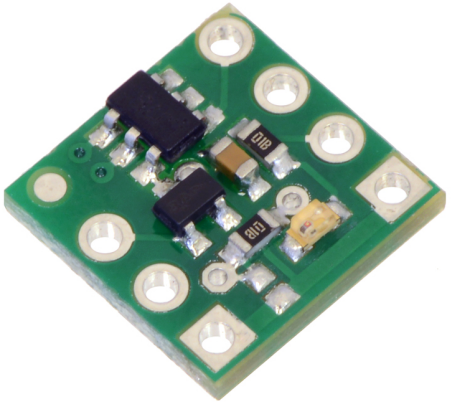
RPicSim

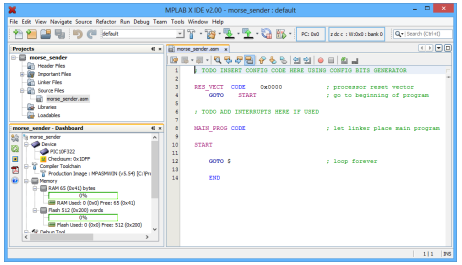
David Grayson
Las Vegas Ruby Meetup
2014-02-12

https://github.com/pololu/rpicsim

The RPicSim gem provides an interface to the MPLAB X PIC simulator that allows you to write simulator-based automated tests of PIC firmware using Ruby and RSpec.



