08 JMP \$0800

60 RTS

example: SENDKEY

20 1B FD JSR \$FD1B

keyboard input, byte now in Accumulator

C9 9B CMP #\$9B

check for ESC

F0 0B BEQ \$0812

return on ESC

20 ED FD JSR \$FDF0

09 80 ORA #\$80

20 03 03 JSR \$0303

4C 00 08 JMP \$0800

60 RTS

example: SENDKEY

20 1B FD	JSR \$FD1B	keyboard input, byte now in Accumulator
C9 9B	CMP #\$9B	check for ESC
F0 0B	BEQ \$0812	return on ESC
20 ED FD	JSR \$FDF0	print byte to screen (local echo)
09 80	ORA #\$80	
20 03 03	JSR \$0303	
4C 00 08	JMP \$0800	
60	RTS	

example: SENDKEY

20 1B FD	JSR \$FD1B	keyboard input, byte now in Accumulator
C9 9B	CMP #\$9B	check for ESC
F0 0B	BEQ \$0812	return on ESC
20 ED FD	JSR \$FDF0	print byte to screen (local echo)
09 80	ORA #\$80	clear bit 7 ("low" vs "high" ASCII)
20 03 03	JSR \$0303	
4C 00 08	JMP \$0800	
60	RTS	

example: SENDKEY

20 1B FD	JSR \$FD1B	keyboard input, byte now in Accumulator
C9 9B	CMP #\$9B	check for ESC
F0 0B	BEQ \$0812	return on ESC
20 ED FD	JSR \$FDF0	print byte to screen (local echo)
09 80	ORA #\$80	clear bit 7 ("low" vs "high" ASCII)
20 03 03	JSR \$0303	send the byte to GP2IO (SENDBYTE)
4C 00 08	JMP \$0800	
60	RTS	

example: SENDKEY

```
20 1B FD JSR $FD1B
C9 9B    CMP  #$9B
F0 0B    BEQ  $0812
20 ED FD JSR $FDF0
09 80    ORA  #$80
20 03 03 JSR $0303
4C 00 08 JMP  $0800
60      RTS
```

keyboard input, byte now in Accumulator
check for ESC
return on ESC
print byte to screen (local echo)
clear bit 7 ("low" vs "high" ASCII)
send the byte to GP2IO (SENDBYTE)
rinse and repeat

example: SENDKEY

20 1B FD	JSR \$FD1B	keyboard input, byte now in Accumulator
C9 9B	CMP #\$9B	check for ESC
F0 0B	BEQ \$0812	return on ESC
20 ED FD	JSR \$FDF0	print byte to screen (local echo)
09 80	ORA #\$80	clear bit 7 ("low" vs "high" ASCII)
20 03 03	JSR \$0303	send the byte to GP2IO (SENDBYTE)
4C 00 08	JMP \$0800	rinse and repeat
60	RTS	return on ESC

But why, Charles?
WHY?

```
attachInterrupt(0, APPLERTS, RISING); // ANNUNCIATOR 0, Apple sending byte
attachInterrupt(1, RECEIVINGBITS, CHANGE); // ANNUNCIATOR 1, Apple sending bits
```

```
[...]
```

```
void APPLERTS() {
    // signal to start receiving bits from Apple II
    bitCount = 0;
    changeCount = 0;
    returnByte = B00000000;
}
```

```
void RECEIVINGBITS() {
    currentMicros = micros();
    if (changeCount % 2 == 1) {
        if ((currentMicros - lastMicros) > 70) { // short bit or long bit?
            receivedBit = 1;
        } else {
            receivedBit = 0;
        }
        byteArray[7 - bitCount] = receivedBit;
        bitCount++;
    }
}
```

```
changeCount++;
```

```
if (bitCount == 8) { // got a BYTE
    receivedByte = arrayToByte(byteArray, 8);
    PROCESSBYTE( byte(receivedByte) );
    TIMEOUTCLOCK = millis();
}
```

```
}
```

```
attachInterrupt(0, APPLERTS, RISING); // ANNUNCIATOR 0, Apple sending byte
attachInterrupt(1, RECEIVINGBITS, CHANGE); // ANNUNCIATOR 1, Apple sending bits
```

[...]

```
void APPLERTS() {
    // signal to start receiving bits from Apple II
    bitCount = 0;
    changeCount = 0;
    returnByte = B00000000;
}
```

```
void RECEIVINGBITS() {
    currentMicros = micros();
    if (changeCount % 2 == 1) {
        if ((currentMicros - lastMicros) > 70) { // short bit or long bit?
            receivedBit = 1;
        } else {
            receivedBit = 0;
        }
        byteArray[7 - bitCount] = receivedBit;
        bitCount++;
    }

    changeCount++;
```



```
    if (bitCount == 8) { // got a BYTE
        receivedByte = arrayToByte(byteArray, 8);
        PROCESSBYTE( byte(receivedByte) );
        TIMEOUTCLOCK = millis();
    }
}
```

Control Byte Mode

Message Format

Control Byte Mode

\$01

RGB LED

Message Format

3 bytes. Order Red, Green, Blue.

