Firmware Scan Functional Description
Dudley B Hiller
Uwharrie Test Solutions LLC
10/04/11

Method for verifying electrical connectivity and testing for stuck-at conditions of the GPIO signals of most microcontrollers using a state machine driven by specialized firmware, a two signal interface and a digital In-Circuit Tester.

1.0 Scope

This document will describe the functionality of a manufacturing test process called Firmware Scan.

2.0 Abstract

Firmware Scan is an electronics test process for the purpose of testing pin connectivity of GPIO pins of a microcontroller to a Printed Circuit Board Assembly (PCBA). This process will also test all of the GPIOs as inputs for stuck-at conditions.

The Firmware Scan process will work generically on all microcontrollers from all manufacturers, except that it may not be necessary or worthwhile on devices that already have testability functions built into the part.

Firmware Scan serves the same purpose as Boundary Scan IEEE 1149.1, except that it will function on microcontrollers that do not have the IEEE 1149.1 electronics embedded into the hardware of the part. Firmware Scan serves the same purpose as a test technique known as NAND Tree, except that it does not require probe access to every input in order to maintain its sequence, and does not require NAND-Tree hardware embedded in the device. Firmware Scan serves the same purpose as capacitive vector-less test techniques known as Test Jet^{tm}, Opens Xpress^{tm} and FrameScan^{tm}, except that, Firmware Scan operates on a powered device, it will additionally test for stuck-at conditions, and it does not require any special test fixture electronics.

The driver program (a State Machine) for Firmware Scan is loaded into the program memory of the microcontroller, usually into Flash Memory, but sometimes into Static Random Access Memory (SRAM) or One Time Programmable Read Only Memory (OTP ROM). The Firmware Scan driver is usually loaded into program memory by a digital In-Circuit Tester, but may also be pre-programmed into the part at another station. The Firmware Scan program is then executed by a Digital In-Circuit Tester and any failures are diagnosed and reported by the Tester.

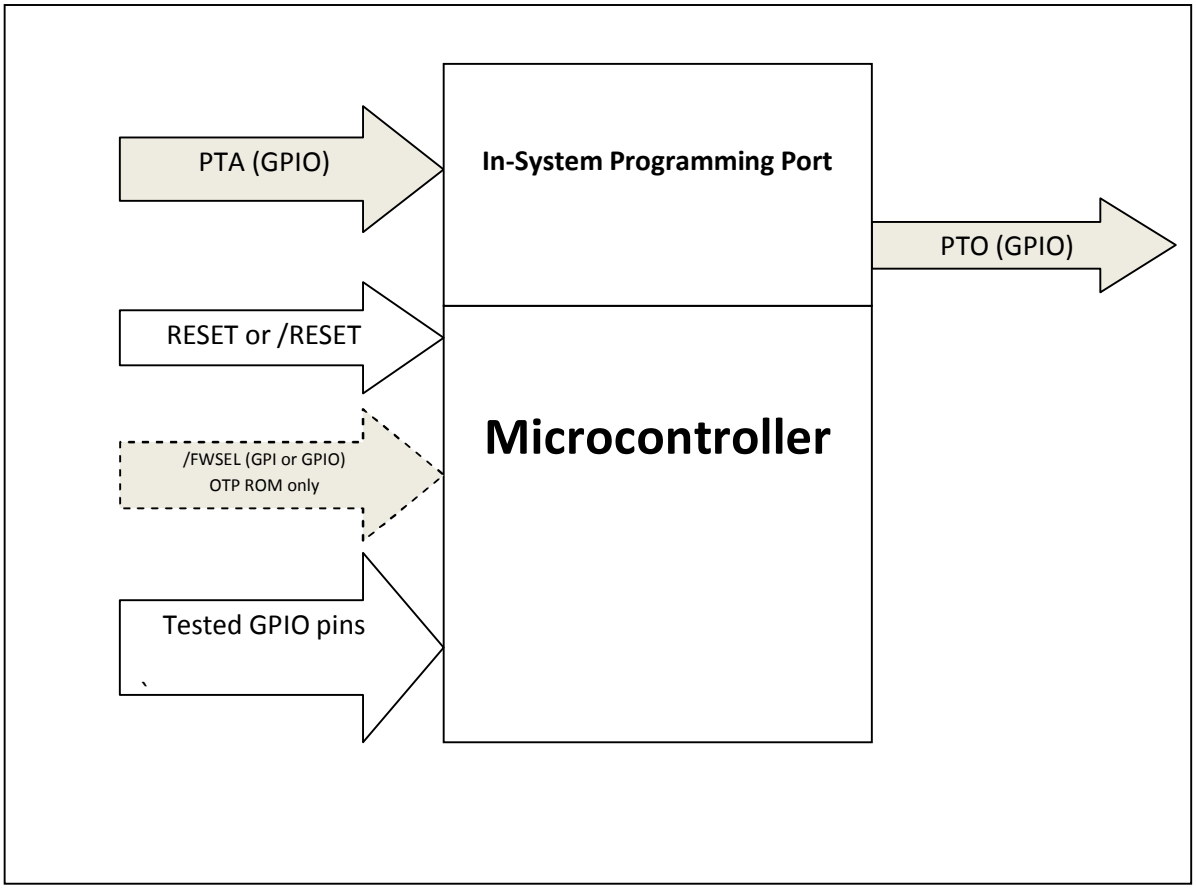
Firmware Scan execution requires the use of only two signals, one input (for control) and one output (for results), aside from the dedicated RESET input of the microcontroller, plus the pin(s) to be tested. In the case of microcontrollers with OTP ROM one additional input is required to select the Firmware Scan program to run instead of the normal user program.

