

Building and Using the MSC1210 Versatile Programmer

Hugo Cheung

Data Acquisition Products—Microsystems

ABSTRACT

The MSC1210 analog-to-digital converter (ADC) [1] is embedded with flash memory (Flash) sizes ranging from 32K bytes (MSC1210Y5) down to 4K bytes (MSC1210Y2). Programming the on-chip Flash is necessary during code development and manufacturing, as well as in application. The MSC1210 provides convenient serial programming operation for in-system programming (ISP), and also provides high-speed parallel programming operation for manufacturing. The MSC1210 built-in boot ROM provides Flash access routines to support Flash access with a simple user interface for in-application programming (IAP). This application report discusses various Flash programming setups and provides a standalone solution for their implementation: the MSC1210 Versatile Programmer (MVP). This programmer operates in both slower serial programming configurations (for low-quantity production programming or code development purposes), as well the high-speed parallel Flash programming configuration (for higher volume production). This report also discusses the MVP schematic, PCB layout, and software implementation details.

Contents

Features	3
Overview	4
Block Diagram	5
Usage	7
TI MSC1210 Downloader.....	7
Operation Screen	8
Hardware.....	10
Power Supply	10
Power Control for the Slave CPU.....	10
RS232 Transceiver	10
Reset and PSEN Control	11
Programming the Master CPU	11
Parallel Programming Interface.....	12
Parallel Bus	13
LED and Key I/O.....	13
Slave/Aux CPU Clock	13
Slave CPU Protection	13
Firmware	14
Interrupt Service Routines	14
Command Parser.....	16
Parallel Access Function.....	17
System Testing.....	18
Other Supported Configurations.....	19
ISP Configurations.....	19
Gang Serial Configuration.....	19
Gang Parallel Programming Configuration.....	20
Appendix A: MSC1210 Versatile Programmer Schematic	21
References.....	22

Figures

Figure 1. MSC1210 Flash Memory System	4
Figure 2. MSC1210 Versatile Programmer Block Diagram	5
Figure 3. MVP Board (Parallel Programming Configuration).....	5
Figure 4. MVP Board (Serial Programming Configuration)	6
Figure 5. TI DownLoader Installation Screen	7
Figure 6. Start-Up Screen (H for Help)	8
Figure 7. Menu Screen	9
Figure 8. Serial Flash Programming Power-On Timing (EA is ignored)	11
Figure 9. Parallel Programming Handshaking and Timing.....	12
Figure 10. Tx and Rx FIFO	15
Figure 11. MVP ISP Configuration.....	19
Figure 12. ISP Configuration Connector	19
Figure 13. MVP Gang Serial Configuration.....	19
Figure 14. MVP Gang Parallel Configuration	20
Figure 15. MVP Gang Parallel System with Five MVP Boards.....	20
Figure 16. MSC1210 Versatile Programmer Schematic	21

COPYRIGHT © Texas Instruments. All rights reserved.

The information in this document is subject to change without notice.

MSC1210 is trademark of Texas Instruments.

Other brands and product names are trademarks of their respective owners.

Features

- Single MSC1210 device programmer
- Reconfigurable for serial programming or parallel programming
- Hot-remove and hot-insert for devices being programmed
- Serial Programming
 - 55 seconds to download 32KB program code (Intel hex format) at 57600 baud
 - User interface: PC running TI-Downloader S/W (PC COM port terminal program)
- Parallel Programming
 - Onboard 32KB+128B Flash data memory buffer for downloading Intel hex data
 - Functions: download, verify, erase, set security, device check, and byte edit
 - 10 seconds to program 32KB from buffer
 - 55 seconds to download 32KB program code from PC to buffer at 57600 baud
 - Standalone operation:
 - Onboard status LED and speaker, or
 - Using TI Downloader software

Overview

MSC1210 [1] Flash programming requires low-level hardware support to control Flash. The Flash control hardware interfaces with the on-chip boot ROM program (see Figure 1).

There are three Flash programming modes (FPMs) for the device. Serial FPM (lower right of Figure 1) is enabled by setting PSEN to a logic 0 and ALE to a logic 1 during device reset. Serial port 0 is used to communicate with the programming host (for example, a PC). Flash programming commands such as Flash Erase, Flash Read, and Flash Write are available through the serial port interface. Intel hex format is supported for downloading the target program. Reference [2] describes the available serial FPM commands.