Control Byte Mode

Message Format

\$01	RGB LED	3 bytes. Order Red, Green, Blue.
\$02	RGB LED 2	1 byte. White LED intensity.

Control Byte Mode Message Format

\$01	RGB LED	3 bytes. Order Red, Green, Blue.
\$02	RGB LED 2	1 byte. White LED intensity.
\$04	BUFFER	1 byte, N bytes. Write to an internal buffer, for retrieval later.

Control Byte Mode Message Format

\$01	RGB LED	3 bytes. Order Red, Green, Blue.
\$02	RGB LED 2	1 byte. White LED intensity.
\$04	BUFFER	1 byte, N bytes. Write to an internal buffer, for retrieval later.
\$08	UART	1 byte, N bytes. Write N bytes to UART serial (buffered).

Control Byte Mode Message Format

\$01	RGB LED	3 bytes. Order Red, Green, Blue.
\$02	RGB LED 2	1 byte. White LED intensity.
\$04	BUFFER	1 byte, N bytes. Write to an internal buffer, for retrieval later.
\$08	UART	1 byte, N bytes. Write N bytes to UART serial (buffered).
\$10	I²C	1 byte, N bytes. Write N bytes to I2C bus.

Control Byte Mode Message Format

\$01	RGB LED	3 bytes. Order Red, Green, Blue.
\$02	RGB LED 2	1 byte. White LED intensity.
\$04	BUFFER	1 byte, N bytes. Write to an internal buffer, for retrieval later.
\$08	UART	1 byte, N bytes. Write N bytes to UART serial (buffered).
\$10	I²C	1 byte, N bytes. Write N bytes to I2C bus.
\$20	SPI	(not yet implemented)
	READPIN	1 byte. Get analog state of any AVR pin.

Control Byte	Mode	Message Format
\$01	RGB LED	3 bytes. Order Red, Green, Blue.
\$02	RGB LED 2	1 byte. White LED intensity.
\$04	BUFFER	1 byte, N bytes. Write to an internal buffer, for retrieval later.
\$08	UART	1 byte, N bytes. Write N bytes to UART serial (buffered).
\$10	I²C	1 byte, N bytes. Write N bytes to I2C bus.
\$20	SPI	(not yet implemented)
	READPIN	1 byte. Get analog state of any AVR pin.
\$40	QUERY	1 byte. GP2IO responds with 1 byte, containing the length of the buffer.

Control Byte	Mode	Message Format
\$01	RGB LED	3 bytes. Order Red, Green, Blue.
\$02	RGB LED 2	1 byte. White LED intensity.
\$04	BUFFER	1 byte, N bytes. Write to an internal buffer, for retrieval later.
\$08	UART	1 byte, N bytes. Write N bytes to UART serial (buffered).
\$10	I²C	1 byte, N bytes. Write N bytes to I2C bus.
\$20	SPI	(not yet implemented)
	READPIN	1 byte. Get analog state of any AVR pin.
\$40	QUERY	1 byte. GP2IO responds with 1 byte, containing the length of the buffer.
\$80	SEND	1 byte. GP2IO responds with the first N bytes of the buffer, zero padded

Control Byte	Mode	Message Format
\$01	RGB LED	3 bytes. Order Red, Green, Blue.
\$02	RGB LED 2	1 byte. White LED intensity.
\$04	BUFFER	1 byte, N bytes. Write to an internal buffer, for retrieval later.
\$08	UART	1 byte, N bytes. Write N bytes to UART serial (buffered).
\$10	I²C	1 byte, N bytes. Write N bytes to I2C bus.
\$20	SPI	(not yet implemented)
	READPIN	1 byte. Get analog state of any AVR pin.
\$40	QUERY	1 byte. GP2IO responds with 1 byte, containing the length of the buffer.
\$80	SEND	1 byte. GP2IO responds with the first N bytes of the buffer, zero padded
\$00	DEBUG	echoes the bytes received from to USB serial out sends one byte from the GP2IO buffer to the Apple when APPLECTS (Annunciator 2) goes high.

But why, Charles?
WHY?

BLINKY LIGHTS!

BLINKY LIGHTS!

(also, WiFi)

BLINKY LIGHTS!