The function of the pins used for Firmware Scan, are programmed temporarily into the memory, and therefore can share the same pins as other user functions. With the exception of microcontrollers with OTP ROM, Firmware Scan does not require any special Design for Testability (DFT) except for the normal rules that must be followed to accommodate In-System Programming (ISP).

Firmware Scan can only test microcontroller pins that can be read directly through a register in the microcontroller's address apace. These are pins known as General Purpose Input-Output (GPIO). All of these GPIO pins are tested as input-only for stuck-at conditions. GPIOs typically make up more than 90% of the pin set of most microcontrollers on the market today. Most every special function pin of most microcontrollers can also serve as a GPIO.

3.0 Signal Description

Shaded areas indicate the signals unique to Firmware Scan.



3.1 RESET or /RESET Input

This is the dedicated reset input of the microcontroller. It may be active high or low. This represents the reset input of any microcontroller. It is not unique to Firmware Scan, and only mentioned to describe its function for starting the Firmware Scan State Machine. RESET is just the generic name given to this signal. It is also called MCLR and RST among other names. This pin is driven by the In-Circuit Tester.

3.2 Pin Test Advance (PTA) Input

This signal is driven by the in-Circuit Tester. A transition from high to low or low to high will advance the Firmware Scan State Machine to begin testing the next pin in the sequence. This signal uses one of the GPIO pins.

3.3 Pin Test Out (PTO) Output

This signal will output the same state that is input on one of the tested GPIO pin. This signal uses a GPIO pin. This pin is received and sensed by the In-Circuit Tester.

3.4 Firmware Scan Select not (/FWSEL) Input

This pin, if pulled low at reset will cause the microcontroller program to execute the Firmware Scan Driver. Otherwise the microcontroller will execute the user code for normal operation. This signal is only used on microcontrollers that have their firmware driver loaded into OTP ROM. This pin can be any GPIO. This pin must be connected to a pull-up resistor to assure that it is always high at reset.