Parallel FPM (lower left of Figure 1) is enabled by setting PSEN to logic 1 and ALE to logic 0. The device under programming is interfaced to the host through a parallel bus that includes the programming address bus, programming data bus, command bus, and handshaking lines. Reference [2] describes the available parallel FPM commands and parallel programming handshake timing.

When PSEN and ALE are not driven with any logic levels during reset, the device is set to user application mode (UAM). UAM allows the application program to erase and read/write the data Flash, as well as code Flash.

This application report describes the details of implementing the MSC1210 Versatile Programmer, a standalone system that can program the device using the serial programming configuration or parallel programming configuration.

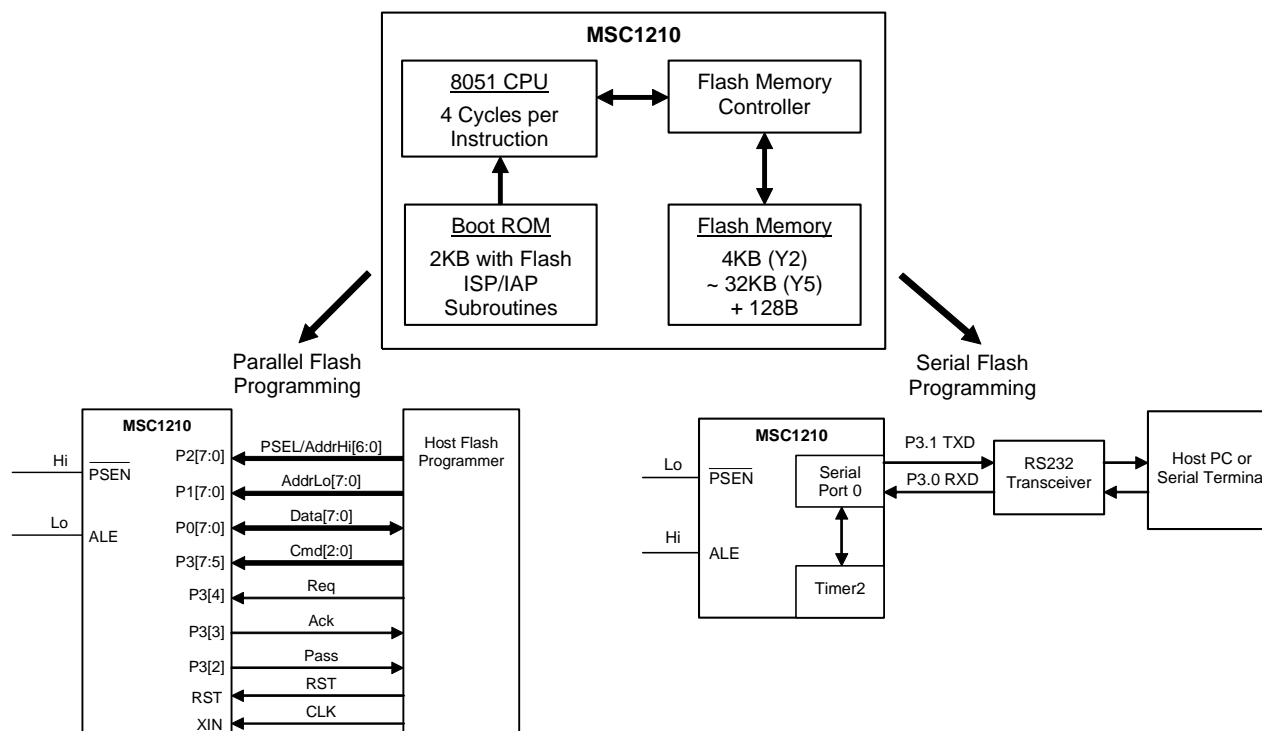


Figure 1. MSC1210 Flash Memory System

Block Diagram

Figure 2 shows the block diagram for the MVP board; Figure 3 is a picture of the board (parallel programming configuration). The MVP programs a single MSC1210 device at a time. It can be configured with DIP switches to use either the MSC1210 parallel or serial flash programming mode.