(also, WiFi)

(but seriously, blinky lights)

example: SETRGB

```
A9 01      LDA #$01
20 03 03   JSR $0303
A5 06      LDA $06
20 03 03   JSR $0303
A5 07      LDA $07
20 03 03   JSR $0303
A5 08      LDA $08
20 03 03   JSR $0303
60         RTS
```

example: SETRGB

A9 01 LDA #\$01

set accumulator to 1, RGBLED mode

20 03 03 JSR \$0303

A5 06 LDA \$06

20 03 03 JSR \$0303

A5 07 LDA \$07

20 03 03 JSR \$0303

A5 08 LDA \$08

20 03 03 JSR \$0303

60 RTS

example: SETRGB

A9 01	LDA #01	set accumulator to 1, RGBLED mode
20 03 03	JSR \$0303	SENDBYTE control byte
A5 06	LDA \$06	
20 03 03	JSR \$0303	
A5 07	LDA \$07	
20 03 03	JSR \$0303	
A5 08	LDA \$08	
20 03 03	JSR \$0303	
60	RTS	

example: SETRGB

A9 01	LDA # \$01	set accumulator to 1, RGBLED mode
20 03 03	JSR \$0303	SENDBYTE control byte
A5 06	LDA \$06	read in red value from zero page
20 03 03	JSR \$0303	
A5 07	LDA \$07	
20 03 03	JSR \$0303	
A5 08	LDA \$08	
20 03 03	JSR \$0303	
60	RTS	

example: SETRGB

A9 01	LDA # \$01	set accumulator to 1, RGBLED mode
20 03 03	JSR \$0303	SENDBYTE control byte
A5 06	LDA \$06	read in red value from zero page
20 03 03	JSR \$0303	SENDBYTE red
A5 07	LDA \$07	
20 03 03	JSR \$0303	
A5 08	LDA \$08	
20 03 03	JSR \$0303	
60	RTS	

example: SETRGB

A9 01	LDA #01	set accumulator to 1, RGBLED mode
20 03 03	JSR \$0303	SENDBYTE control byte
A5 06	LDA \$06	read in red value from zero page
20 03 03	JSR \$0303	SENDBYTE red
A5 07	LDA \$07	read green value
20 03 03	JSR \$0303	
A5 08	LDA \$08	
20 03 03	JSR \$0303	
60	RTS	

example: SETRGB

A9 01	LDA #01	set accumulator to 1, RGBLED mode
20 03 03	JSR \$0303	SENDBYTE control byte
A5 06	LDA \$06	read in red value from zero page
20 03 03	JSR \$0303	SENDBYTE red
A5 07	LDA \$07	read green value
20 03 03	JSR \$0303	SENDBYTE green
A5 08	LDA \$08	
20 03 03	JSR \$0303	
60	RTS	

example: SETRGB

A9 01	LDA #01	set accumulator to 1, RGBLED mode
20 03 03	JSR \$0303	SENDBYTE control byte
A5 06	LDA \$06	read in red value from zero page
20 03 03	JSR \$0303	SENDBYTE red
A5 07	LDA \$07	read green value
20 03 03	JSR \$0303	SENDBYTE green
A5 08	LDA \$08	read blue value
20 03 03	JSR \$0303	
60	RTS	

example: SETRGB

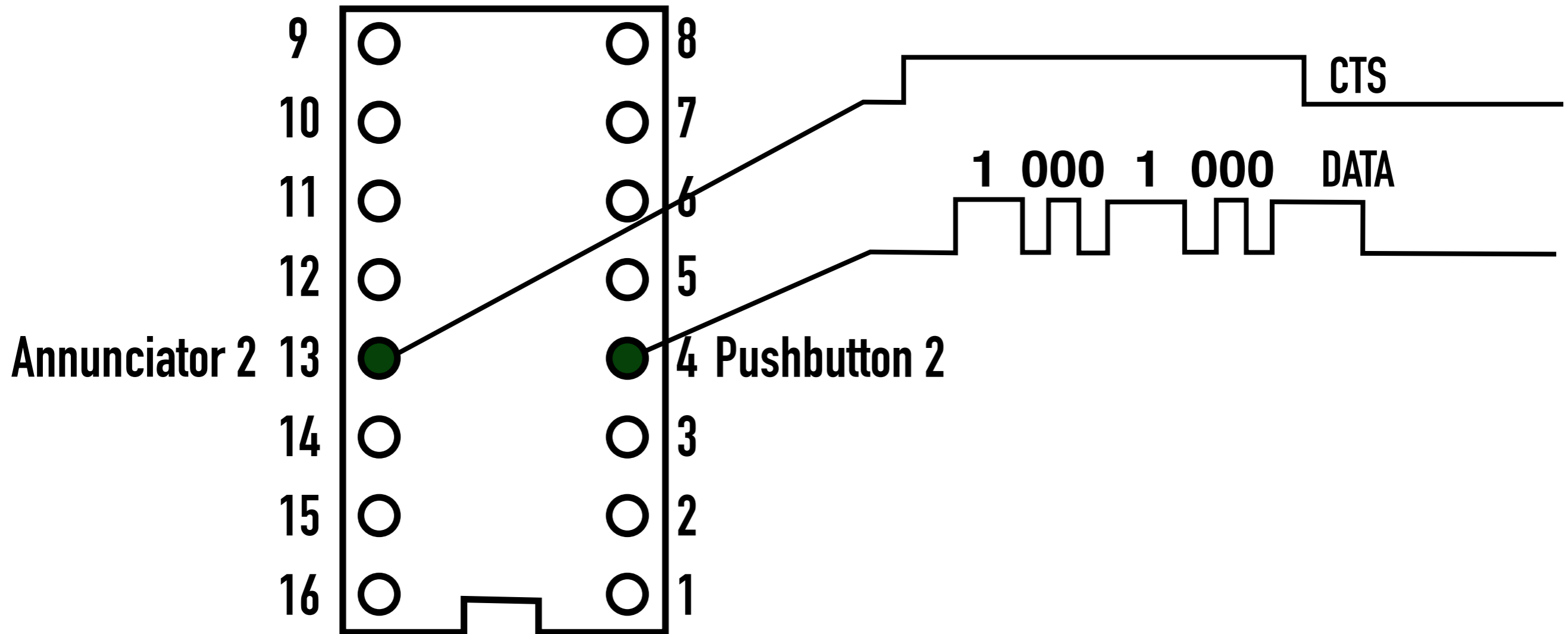
A9 01	LDA #01	set accumulator to 1, RGBLED mode
20 03 03	JSR \$0303	SENDBYTE control byte
A5 06	LDA \$06	read in red value from zero page
20 03 03	JSR \$0303	SENDBYTE red
A5 07	LDA \$07	read green value
20 03 03	JSR \$0303	SENDBYTE green
A5 08	LDA \$08	read blue value
20 03 03	JSR \$0303	SENDBYTE blue
60	RTS	