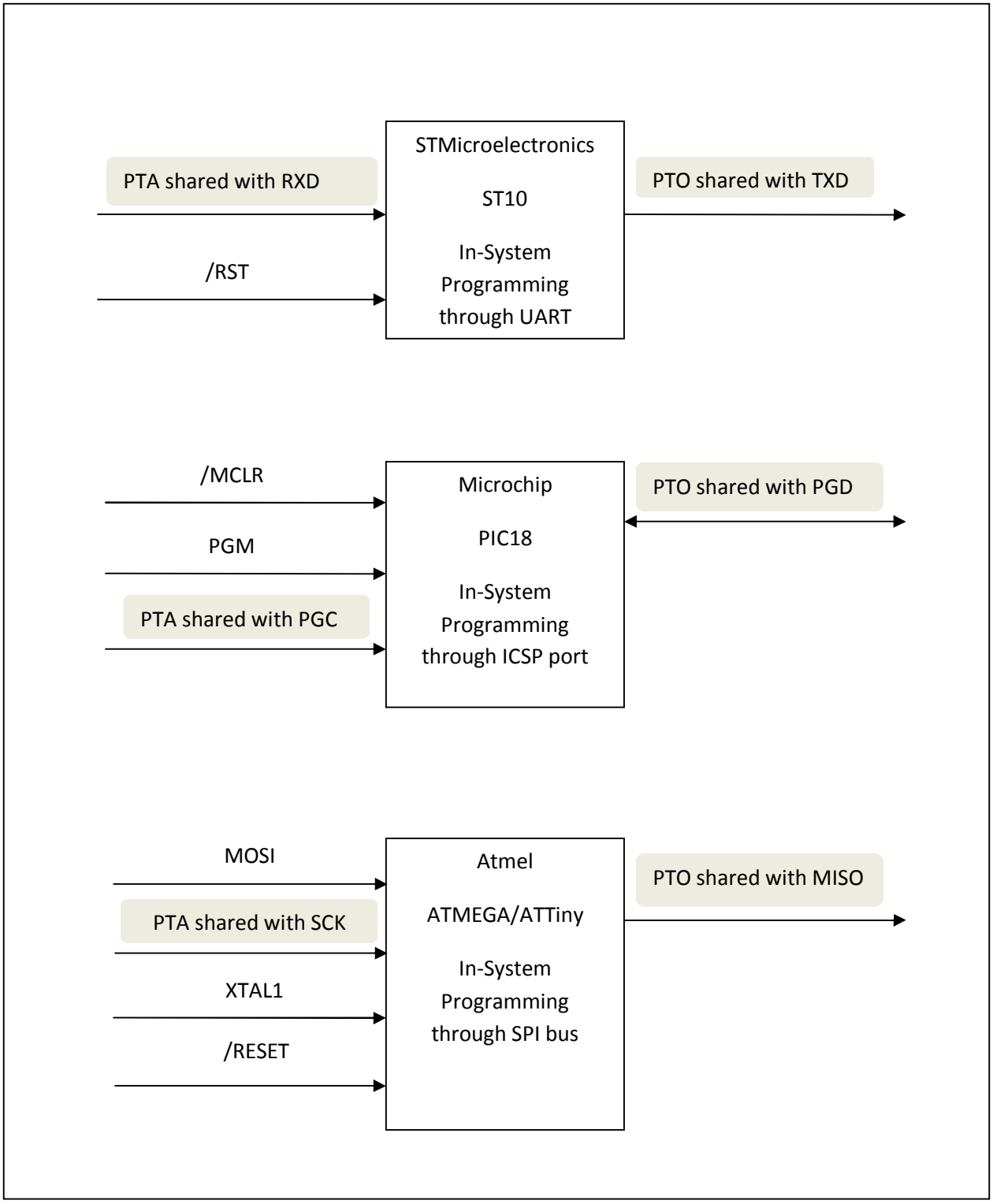
3.5 GPIO tested input pins

These are the pins tested for connectivity and stuck-at conditions. These pins are driven by the In-Circuit Tester.

