P. O. Box 462101 Garland,Tx 75046
(972) 272-9392 Fax: (972) 494-5814 Email: sales@tech-tools.com

In-Circuit Programming with QuickWriter™

V1.3

This is an application note describing how to design your target for proper in-circuit programming with a QuickWriter.

THE'IDEALS'

The IDEAL programming environment varies widely, depending on whose perspective you take.

Programming hardware's Ideal

From the Programming hardware's perspective, the ideal setup is a chip in a socket, with nothing else attached to the chip. This gives the programming hardware full control over the process, eliminating any unknowns.

The next best setup is a chip in a target board with all of its signals isolated from the target during programming.

Firmware Developer's Ideal

The firmware developer's ideal would be to ignore programming issues. He would not want to give up any pins or make any concessions to accommodate in-circuit programming. In-circuit or in-socket is a production issue.

Hardware Developer's Ideal

The hardware developer would like to simply attach a programming connector to the appropriate signals (or even clip a connector over the chip) and forget about it. He would like to use all of the signals in the target without worrying about what the programmer might need.

THE REALITY

In-circuit programming is more complicated than in-socket programming. It is possible to use a programmer that was not designed for in-circuit programming if the hardware designer takes extreme measures to make the device appear to be in-socket. In other words, completely isolate the device from the target during programming. Although this is possible, it is rarely done. The reality is that NONE of the IDEALs are ever realized.

In-circuit programming support must be designed into the programming hardware AND the target hardware.

IMPLEMENTATION

QuickWriter is designed to make In-Circuit programming as hassle free as possible. It allows the hardware and firmware developers to come as close to their ideals as possible and still do reliable in-circuit programming. QuickWriter includes several features to support this.

High Dynamic and Static Drive

QuickWriter includes high dynamic current and static current capabilities on MCLR and VCC. This allows it to over-come capacitance and loads on these circuits.

Inter-locked VPP and VCC enable/disable

QuickWriter monitors the voltage on the MCLR and VCC pins, allowing it to adapt to different target loads. Each time QuickWriter enables or disables either of these signals, it monitors its voltage to ensure it has risen or fallen fully before continuing to the next phase of the programming algorithm. This interlock is crucial to reliable programming with a wide range of power, load and capacitance variations in the target.

Target Monitoring

QuickWriter checks for error conditions and complains if it sees indications that the target is not properly designed to do in-circuit programming safely. For example, it verifies that it can pull MCLR low before starting. A failure here indicates a wiring problem or a very low value pull-up on MCLR. QuickWriter verifies that when it turns on VCC, it rises to 5V +- .5 V within a reasonable amount of time. A failure here would indicate improper wiring, excessive capacitance on VCC or excessive loading on VCC. Similar checks are made when enabling MCLR. Additionally, QuickWriter verifies that when MCLR is enabled, it does not drive VCC above its nominal level, indicating that MCLR is too tightly coupled to VCC or that there is a wiring error.

Passive MCLR/VPP and VCC pull-downs

The MCLR and VCC drivers use active HIGH DRIVE but passive LOW DRIVE. This allows the programmer to safely evaluate the target's suitability for in-circuit programming before it applies ANY power to the board. It also permits monitoring of interaction between the two power rails.

DESIGNING FOR IN-CIRCUIT PROGRAMMING

In addition to the in-circuit support built into QuickWriter, there are still a few basic requirements on the target to make it in-circuit programmable. The following defines the key issues related to each signal involved in programming. The numbers in () are the in-circuit programming cable pin numbers. Don't let the lengthy discussion put you off; it really is pretty straightforward and most targets require very little effort to make them in-circuit programmable. This discussion just seams a little lengthy because we are providing the reasons for the requirements as well as the requirements themselves. There is a condensed 'design recommendations' summary at the end.

MCLR/VPP (1)

HIGH VOLTAGE

This signal is driven to over 13V during programming. Your target must be able to tolerate this high voltage on this signal. The PIC® MCU is designed to accept this voltage of course. However, anything else attached to this pin probably will be damaged. You will need to isolate the MCLR/VPP line from anything else in your target that could be damaged by the programming voltages. Possible devices include a buffer (used to drive the reset to other devices), a power supply monitor/reset chip or a logic output used to generate the reset in lieu of the standard RC setup.