example: SETRGB

A9 01	LDA #01	set accumulator to 1, RGBLED mode
20 03 03	JSR \$0303	SENDBYTE control byte
A5 06	LDA \$06	read in red value from zero page
20 03 03	JSR \$0303	SENDBYTE red
A5 07	LDA \$07	read green value
20 03 03	JSR \$0303	SENDBYTE green
A5 08	LDA \$08	read blue value
20 03 03	JSR \$0303	SENDBYTE blue
60	RTS	return.

demo, CYCLERGB

(GP)²IO \Rightarrow Apple II



```
void sendByte (byte byteToSend) {  
  
    delay(100); // ~20ms  
    digitalWrite(PB1_PIN, HIGH);  
  
    for (int i = 0; i < 8; i++) { // send 8 bits, 9 transitions  
  
        int bitToSend = bitRead(byteToSend, 7 - i);  
  
        if (bitToSend == 1) {  
            delay(80); // ~13ms  
        }  
  
        delay(20); // ~7ms  
  
        if ( i % 2 ) {  
            digitalWrite(PB1_PIN, HIGH);  
        } else {  
            digitalWrite(PB1_PIN, LOW);  
        }  
    }  
  
    delay(100); // ~20ms  
    digitalWrite(PB1_PIN, HIGH);  
    delay(100); // ~20ms  
    digitalWrite(PB1_PIN, LOW);  
  
}
```


A2	09		LDX	#\$09
A9	00		LDA	#\$00
85	EF		STA	\$EF
85	EE		STA	\$EE
8D	5D	C0	STA	\$C05D
A0	FF		LDY	#\$FF
C8			INY	
AD	62	C0	LDA	\$C062
29	80		AND	#\$80
C5	EE		CMP	\$EE
D0	02		BNE	\$0363
F0	F4		BEQ	\$0357
85	EE		STA	\$EE
C0	44		CPY	#\$44
26	EF		ROL	\$EF
CA			DEX	
D0	E9		BNE	\$0355
8D	5C	C0	STA	\$C05C
A5	EF		LDA	\$EF
60			RTS	

A2 09 LDX #\$09

reading 8 bits. requires 9 transitions.

A9 00 LDA #\$00

85 EF STA \$EF

85 EE STA \$EE

8D 5D C0 STA \$C05D

A0 FF LDY #\$FF

C8 INY

AD 62 C0 LDA \$C062

29 80 AND #\$80

C5 EE CMP \$EE

D0 02 BNE \$0363

F0 F4 BEQ \$0357

85 EE STA \$EE

C0 44 CPY #\$44

26 EF ROL \$EF

CA DEX

D0 E9 BNE \$0355

8D 5C C0 STA \$C05C

A5 EF LDA \$EF

60 RTS

A2 09 LDX #\$09

reading 8 bits. requires 9 transitions.