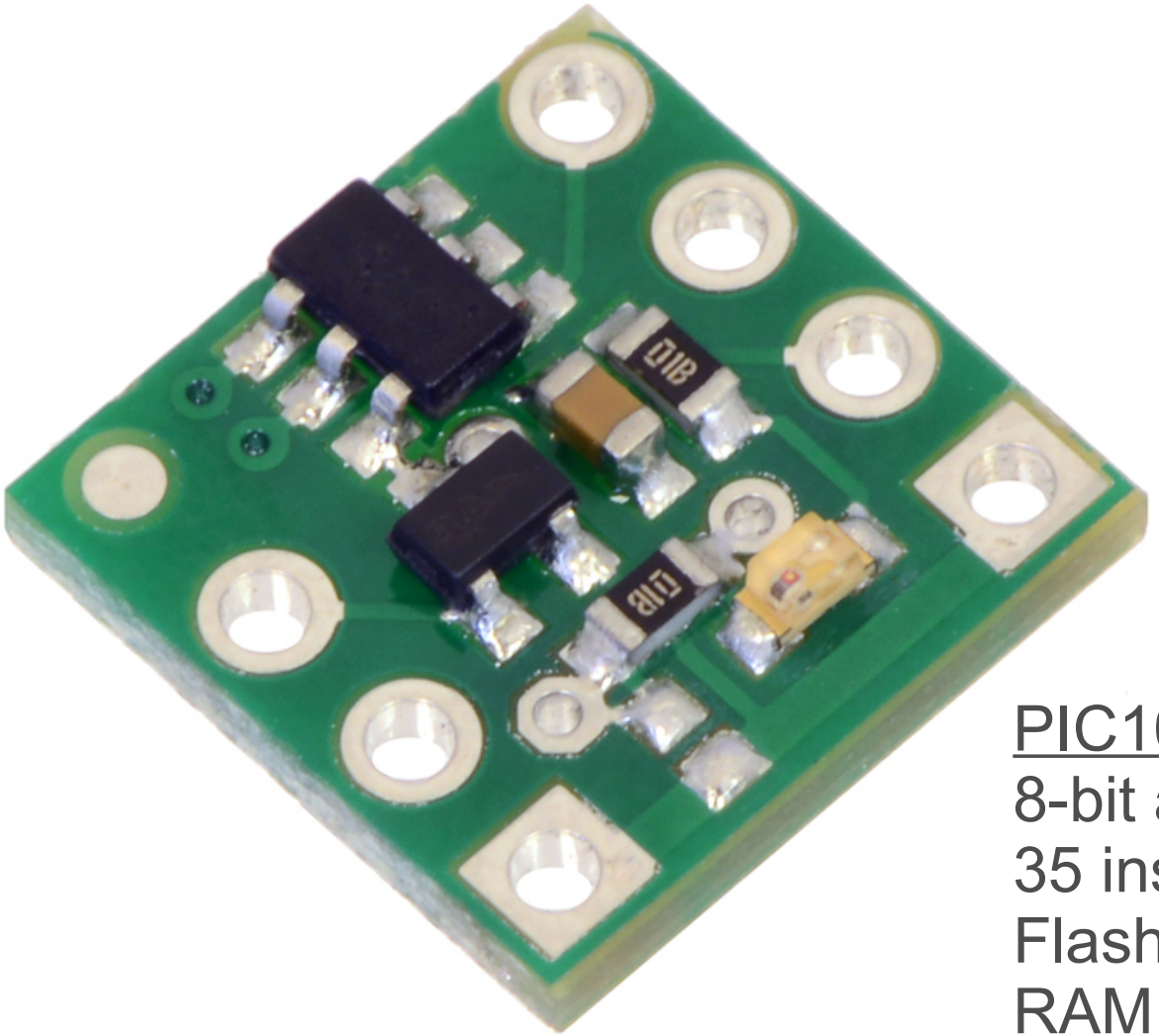
PIC microcontrollers



PIC development tools

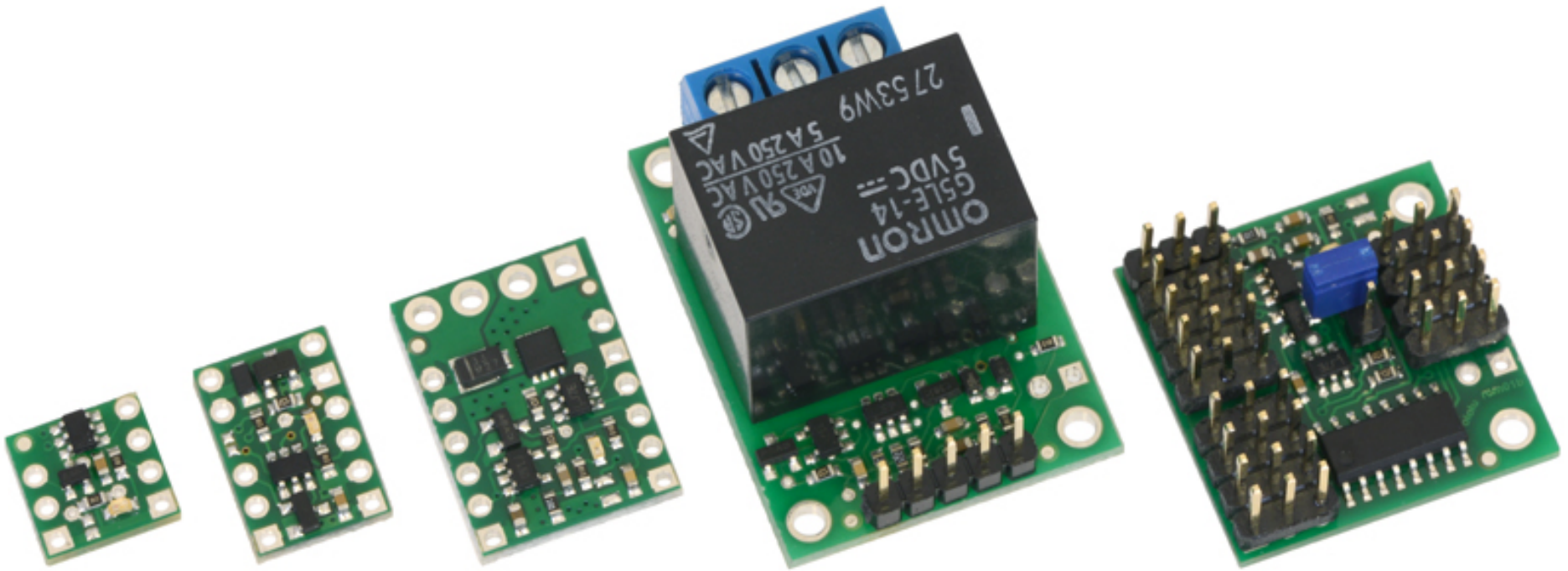
RPicSim

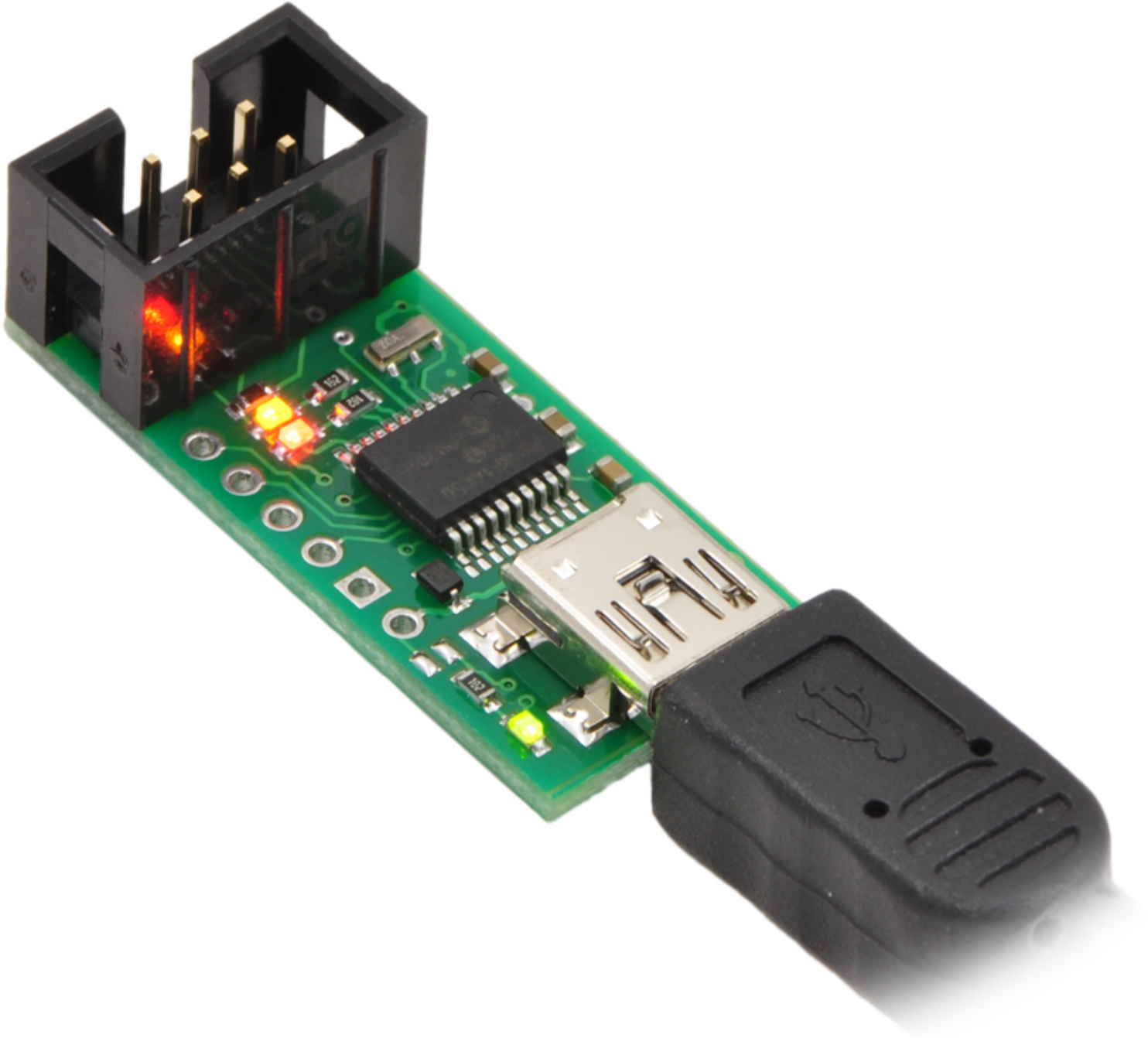
