0. Introduction

The latest family of PIC microcontrollers are able to write to their own program memory space. This allows a small bootloader program to receive new program code over the SPI bus (which is normally used to communicate with the M&C bus).

1. Implementation

A small bootloader routine resides in a protected area in the top 256 words of program memory and the reset vector contains a jump to the start of this routine. The first action of the routine is to examine the SPI chip select line. If this line is inactive the bootloader jumps to the start of normal code. If the line is active the bootloader reads new code from the SPI bus and reprograms the memory. This process is explained in detail in the next section.

Whilst new code is being loaded the PIC must be regarded as unusable and a flag at the very top of memory is set to indicate this state. The flag is only reset when the full program has been satisfactorily loaded and will be left set if the programming sequence fails for any reason. If the bootloader finds that the flag is set it will not allow normal program execution to start.

2. Limitations

Three limitations must be pointed out. If the unprotected code at the reset vector (three words because of the need to page switch) becomes corrupt the scheme bcomes unworkable and the device must be reprogrammed in the normal way. Also the main program code must always start at a fixed location. This should not usually be a problem as the fixed location can be regarded as an alternative restart vector and be programmed with a jump. Finally it is not possible to reprogram the EEPROM data memory.

3. Bootloader process

The routine waits until the chip select line is deasserted then enables the SPI system. It then sends NULL bytes until it receives an RDY byte in response. If the RDY byte is not received within a few seconds it is assumed that the PIC is disconnected or broken and the programming attempt is abandoned.

The programming device then sends a single record, in the format described below, without any handshaking. Having received the entire record satisfactorily, without any data overruns, and stored it in RAM, the bootloader evaluates the CRC. If this is correct it programs the block, a process that can take ten milliseconds per word.

The programming device polls the routine by sending NULL bytes which will be returned unchanged until the programming is complete or has been aborted. The routine will return a code either indicating that the programming has completed successfully or indicating the nature of the fault.

This continues until a special last transfer record is received, after which the bootloader will stop execution. The last transfer record is characterised by having a special header.

4. Error Checking

The bootloader checks on each byte that there has been no data overrun. It checks that the CRC is correct. It checks that the number of words in each record is equal to that given in the header. This latter check requires two implementation details. First: the chip select line must be active during the entire block, not activated on a byte-by-byte basis. Second: the chip select line must be connected to another input pin on the PIC so that it can be read by the program. If these conditions cannot be met the length checks cannot be made. Even without those checks a correct CRC gives a very high level of confidence that the transfer was successful.

The programming device must make the following checks before sending the record. The block length is no more than 32 words. The address for each word must be greater than 0x03 and less than 0x1F00 (16F876/7) or 0x0F00 (16F873/4) or 0x700 (16F870/1). Each data word is not greater than 0x3FFF.