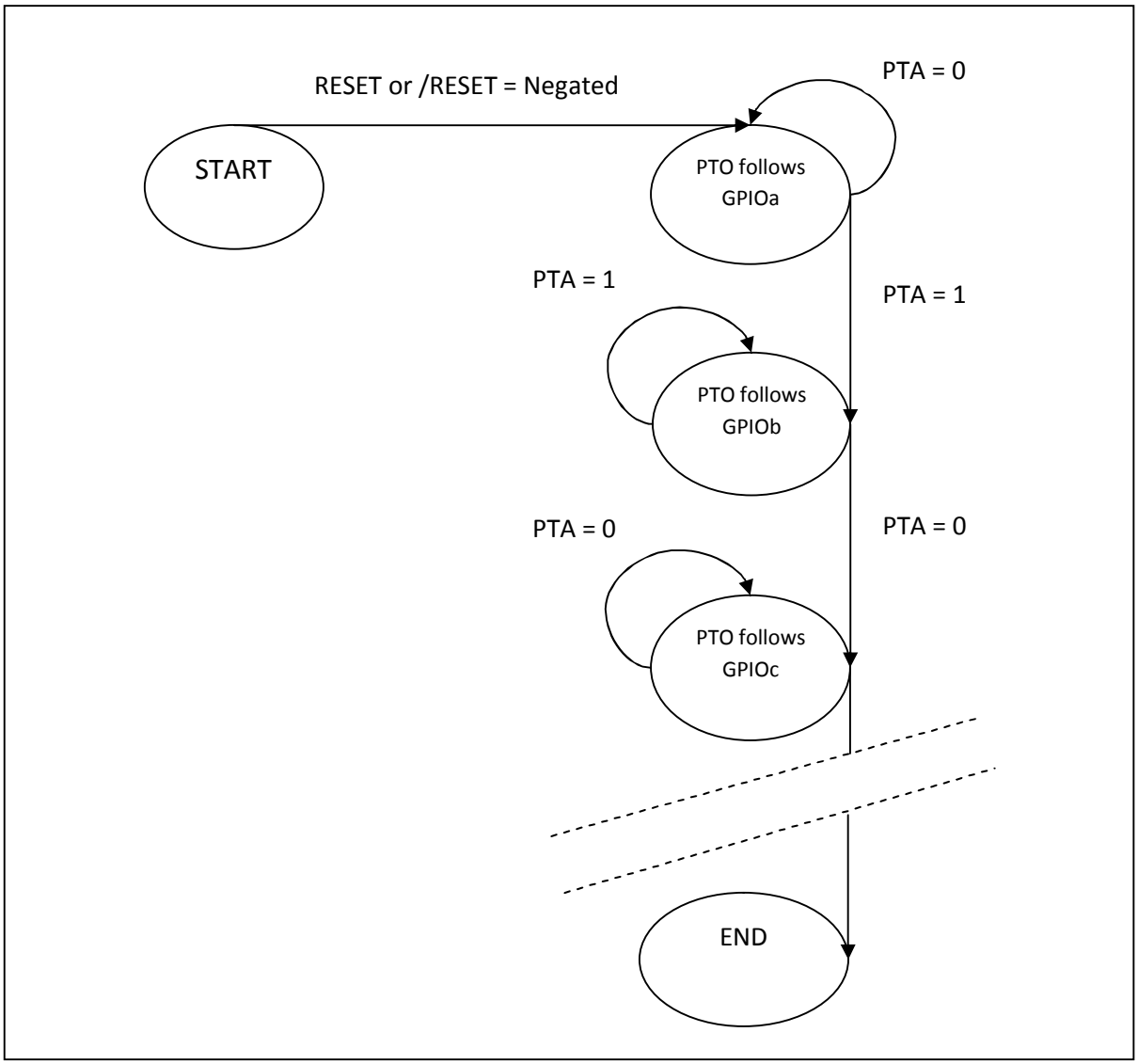
3.6 Signal sharing with the In-System Programming port of the Microcontroller.

The PTA and PTO signals, which are GPIOs, are strategically sharing the same pins as signals used by the In-System Programming port of the microcontroller. This is done so that Firmware Scan does not conflict with any functional signal of the user's design. If the PCBA was designed to allow In-System Programming, it will allow Firmware Scan as well and therefore have no signal conflicts.

These drawings show how PTA and PTO share the same pin with other logic signals of the In-System Programming port for three microcontroller examples, from three different manufacturers. Shaded areas show the signals used by Firmware Scan, all others are existing pins on the microcontroller.