A9 00 LDA #\$00

clear the accumulator

85 EF STA \$EF

\$EF is staging for received byte, set to 0

85 EE STA \$EE

\$EE is staging for each bit, set to 0

8D 5D C0 STA \$C05D

A0 FF LDY #\$FF

C8 INY

AD 62 C0 LDA \$C062

29 80 AND #\$80

C5 EE CMP \$EE

D0 02 BNE \$0363

F0 F4 BEQ \$0357

85 EE STA \$EE

C0 44 CPY #\$44

26 EF ROL \$EF

CA DEX

D0 E9 BNE \$0355

8D 5C C0 STA \$C05C

A5 EF LDA \$EF

60 RTS

A2	09		LDX	#\$09	reading 8 bits. requires 9 transitions.
A9	00		LDA	#\$00	clear the accumulator
85	EF		STA	\$EF	\$EF is staging for received byte, set to 0
85	EE		STA	\$EE	\$EE is staging for each bit, set to 0
8D	5D	C0	STA	\$C05D	set ANN2 HIGH, indicate to AVR "Clear to Send"
A0	FF		LDY	#\$FF	
C8			INY		
AD	62	C0	LDA	\$C062	
29	80		AND	#\$80	
C5	EE		CMP	\$EE	
D0	02		BNE	\$0363	
F0	F4		BEQ	\$0357	
85	EE		STA	\$EE	
C0	44		CPY	#\$44	
26	EF		ROL	\$EF	
CA			DEX		
D0	E9		BNE	\$0355	
8D	5C	C0	STA	\$C05C	
A5	EF		LDA	\$EF	
60			RTS		

A2	09		LDX	#\$09	reading 8 bits. requires 9 transitions.
A9	00		LDA	#\$00	clear the accumulator
85	EF		STA	\$EF	\$EF is staging for received byte, set to 0
85	EE		STA	\$EE	\$EE is staging for each bit, set to 0
8D	5D	C0	STA	\$C05D	set ANN2 HIGH, indicate to AVR "Clear to Send"
A0	FF		LDY	#\$FF	start wait loop
C8			INY		Increment Y - rolls over to 0 on first run
AD	62	C0	LDA	\$C062	
29	80		AND	#\$80	
C5	EE		CMP	\$EE	
D0	02		BNE	\$0363	
F0	F4		BEQ	\$0357	
85	EE		STA	\$EE	
C0	44		CPY	#\$44	
26	EF		ROL	\$EF	
CA			DEX		
D0	E9		BNE	\$0355	
8D	5C	C0	STA	\$C05C	
A5	EF		LDA	\$EF	
60			RTS		

A2	09		LDX	#\$09	reading 8 bits. requires 9 transitions.
A9	00		LDA	#\$00	clear the accumulator
85	EF		STA	\$EF	\$EF is staging for received byte, set to 0
85	EE		STA	\$EE	\$EE is staging for each bit, set to 0
8D	5D	C0	STA	\$C05D	set ANN2 HIGH, indicate to AVR "Clear to Send"
A0	FF		LDY	#\$FF	start wait loop
C8			INY		Increment Y - rolls over to 0 on first run
AD	62	C0	LDA	\$C062	put PB1 status into Accumulator
29	80		AND	#\$80	
C5	EE		CMP	\$EE	
D0	02		BNE	\$0363	
F0	F4		BEQ	\$0357	
85	EE		STA	\$EE	
C0	44		CPY	#\$44	
26	EF		ROL	\$EF	
CA			DEX		
D0	E9		BNE	\$0355	
8D	5C	C0	STA	\$C05C	
A5	EF		LDA	\$EF	
60			RTS		

A2	09		LDX	#\$09	reading 8 bits. requires 9 transitions.
A9	00		LDA	#\$00	clear the accumulator
85	EF		STA	\$EF	\$EF is staging for received byte, set to 0
85	EE		STA	\$EE	\$EE is staging for each bit, set to 0
8D	5D	C0	STA	\$C05D	set ANN2 HIGH, indicate to AVR "Clear to Send"
A0	FF		LDY	#\$FF	start wait loop
C8			INY		Increment Y - rolls over to 0 on first run
AD	62	C0	LDA	\$C062	put PB1 status into Accumulator
29	80		AND	#\$80	clear bits 0-6 (just need HIGH vs LOW, bit 7)
C5	EE		CMP	\$EE	
D0	02		BNE	\$0363	
F0	F4		BEQ	\$0357	
85	EE		STA	\$EE	
C0	44		CPY	#\$44	
26	EF		ROL	\$EF	
CA			DEX		
D0	E9		BNE	\$0355	
8D	5C	C0	STA	\$C05C	
A5	EF		LDA	\$EF	
60			RTS		

A2	09		LDX	#\$09	reading 8 bits. requires 9 transitions.
A9	00		LDA	#\$00	clear the accumulator
85	EF		STA	\$EF	\$EF is staging for received byte, set to 0
85	EE		STA	\$EE	\$EE is staging for each bit, set to 0
8D	5D	C0	STA	\$C05D	set ANN2 HIGH, indicate to AVR "Clear to Send"
A0	FF		LDY	#\$FF	start wait loop
C8			INY		Increment Y - rolls over to 0 on first run
AD	62	C0	LDA	\$C062	put PB1 status into Accumulator
29	80		AND	#\$80	clear bits 0-6 (just need HIGH vs LOW, bit 7)
C5	EE		CMP	\$EE	compare Accumulator bit 7 with previous PB2 value
D0	02		BNE	\$0363	
F0	F4		BEQ	\$0357	
85	EE		STA	\$EE	
C0	44		CPY	#\$44	
26	EF		ROL	\$EF	
CA			DEX		
D0	E9		BNE	\$0355	
8D	5C	C0	STA	\$C05C	
A5	EF		LDA	\$EF	
60			RTS		