Other Drivers

If you use logic to reset the chip in normal operation (rather than depend on the RC), then these drivers must be isolated unless they can tolerate > 13V AND they do not actively drive (HIGH or LOW) the MCLR line during programming.

Pull-ups

Any pull-up on MCLR must be 10K or larger. This prevents the programming voltage from forcing the target power supply above its nominal 5V. It also allows QuickWriter to passively pull-down MCLR to a reset state. If you have a smaller (stronger) pull-up, QuickWriter's internal 1K pull-down will not be able to pull MCLR to a valid low. QuickWriter will notice this and complain. If you have an existing target with a strong pull-up and changing it to 10K is not an option, you can compensate by strengthening QuickWriter's pull-down proportionally. For example, if your pull-up is 4.7K (about 2x as strong), adding a 1K pull-down between MCLR and ground on QuickWriter's ICSP cable will double our pull-down strength.

Capacitors

Although capacitors are generally not used on PICmicro® MCU reset circuits, but they can be and sometimes are used. QuickWriter can tolerate up to 10uF of capacitance from MCLR to ground. The voltage interlocks will wait until the capacitor is fully charged/discharged before continuing. NOTE that this cap must be rated for > 13V!

Devices with Internal MCLR

Some devices require that MCLR be driven BEFORE VCC. This is very unusual and damaging to most chips. For these devices, the programmer will drive 13V to the MCLR pin before applying power to the target. If you have anything at all connected to MCLR, you should evaluate its tolerance to 13V **EVEN WHEN IT IS NOT POWERED**. For these devices, you will want to diode isolate MCLR from VCC so that MCLR does not attempt to power the VCC rail. Generally, place a diode in series with the MCLR pull-up (anode to VCC) or completely isolate MCLR from VCC with a jumper.

VCC (2)

Who powers the target?

It is possible to do in-circuit programming with the programmer powering the target or with the target powering itself. There are trade-offs and limitations with each choice.

Programmer Powered

The programmer can provide up to 100ma of power to power up your target during programming. This gives it complete control over programming. However, there are a few things to watch for.

Back-feeding Regulators

Most power regulators regulate by adjusting how much current they DRIVE into the circuit. Few are capable of sinking current. They depend on the load itself to sink current. Since they are not designed to sink current, many regulators can be damaged if their OUTPUT becomes higher than their INPUT pin. This is exactly what happens if you allow the programmer to power the target and do not power the input to the regulator. Most 3-terminal regulator design guides specify if the regulator is tolerant of this condition or how to design to make it tolerant.

Brown-outs

It is possible that the programmer can provide enough current to power up the target until actual programming begins. At that time, the additional power required for programming could cause the entire target board voltage to drop and the programming to fail. Alternatively, the average voltage could stay valid, but noise could appear on VCC in response to surge current requirements by the circuit of device being programmed. Either symptom would be an indication that the board requires more than the 100ma of current the programmer is capable of supplying.

Partially powered targets

If you allow the programmer to back-feed the VCC power rail on the target, that would power up VCC, but may leave other power rails on the target un-powered. This could be damaging to some components.

For example, if you have a 12 volt power supply powering some devices and this is normally regulated down to 5V for the PICmico MCU and others, then the 12V devices would not get powered when the programmer drives current into the VCC (5V) rail. This could cause 5V to be driven into some un-powered devices.

Similar problems can occur even in 5V only systems if they have multiple isolated power rails.

Capacitance

QuickWriter can drive up to 1000uF on the VCC rail. If there is more capacitance than this, QuickWriter will report that VCC failed to rise fast enough and will abort programming.

The larger the VCC capacitance, the longer it takes for them to bleed down after programming is complete, depending on the target load. Large capacitance could increase programming times. If this becomes a problem, you can add a bleed resister between VCC and GROUND on the ICSP cable to speed up the bleed-down. Keep the VCC load under 100ma (target + bleed resister.)