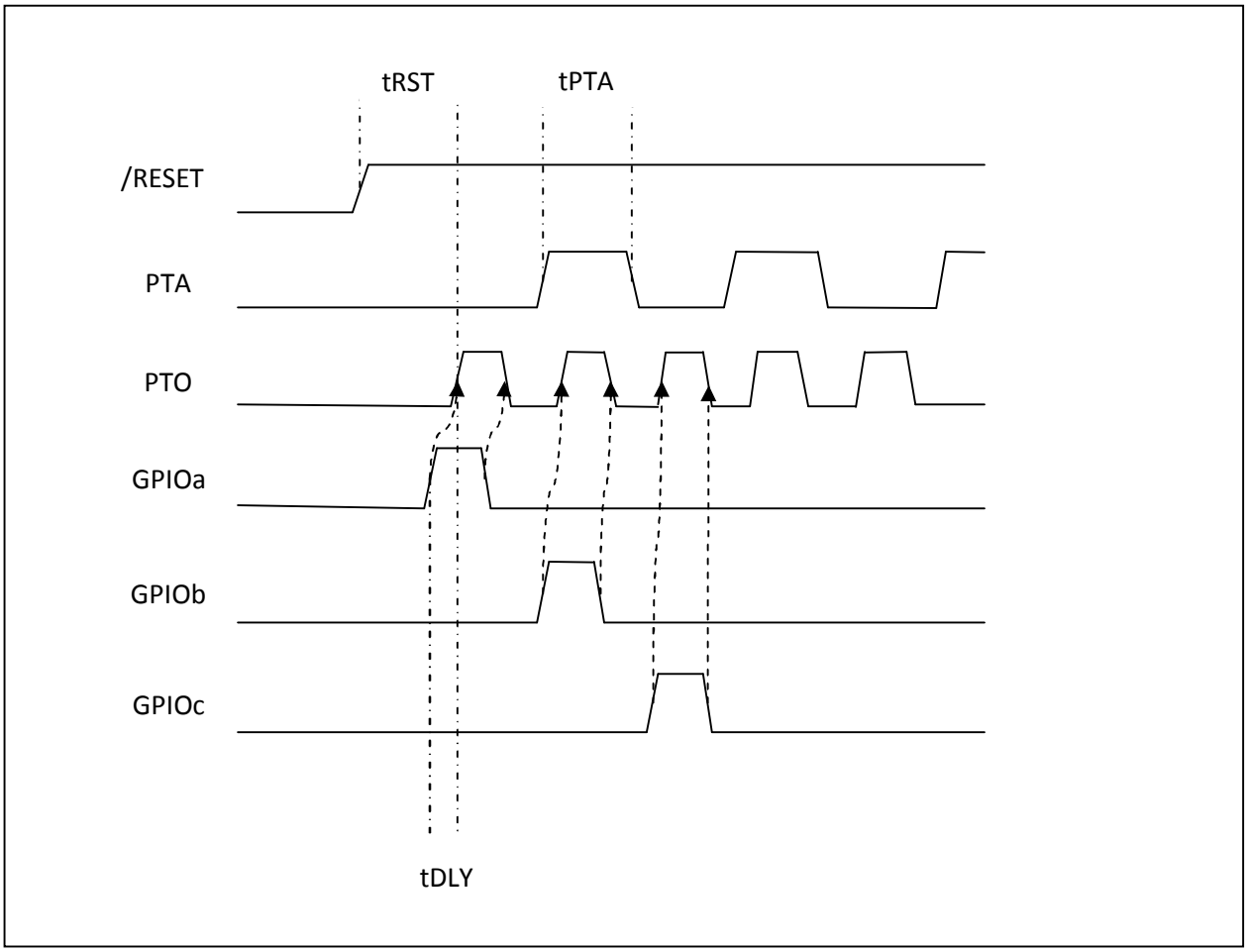
4.0 Firmware Scan State Machine



When /RESET is negated, the State Machine advances immediately, or as soon as possible to the first state. At this time the In-Circuit Tester can drive GPIOa and expect the PTO output to follow. The microcontroller remains in this state until the In-Circuit tester drives a logic-high to the PTA input. At this state the In-Circuit Tester can drive GPIOb and expect the PTO output to follow. This sequence continues until all of the GPIO pins are tested.

The State Machine was designed to be controlled by PTA alone. Unlike NAND Tree it is not necessary to control the tested GPIOs to maintain the sequence. This makes Firmware Scan more adaptable for designs that cannot provide probe access to every pin.

5.0 Signal Timing



4.1 tRST

This is the delay time from the release of reset to the beginning of the functionality of the state machine. This time varies depending on the initialization time and the clock speed of the microcontroller being tested.

4.2 tPTA

This is the minimum high and low time of the PTA signal. This minimum depends on the efficiency of the Firmware Scan driver code and the clock speed of the microcontroller. tPTA has no maximum time.

4.3 tDLY

This is the delay time from the transition of a GPIO to the subsequent transition of the PTO signal. This delay time depends on the efficiency of the Firmware Scan driver code and the clock speed of the microcontroller.