A2	09		LDX	#\$09	reading 8 bits. requires 9 transitions.
A9	00		LDA	#\$00	clear the accumulator
85	EF		STA	\$EF	\$EF is staging for received byte, set to 0
85	EE		STA	\$EE	\$EE is staging for each bit, set to 0
8D	5D	C0	STA	\$C05D	set ANN2 HIGH, indicate to AVR "Clear to Send"
A0	FF		LDY	#\$FF	start wait loop
C8			INY		Increment Y - rolls over to 0 on first run
AD	62	C0	LDA	\$C062	put PB1 status into Accumulator
29	80		AND	#\$80	clear bits 0-6 (just need HIGH vs LOW, bit 7)
C5	EE		CMP	\$EE	compare Accumulator bit 7 with previous PB2 value
D0	02		BNE	\$0363	if PB2 has changed state, store in \$EE
F0	F4		BEQ	\$0357	
85	EE		STA	\$EE	
C0	44		CPY	#\$44	
26	EF		ROL	\$EF	
CA			DEX		
D0	E9		BNE	\$0355	
8D	5C	C0	STA	\$C05C	
A5	EF		LDA	\$EF	
60			RTS		

A2	09		LDX	#\$09	reading 8 bits. requires 9 transitions.
A9	00		LDA	#\$00	clear the accumulator
85	EF		STA	\$EF	\$EF is staging for received byte, set to 0
85	EE		STA	\$EE	\$EE is staging for each bit, set to 0
8D	5D	C0	STA	\$C05D	set ANN2 HIGH, indicate to AVR "Clear to Send"
A0	FF		LDY	#\$FF	start wait loop
C8			INY		Increment Y - rolls over to 0 on first run
AD	62	C0	LDA	\$C062	put PB1 status into Accumulator
29	80		AND	#\$80	clear bits 0-6 (just need HIGH vs LOW, bit 7)
C5	EE		CMP	\$EE	compare Accumulator bit 7 with previous PB2 value
D0	02		BNE	\$0363	if PB2 has changed state, store in \$EE
F0	F4		BEQ	\$0357	bit hasn't changed yet, return to wait loop
85	EE		STA	\$EE	
C0	44		CPY	#\$44	
26	EF		ROL	\$EF	
CA			DEX		
D0	E9		BNE	\$0355	
8D	5C	C0	STA	\$C05C	
A5	EF		LDA	\$EF	
60			RTS		

A2	09		LDX	#\$09	reading 8 bits. requires 9 transitions.
A9	00		LDA	#\$00	clear the accumulator
85	EF		STA	\$EF	\$EF is staging for received byte, set to 0
85	EE		STA	\$EE	\$EE is staging for each bit, set to 0
8D	5D	C0	STA	\$C05D	set ANN2 HIGH, indicate to AVR "Clear to Send"
A0	FF		LDY	#\$FF	start wait loop
C8			INY		Increment Y - rolls over to 0 on first run
AD	62	C0	LDA	\$C062	put PB1 status into Accumulator
29	80		AND	#\$80	clear bits 0-6 (just need HIGH vs LOW, bit 7)
C5	EE		CMP	\$EE	compare Accumulator bit 7 with previous PB2 value
D0	02		BNE	\$0363	if PB2 has changed state, store in \$EE
F0	F4		BEQ	\$0357	bit hasn't changed yet, return to wait loop
85	EE		STA	\$EE	store new PB2 state in \$EE
C0	44		CPY	#\$44	
26	EF		ROL	\$EF	
CA			DEX		
D0	E9		BNE	\$0355	
8D	5C	C0	STA	\$C05C	
A5	EF		LDA	\$EF	
60			RTS		

A2	09		LDX	#\$09	reading 8 bits. requires 9 transitions.
A9	00		LDA	#\$00	clear the accumulator
85	EF		STA	\$EF	\$EF is staging for received byte, set to 0
85	EE		STA	\$EE	\$EE is staging for each bit, set to 0
8D	5D	C0	STA	\$C05D	set ANN2 HIGH, indicate to AVR "Clear to Send"
A0	FF		LDY	#\$FF	start wait loop
C8			INY		Increment Y - rolls over to 0 on first run
AD	62	C0	LDA	\$C062	put PB1 status into Accumulator
29	80		AND	#\$80	clear bits 0-6 (just need HIGH vs LOW, bit 7)
C5	EE		CMP	\$EE	compare Accumulator bit 7 with previous PB2 value
D0	02		BNE	\$0363	if PB2 has changed state, store in \$EE
F0	F4		BEQ	\$0357	bit hasn't changed yet, return to wait loop
85	EE		STA	\$EE	store new PB2 state in \$EE
C0	44		CPY	#\$44	if the loop count is more than 68, bit is one. Bit is set
26	EF		ROL	\$EF	
CA			DEX		
D0	E9		BNE	\$0355	
8D	5C	C0	STA	\$C05C	
A5	EF		LDA	\$EF	
60			RTS		