PIC microcontrollers

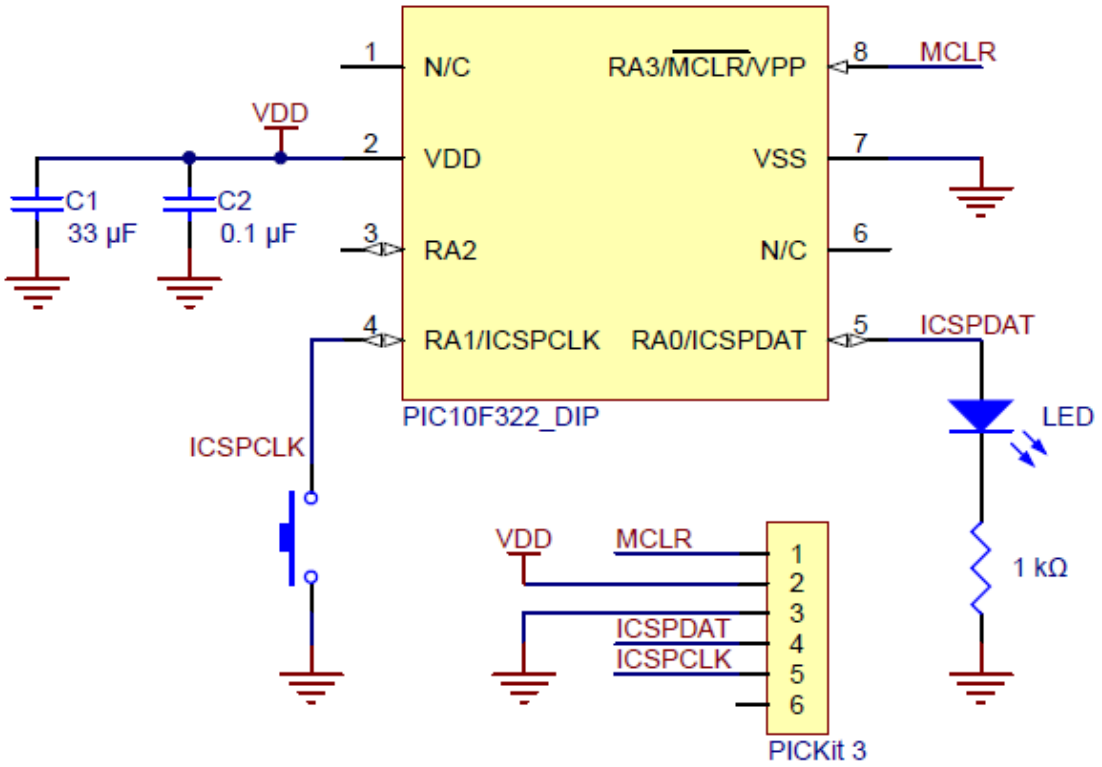


PIC10F322
8-bit architecture
35 instructions
Flash: 512 words
RAM: 64 bytes

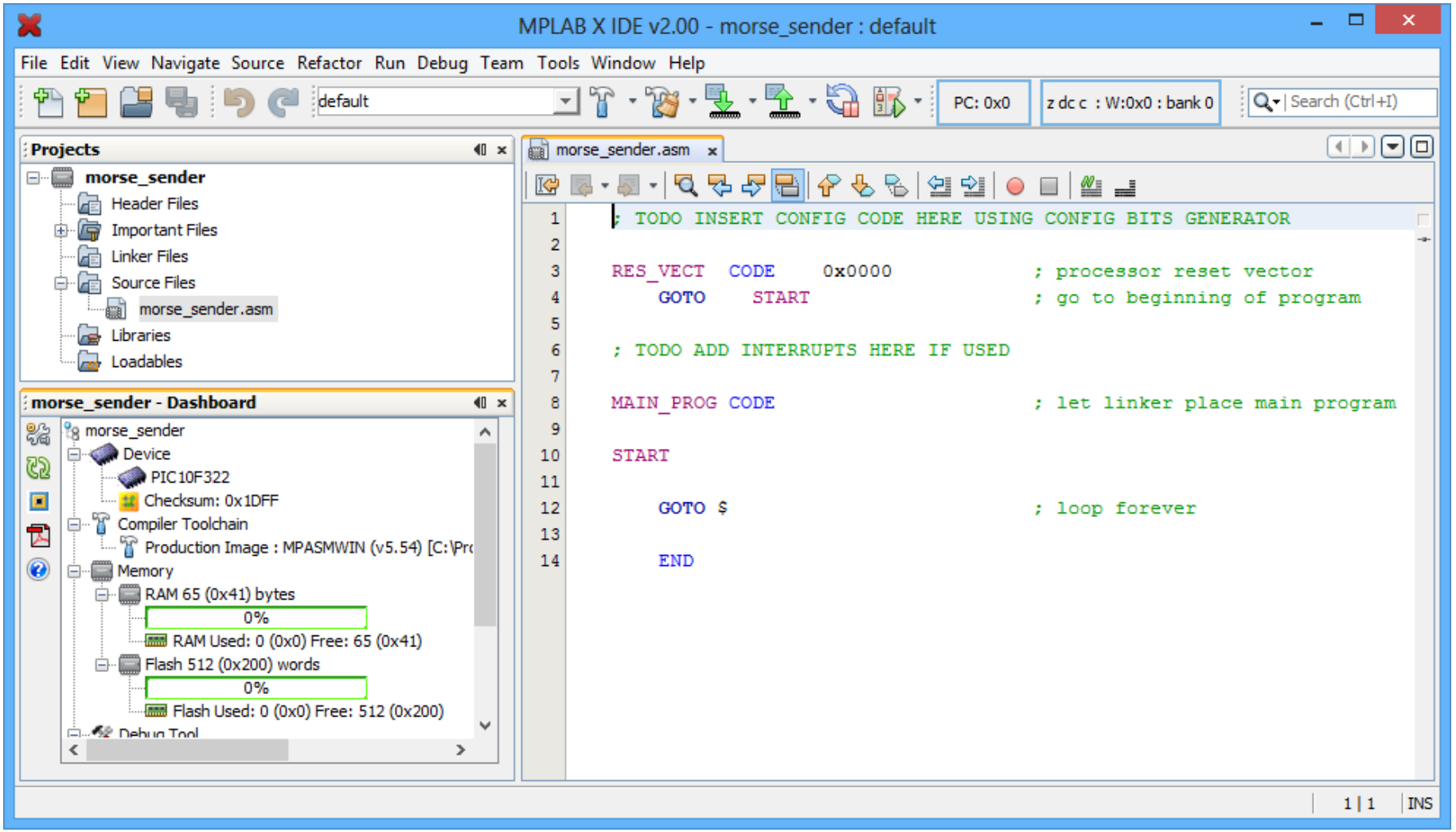