5.0 Firmware Driver

Because the State Machine must ideally respond in an instantaneous manner to changes in the level of the PTA and tested GPIO pins, the best language for the code of the firmware driver is the Assembly Language of the microcontroller. Even so, there will always be a delay from a GPIO transition to a PTO transition, or a PTA transition to the advance of the State Machine. This is typically a few microseconds.

The Firmware Scan Driver code will always execute immediately and unconditionally after reset. No interrupts are ever used. No PUSH, POP, CALL or RETurn instructions are used. A stack is never defined or used. All code is straight in-line code except for loop-backs to maintain the current state until the correct PTA condition is met.

Firmware Scan functions most efficiently on Microcontrollers that have a single bit move command.

Some initialization is required before the start of the state machine to assure that the GPIO pin that serves as PTO is programmed to be an output and all other GPIOs are inputs. It is also sometimes necessary to shut down certain functions of the microcontroller so the GPIOs will function as generic GPIOs.

The following code examples are firmware scan drivers.

Firmware Scan Driver example for PIC18

```
EQU PTA PORTB,RB6
EQU PTO PORTB,RB7
EQU GPIOA PORTA,RA0
EQU GPIOB PORTA,RA1
EQU GPIOC PORTA,RA2
EQU GPIOD PORTA,RA3

INIT:
    MOVLW 070h     ; Switch to 8-mhz CPU clock
    MOVWF OSCCON
                   ; Initialize all ports
    CLRF PORTA     ; Clear output data latches
    CLRF PORTB
    CLRF PORTC
    CLRF PORTD
    CLRF LATD
    CLRF PORTE
    MOVLW 0Fh      ; Configure A/D
    MOVWF ADCON1   ; For digital input
    MOVLW 07h      ; Configure comparators
    MOVWF CMCON    ; for digital input
    MOVLW 07Fh     ; Set RB<6:0> as inputs, RB7 as output
    MOVWF TRISB
                   ; All other TRIS registers default to 1, as inputs
START:
TEST_GPIOA:
    BTFSS GPIOA
    BCF PTO        ; Set PTO low if test pin is low
    BTFSC GPIOA
    BSF PTO        ; Set PTO high if test pin is high
    BTFSS PTA      ; Advance to next test pin if PTA is high
    BRA GPIOA
TEST_GPIOB:
    BTFSS GPIOB
    BCF PTO        ; Set PTO low if test pin is low
    BTFSC GPIOB
    BSF PTO        ; Set PTO high if test pin is high
    BTFSC PTA      ; Advance to next test pin if PTA is low
    BRA TEST_GPIOB
TEST_GPIOC:
    BTFSS GPIOC
    BCF PTO        ; Set PTO low if test pin is low
    BTFSC GPIOC
    BSF PTO        ; Set PTO high if test pin is high
    BTFSS PTA      ; Advance to next test pin if PTA is high
    BRA TEST_GPIOC
TEST_GPIOD:
    BTFSS GPIOD
    BCF PTO        ; Set PTO low if test pin is low
    BTFSC GPIOD
    BSF PTO        ; Set PTO high if test pin is high
    BTFSC PTA      ; Advance to next test pin if PTA is low
    BRA TEST_GPIOD
                                        ; Test the rest of the GPIOs
```

Generic C example of Firmware Scan Driver

```
BOOL PTA, PTO;
BOOL GPIOa, GPIOb, GPIOc, GPIOd, GPIOe;

main(void)
    {

  // TEST_GPIOa
  while(PTA = 0)   // Until PTA is low
    {
    PTO = GPIOa;   // Copy GPIOa to PTO
    }

  // TEST_GPIOb
  while(PTA = 1)   // Until PTA is high
    {
    PTO = GPIOb;   // Copy GPIOb to PTO
    }

  // TEST_GPIOc
  while(PTA = 0)   // Until PTA is low
    {
    PTO = GPIOc;   // Copy GPIOc to PTO
    }

  // TEST_GPIOd
  while(PTA = 1)   // Until PTA is high
    {
    PTO = GPIOd;   // Copy GPIOd to PTO
    }

  // TEST_GPIOe
  while(PTA = 0)   // Until PTA is low
    {
    PTO = GPIOe;   // Copy GPIOe to PTO
    }
}
```