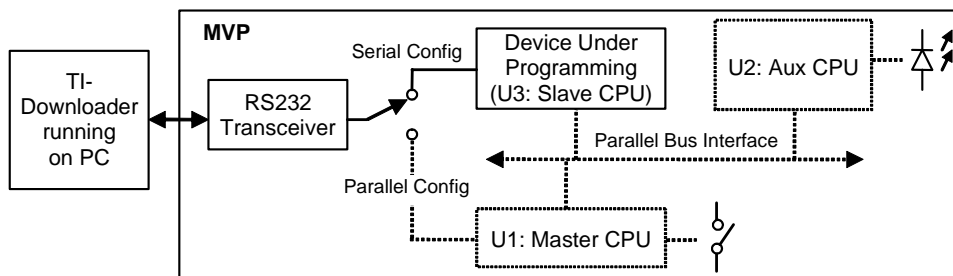


Figure 2. MSC1210 Versatile Programmer Block Diagram

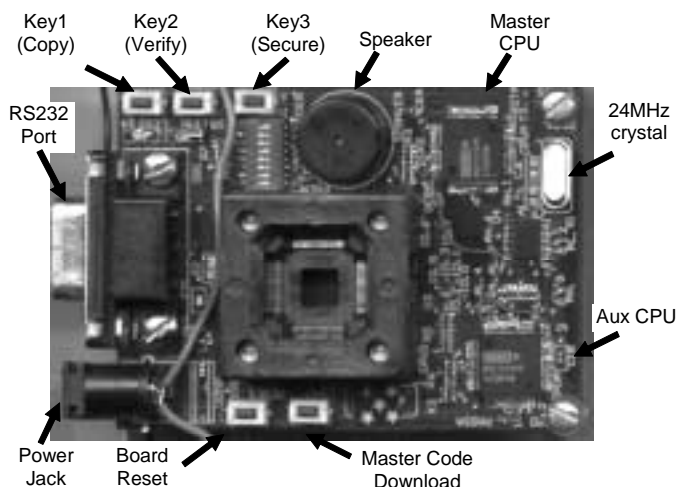


Figure 3. MVP Board (Parallel Programming Configuration)

TI-Downloader software is the host program running on the host PC. U3 is the MSC1210 device under programming. The device being programmed is the slave CPU. The slave CPU uses a 64-pin TQFP socket for easy loading. There are two other CPUs on the programmer: U1 (master) CPU and U2 (aux) CPU. Both master and aux CPUs are implemented with MSC1210 devices. Master and aux CPUs are used only when the MVP board is under parallel programming configuration.

When the programmer is used for parallel programming, the RS232 transceiver in Figure 2 is switched to the master CPU. The host communicates with the master CPU only.

The programmer provides onboard 32KB + 128B Flash to buffer up to 32KB code memory space and 128B hardware configuration memory. These onboard buffer memories are implemented with U2, the aux CPU. The host initializes the buffer memory through the master CPU. After initial buffer downloading, data for programming consecutive devices is accessed from the onboard buffer storage.

When using the serial programming configuration, the master and aux CPUs are not used. If the programmer is used for serial programming configuration only, the master and aux CPU do not have to be installed. Figure 4 shows a picture of the serial programming configuration MVP board that requires significantly fewer components. The slave CPU may communicate with the host PC using the RS232 interface during serial programming configuration.

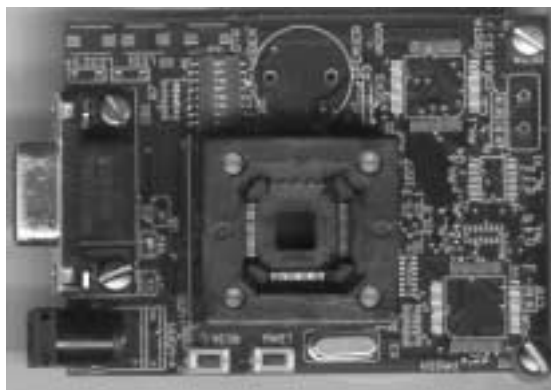


Figure 4. MVP Board (Serial Programming Configuration)

Usage

TI MSC1210 Downloader

The TI MSC1210 Downloader will download an Intel hex file from the PC to the MVP. The Downloader program must be installed using the setup.exe program. The setup program and the Downloader usage document are located at <http://www.ti.com/msc> under Development Tools -> Related Software.