http://www.pololu.com/product/2801


OMRON
G5LE-14
5VDC
10 A 250 V AC
5 A 250 V AC
27 53W9



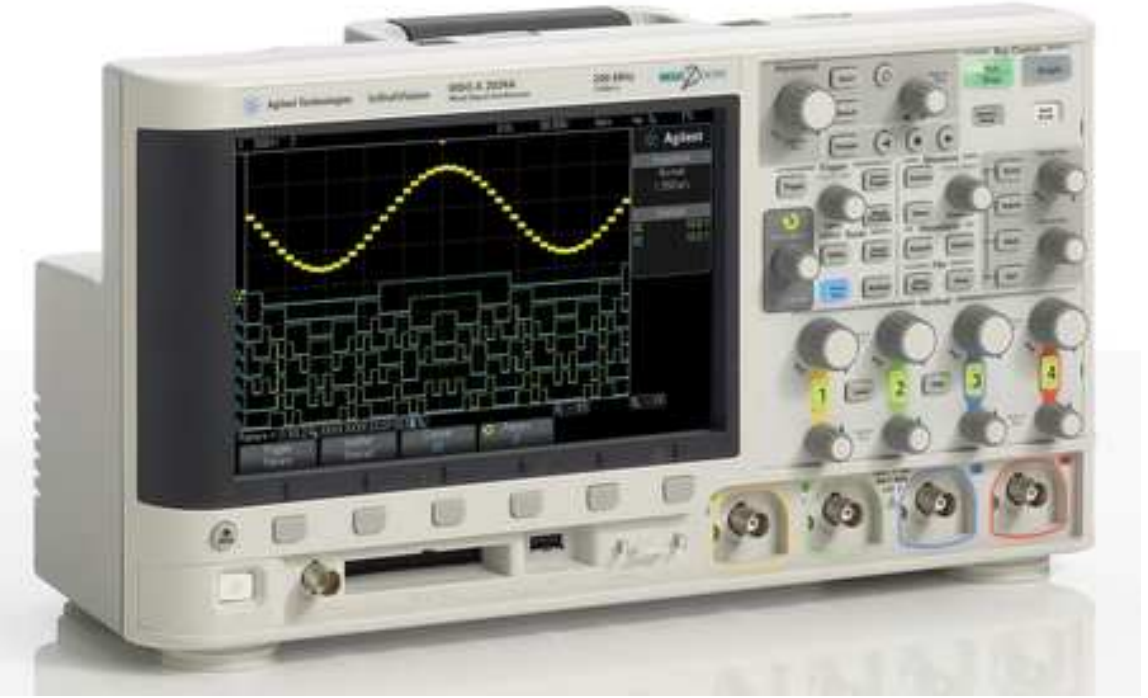

1
N/C
RA3/MCLR/VPP
8
MCLR
VDD
2
VDD
VSS
7
C1
33 µF
C2
0.1 µF
3
RA2
N/C
6
4
RA1/ICSPCLK
RA0/ICSPDAT
5
ICSPDAT
PIC10F322_DIP
LED
ICSPCLK
VDD
MCLR
1
2
3
4
5
6
ICSPDAT
ICSPCLK
1 kΩ
PICKit 3