Self Powered

This is often the best choice. It eliminates any concern over back-feeding power supplies or about multi-voltage targets being partially powered. It most closely mimics the target's intended operational modes. However, there is one situation in which programmer powered is better.

Internal MCLR

We normally use MCLR to hold the chip in reset until we are ready to enter programming mode. At that time, we quickly drive MCLR to VPP levels. To properly enter programming mode, we must drive MCLR to VPP before the oscillator has run more than a few dozen cycles. Otherwise, the device begins to run and advances the program counter before we get into programming mode. The result is that we start programming beyond the start of code space, causing a failure.

Some devices have the option of configuring them for internal MCLR, freeing up a pin for general I/O usage. However, once the pin is programmed as an I/O pin instead of a MCLR function, we can no longer hold the chip in reset. It simply starts running as soon as it is powered. This makes it much more difficult to ensure we get MCLR up to VPP levels before the PC starts advancing, particularly during in-circuit programming.

Microchip solved this 'catch-22' by changing the programming specs for these devices so that we apply VPP BEFORE VCC. This forces the chip into programming mode even if its MCLR function has been internalized, but ONLY if we can control VCC as well as VPP.

For these devices, it is desirable to allow the programmer to control VCC so that it can properly enter programming mode, even if the device has been previously programmed to use internal MCLR. You can further increase the probability of success by enabling the device's power up timer. This causes the device to wait longer before starting to execute code.

If other limitations preclude powering the target from the programmer, you could still program factory fresh devices. You just may not be able to re-program them or to do another verify AFTER the fuses are programmed. During normal programming, we program and verify everything but the fuses and THEN program the configuration fuses. The configuration fuses are the last things we program so we maintain control over MCLR until the very last operation. Once the fuses are programmed to internalize MCLR, we would loose control over the device.

Another possibility might be to use the programmer's VCC to control a small relay that actually powers the target. Make sure the relay coil uses less than 100ma at 5VDC and that you provide a reverse biased diode across the coil to absorb any inductive 'kick' when the relay is opened.

GND (3)

Always connect the programmer's ground to the target's ground.

DATA/RB7(4) and CLOCK/RB6 (5)

These signals are driven by PICmicro MCU port pin with a small (~33 Ohm) series resister.

Other drivers

You must isolate any other drivers from these signals so that the programmer can drive them during programming, They can be isolated via jumpers or by designing your circuit such that they are tri-stated during programming. The easiest way to deal with this is to use these pins for OUTPUTs in the actual circuit. This way, you won't have to do anything special for in-circuit programming.

Loads

If these lines have excessive loads like LEDs or especially capacitive loads, they should be isolated during programming. If the rise and fall times of these signals are affected by the target, they could become skewed enough to cause violations of the programming timing specifications.

Key (6)

This is a keying plug. There is no wire connected to this location in the connector.

LVPEN (7)

This signal is driven to ground during programming by a PICmicro MCU port pin. Some devices have a LVPEN pin that must be held inactive during programming. This is the RB3 pin on the 16F87x and 16F62x devices. You can avoid connecting this signal if your circuit has a pull-down on this pin or otherwise drives it low during programming. Of course, if the device you are programming does not have a LVPEN pin, you would not need to connect this signal.

You could also use this pin as a control input to logic on your target board. This could be used to control tri-state enables or transmission gates to isolate signals during programming. Simply put a pull up on the target. When programming begins, the signal is pulled low, effecting the isolation.

Other drivers

See the DATA & CLOCK section.

Loads

See the DATA & CLOCK section.

Target Design Recommendations

- Use a 10K Ohm or larger pull up on MCLR Use 10uF or less capacitance on MCLR (most targets use none).
- Avoid or isolate any other circuitry that touches the MCLR line.
- Ideally, use RB6,RB7 and RB3 as OUTPUTs in your circuit design OR connect them to outside-world connections or NO switches so that nothing is driving them during programming. This avoids any need for isolation. If that is not desirable, design to isolate or tri-state anything that drives these lines.
- Always connect the ground pin
- Decide who powers the target and address the trade-offs involved.
- Enable the PICmicro MCU's power-up timer if possible. Particularly if using a device with internal MCLR.