A2	09		LDX	#\$09	reading 8 bits. requires 9 transitions.
A9	00		LDA	#\$00	clear the accumulator
85	EF		STA	\$EF	\$EF is staging for received byte, set to 0
85	EE		STA	\$EE	\$EE is staging for each bit, set to 0
8D	5D	C0	STA	\$C05D	set ANN2 HIGH, indicate to AVR "Clear to Send"
A0	FF		LDY	#\$FF	start wait loop
C8			INY		Increment Y - rolls over to 0 on first run
AD	62	C0	LDA	\$C062	put PB1 status into Accumulator
29	80		AND	#\$80	clear bits 0-6 (just need HIGH vs LOW, bit 7)
C5	EE		CMP	\$EE	compare Accumulator bit 7 with previous PB2 value
D0	02		BNE	\$0363	if PB2 has changed state, store in \$EE
F0	F4		BEQ	\$0357	bit hasn't changed yet, return to wait loop
85	EE		STA	\$EE	store new PB2 state in \$EE
C0	44		CPY	#\$44	if the loop count is more than 68, bit is one. Bit is set
26	EF		ROL	\$EF	rotate the new bit into \$EF, our result byte.
CA			DEX		
D0	E9		BNE	\$0355	
8D	5C	C0	STA	\$C05C	
A5	EF		LDA	\$EF	
60			RTS		

A2	09	LDX	#\$09	reading 8 bits. requires 9 transitions.	
A9	00	LDA	#\$00	clear the accumulator	
85	EF	STA	\$EF	\$EF is staging for received byte, set to 0	
85	EE	STA	\$EE	\$EE is staging for each bit, set to 0	
8D	5D	C0	STA	\$C05D	set ANN2 HIGH, indicate to AVR "Clear to Send"
A0	FF	LDY	#\$FF	start wait loop	
C8		INY		Increment Y - rolls over to 0 on first run	
AD	62	C0	LDA	\$C062	put PB1 status into Accumulator
29	80	AND	#\$80	clear bits 0-6 (just need HIGH vs LOW, bit 7)	
C5	EE	CMP	\$EE	compare Accumulator bit 7 with previous PB2 value	
D0	02	BNE	\$0363	if PB2 has changed state, store in \$EE	
F0	F4	BEQ	\$0357	bit hasn't changed yet, return to wait loop	
85	EE	STA	\$EE	store new PB2 state in \$EE	
C0	44	CPY	#\$44	if the loop count is more than 68, bit is one. Bit is set	
26	EF	ROL	\$EF	rotate the new bit into \$EF, our result byte.	
CA		DEX		decrement X, our bit count	
D0	E9	BNE	\$0355		
8D	5C	C0	STA	\$C05C	
A5	EF	LDA	\$EF		
60		RTS			

A2	09		LDX	#\$09
A9	00		LDA	#\$00
85	EF		STA	\$EF
85	EE		STA	\$EE
8D	5D	C0	STA	\$C05D
A0	FF		LDY	#\$FF
C8			INY	
AD	62	C0	LDA	\$C062
29	80		AND	#\$80
C5	EE		CMP	\$EE
D0	02		BNE	\$0363
F0	F4		BEQ	\$0357
85	EE		STA	\$EE
C0	44		CPY	#\$44
26	EF		ROL	\$EF
CA			DEX	
D0	E9		BNE	\$0355
8D	5C	C0	STA	\$C05C
A5	EF		LDA	\$EF
60			RTS	

reading 8 bits. requires 9 transitions.

clear the accumulator

\$EF is staging for received byte, set to 0

\$EE is staging for each bit, set to 0

set ANN2 HIGH, indicate to AVR "Clear to Send"

start wait loop

Increment Y - rolls over to 0 on first run

put PB1 status into Accumulator

clear bits 0-6 (just need HIGH vs LOW, bit 7)

compare Accumulator bit 7 with previous PB2 value

if PB2 has changed state, store in \$EE

bit hasn't changed yet, return to wait loop

store new PB2 state in \$EE

if the loop count is more than 68, bit is one. Bit is set

rotate the new bit into \$EF, our result byte.

decrement X, our bit count

if bit count is not yet full, loop back to (loopcount)