PIC development tools



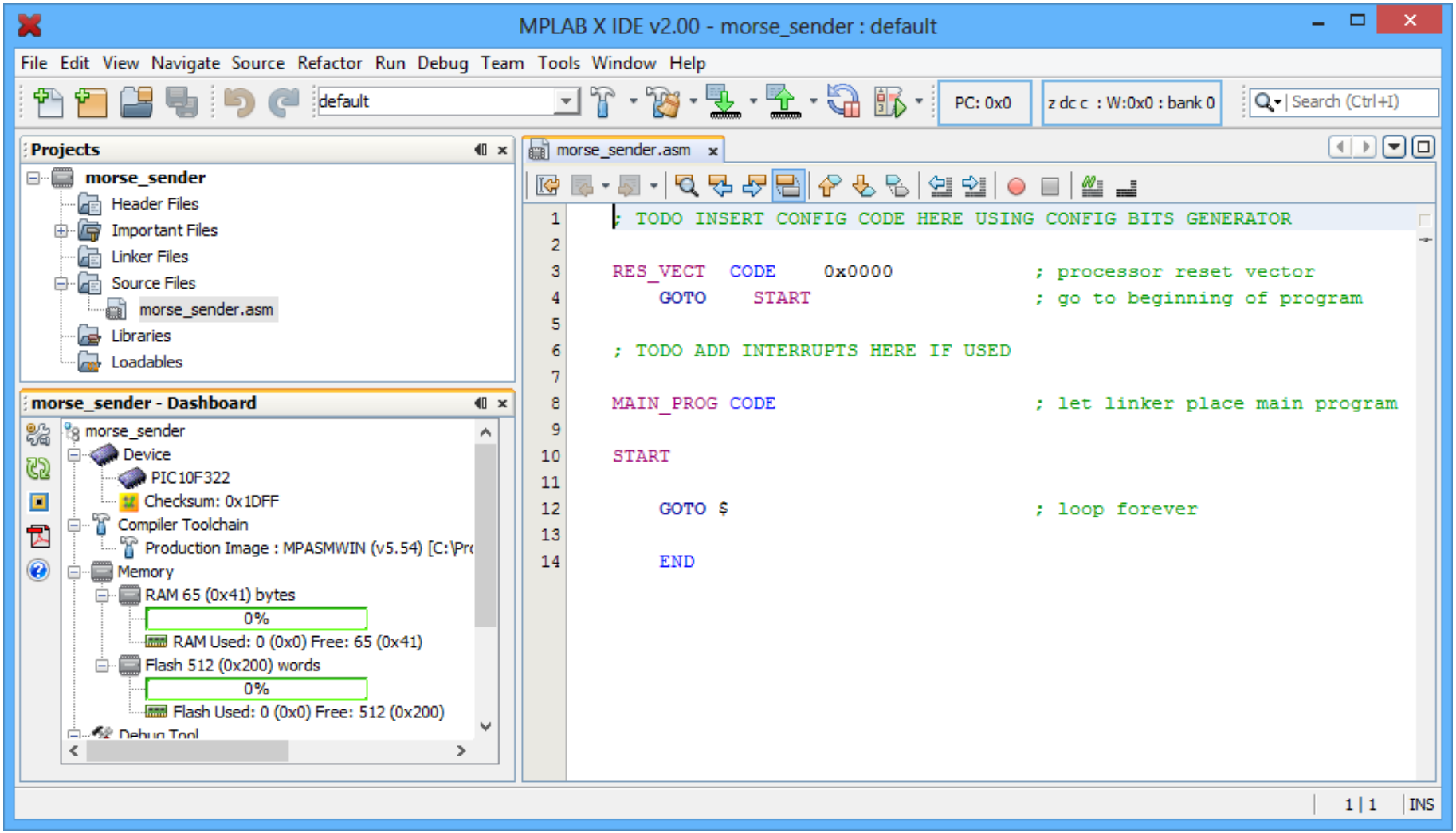
Debugging firmware

- Oscilloscope
- Dummy code to toggle I/O lines



http://www.zorotools.com/g/00073078/k-G3702447

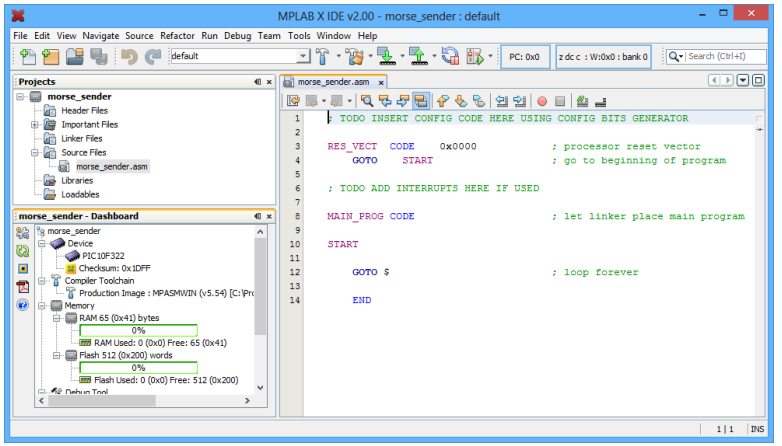
MPLAB X Simulator



RPicSim

The RPicSim gem provides an interface to the MPLAB X PIC simulator that allows you to write simulator-based automated tests of PIC firmware using Ruby and RSpec.

MPLAB X IDE



MPLAB X Simulator
(jar files)

Simulating I/O

```
it "continuously mirrors" do
  main_input.set false
  run_cycles 10
  expect(main_output).to be_driving_low

  main_input.set true
  run_cycles 10
  expect(main_output).to be_driving_high
end
```

Persistent expectations

```
expect(main_output).to be_driving_low
```

```
expecting main_output => be_driving_low
```

Unit test

```
it 'adds 70 to 22' do
  addend1.value = 70
  addend2.value = 22
  run_subroutine :addition, cycle_limit: 100
  expect(sum.value).to eq 92
end
```

Unit test with RAM watcher

```
it 'adds 70 to 22' do
  addend1.value = 70
  addend2.value = 22
  step; ram_watcher.clear
  run_subroutine :addition, cycle_limit: 100
  expect(sim.ram_watcher.writes).to eq({sum: 92})
end
```

Setup

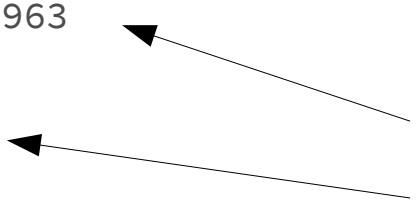
```
require 'rpicsim/rspec'

class MySim < RPicSim::Sim
  device_is 'PIC10F322'
  filename_is File.dirname(__FILE__) + '../src/dist/firmware.cof'

  # pin aliases
  # variable definitions
  # helper methods
end
```

```
describe 'some part of the firmware' do
  before do
    start_sim MySim
  end

  # examples
end
```

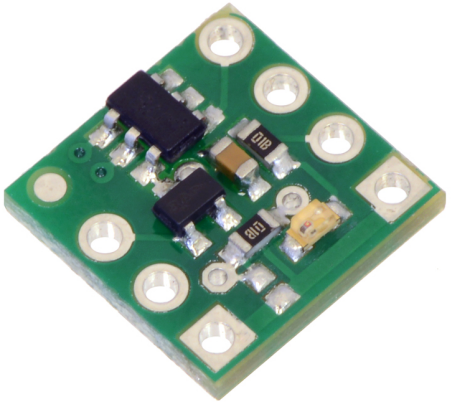
Stubs

```
@foo_calls = []
every_step do
  if pc.value == label(:foo).address
    @foo_calls << { a: foo_param_a.value,
                    b: foo_param_b.value }
    sim.return
  end
end
```

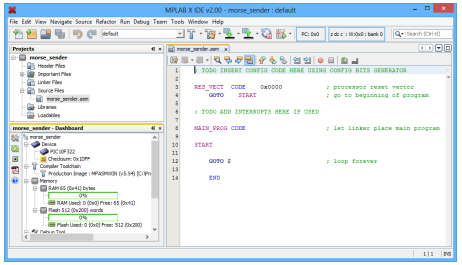
```
.............................................F.....

Failures:

  1) FooWidget when exposed to 1.5 ms pulses behaves correctly
     Failure/Error: run_microseconds 1500
       expected INTCON to satisfy block
     # ./lib/rpicsim/rspec/persistent_expectations.rb:29:in `check_expectations'
     # ./lib/rpicsim/rspec/persistent_expectations.rb:27:in `check_expectations'
     # ./lib/rpicsim/rspec/helpers.rb:25:in `start_sim'
     # ./lib/rpicsim/sim.rb:574:in `step'
     # ./lib/rpicsim/sim.rb:716:in `run_to_cycle_count'
     # ./lib/rpicsim/sim.rb:708:in `run_cycles'
     # ./spec/foo_widget_spec.rb:10:in `(root)'

     Simulation cycle count: 78963

     Simulation stack trace:
     0x01A0 = startMotor
     0x0044 = motorService+0x14
     0x0B12 = mainLoop+0x2
     0x008C = start2

Finished in 4.55 seconds
44 examples, 1 failure

Failed examples:

rspec ./spec/example/nice_error_spec.rb:8 # FooWidget when exposed to 1.5ms pulses behaves
correctly
```