A2	09		LDX # \$09	reading 8 bits. requires 9 transitions.
A9	00		LDA # \$00	clear the accumulator
85	EF		STA \$EF	\$EF is staging for received byte, set to 0
85	EE		STA \$EE	\$EE is staging for each bit, set to 0
8D	5D	C0	STA \$C05D	set ANN2 HIGH, indicate to AVR "Clear to Send"
A0	FF		LDY # \$FF	start wait loop
C8			INY	Increment Y - rolls over to 0 on first run
AD	62	C0	LDA \$C062	put PB1 status into Accumulator
29	80		AND # \$80	clear bits 0-6 (just need HIGH vs LOW, bit 7)
C5	EE		CMP \$EE	compare Accumulator bit 7 with previous PB2 value
D0	02		BNE \$0363	if PB2 has changed state, store in \$EE
F0	F4		BEQ \$0357	bit hasn't changed yet, return to wait loop
85	EE		STA \$EE	store new PB2 state in \$EE
C0	44		CPY # \$44	if the loop count is more than 68, bit is one. Bit is set
26	EF		ROL \$EF	rotate the new bit into \$EF, our result byte.
CA			DEX	decrement X, our bit count
D0	E9		BNE \$0355	if bit count is not yet full, loop back to (loopcount)
8D	5C	C0	STA \$C05C	if bit count is full, set ANN2 LOW, CTS off
A5	EF		LDA \$EF	
60			RTS	

A2	09	LDX	#\$09	reading 8 bits. requires 9 transitions.	
A9	00	LDA	#\$00	clear the accumulator	
85	EF	STA	\$EF	\$EF is staging for received byte, set to 0	
85	EE	STA	\$EE	\$EE is staging for each bit, set to 0	
8D	5D	C0	STA	\$C05D	set ANN2 HIGH, indicate to AVR "Clear to Send"
A0	FF	LDY	#\$FF	start wait loop	
C8		INY		Increment Y - rolls over to 0 on first run	
AD	62	C0	LDA	\$C062	put PB1 status into Accumulator
29	80	AND	#\$80	clear bits 0-6 (just need HIGH vs LOW, bit 7)	
C5	EE	CMP	\$EE	compare Accumulator bit 7 with previous PB2 value	
D0	02	BNE	\$0363	if PB2 has changed state, store in \$EE	
F0	F4	BEQ	\$0357	bit hasn't changed yet, return to wait loop	
85	EE	STA	\$EE	store new PB2 state in \$EE	
C0	44	CPY	#\$44	if the loop count is more than 68, bit is one. Bit is set	
26	EF	ROL	\$EF	rotate the new bit into \$EF, our result byte.	
CA		DEX		decrement X, our bit count	
D0	E9	BNE	\$0355	if bit count is not yet full, loop back to (loopcount)	
8D	5C	C0	STA	\$C05C	if bit count is full, set ANN2 LOW, CTS off
A5	EF	LDA	\$EF	puts received byte into Accumulator	
60		RTS		return with byte in Accumulator	

You do the Hokey Pokey

and you turn yourself around

demo time.

that's what it's all about

So, What Can It Do?

Read and write to **I²C**, **(SPI)** and **UART** devices (among others), i.e. any Arduino compatible peripheral.

Blinky lights, duh.

Data acquisition, read analog and digital values.

Use case ideas include:

- light indicator for disk activity, RWTS function

- Trackstar-type indicator for bootloaders

- game status indicators - blink on collision, red for damage, etc

- get current internet time via NTP for real time clock function

So, What Can't It Do?

Apple IIc and IIc+

no 16-pin game port, no annunciator pins

Why not use serial?

Apple IIgs

has the 16-pin port, but 2.8mhz

Why not use serial?

Why not use serial?

github.com/option8/gp2io

On the shoulders of giants.

- **Mark Pilgrim** for checking and optimizing my code, and helping out testing the various use cases.
- **David Schmidt** for sanity checking my serial implementation and proof-of-concept.
- **KansasFest committee** for allowing me to present, public beta test the GP2IO at KansasFest 2016.
- **Chris Torrence** and **Roger Wagner** for Assembly Lines, both the book and searchable PDF.
- **Michael Mahon**, for inventing such a nice wheel that I had to go and reinvent it.

Mini Hackfest

The Inaugural, First Annual, and Probably Last, RetroConnector

Mini Hackfest

Judging criteria

Judging criteria

- Is it cool?

Judging criteria

- Is it cool?
- Does it do something the Apple II couldn't do before?

Judging criteria

- Is it cool?
- Does it do something the Apple II couldn't do before?
- Is it **really** cool?

Judging criteria

- Is it cool?
- Does it do something the Apple II couldn't do before?
- Is it **really** cool?
- Originality.

Judging criteria

- Is it cool?
- Does it do something the Apple II couldn't do before?
- Is it **really** cool?
- Originality.
- No, seriously. How cool is **that**? I'm angry I didn't think of it myself.