Useful error messages



PIC microcontrollers



PIC development tools

RPicSim

For more info:
https://github.com/pololu/rpicsim

TABLE 23-2: PIC10(L)F320/322 **INSTRUCTION SET**

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb			LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C, DC, Z	1, 2
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z	1, 2
CLRF	f	Clear f	1	00	0001	lfff	ffff	Z	2
CLRW	–	Clear W	1	00	0001	0xxx	xxxx	Z	
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z	1, 2
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z	1, 2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00	1011	dfff	ffff		1, 2, 3
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z	1, 2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00	1111	dfff	ffff		1, 2, 3
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z	1, 2
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z	1, 2
MOVWF	f	Move W to f	1	00	0000	lfff	ffff		
NOP	–	No Operation	1	00	0000	0xx0	0000		
RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	C	1, 2
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C	1, 2
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C, DC, Z	1, 2
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff		1, 2
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	1, 2
BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff		1, 2
BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff		1, 2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		3
LITERAL AND CONTROL OPERATIONS									
ADDLW	k	Add literal and W	1	11	111x	kkkk	kkkk	C, DC, Z	
ANDLW	k	AND literal with W	1	11	1001	kkkk	kkkk	Z	
CALL	k	Call Subroutine	2	10	0kkk	kkkk	kkkk		
CLRWDT	–	Clear Watchdog Timer	1	00	0000	0110	0100	$\overline{TO}$, $\overline{PD}$	
GOTO	k	Go to address	2	10	1kkk	kkkk	kkkk		
IORLW	k	Inclusive OR literal with W	1	11	1000	kkkk	kkkk	Z	
MOVLW	k	Move literal to W	1	11	00xx	kkkk	kkkk		
RETFIE	–	Return from interrupt	2	00	0000	0000	1001		
RETLW	k	Return with literal in W	2	11	01xx	kkkk	kkkk		
RETURN	–	Return from Subroutine	2	00	0000	0000	1000		
SLEEP	–	Go into Standby mode	1	00	0000	0110	0011	$\overline{TO}$, $\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11	110x	kkkk	kkkk	C, DC, Z	
XORLW	k	Exclusive OR literal with W	1	11	1010	kkkk	kkkk	Z	

TABLE 2-2: PIC10(L)F320/322 MEMORY MAP (BANK 0)

INDF(*)	00h	PMADRL	20h		40h		60h
TMR0	01h	PMADRH	21h				
PCL	02h	PMDATL	22h				
STATUS	03h	PMDATH	23h				
FSR	04h	PMCON1	24h				
PORTA	05h	PMCON2	25h				
TRISA	06h	CLKRCON	26h				
LATA	07h	NCO1ACCL	27h				
ANSELA	08h	NCO1ACCH	28h				
WPUA	09h	NCO1ACCU	29h				
PCLATH	0Ah	NCO1INCL	2Ah				
INTCON	0Bh	NCO1INCH	2Bh				
PIR1	0Ch	Reserved	2Ch				
PIE1	0Dh	NCO1CON	2Dh				
OPTION_REG	0Eh	NCO1CLK	2Eh	General Purpose Registers 32 Bytes		General Purpose Registers 32 Bytes	
PCON	0Fh	Reserved	2Fh				
OSCCON	10h	WDTCON	30h				
TMR2	11h	CLC1CON	31h				
PR2	12h	CLC1SEL1	32h				
T2CON	13h	CLC1SEL2	33h				
PWM1DCL	14h	CLC1POL	34h				
PWM1DC	15h	CLC1GATE1	35h				
PWM1CON	16h	CLC1GATE2	36h				
PWM2DCL	17h	CLC1GATE3	37h				
PWM2DC	18h	CLC1GATE4	38h				
PWM2CON	19h	CWG1CON0	39h				
IOCAP	1Ah	CWG1CON1	3Ah				
IOCAN	1Bh	CWG1ASD	3Bh				
IOCAF	1Ch	CWG1RC	3Ch				
FVRCON	1Dh	CWG1FC	3Dh				
ADRES	1Eh	VREGCON	3Eh				
ADCON	1Fh	BORCON	3Fh		5Fh		7Fh

Legend: = Unimplemented data memory locations, read as '0'.

* = Not a physical register.

Simplified PIC development steps