Figure 5. TI DownLoader Installation Screen

Run setup, follow the prompts, and complete the installation. *Note: if the installation does not complete, run the setup.exe program a second time.* The TI Downloader can be programmed with command line arguments by setting up a shortcut to the program and then setting the properties. The download.exe program is placed in the Windows directory by default.

The TI Downloader controls the RS232 DTR and RTS port lines for Reset and ProgLoad operations. For the next program, use the following command line in the *Start ->Run* window:

```
c:\windows\download.exe /S /P1 /X24 /B57600 /T57600 /D
```

Users must enter the appropriate path for the file *download.exe*, and the correct PC COM port number (for example, /P1 as in PC COM1). *Note: The download baud rate must set to 57600 baud for MVP usage.* When the MVP is serial downloading a program to Flash, the download baud rate is fixed at 57600 baud. Option /X24 sets up a 24MHz system crystal, /D turns on debug screen while downloading, and /S postpones download operation.

Operation Screen

The screen shot in Figure 6 shows the initial startup screen. Users must startup the TI-Downloader at 57600 baud.



Figure 6. Start-Up Screen (H for Help)

Pressing 'H' at the startup screen (Figure 6) opens the Help Menu screen (Figure 7), which lists the following basic operations:

- Copy
- Blank Check
- Verify
- Erase
- Secure
- Single Byte Write
- Download from PC to Buffer (Aux CPU) or Target (Slave CPU)
- Buffer or Target Memory Display:
 - a: address setting
 - <: page up
 - >: page down
 - r: page refresh

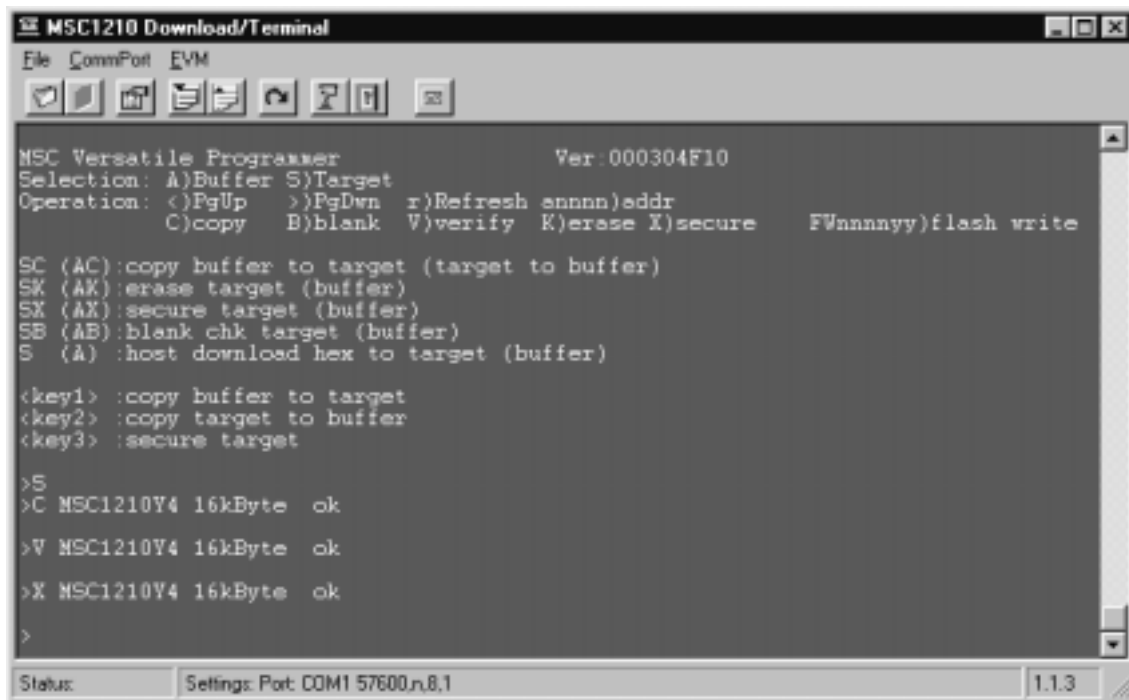


Figure 7. Menu Screen

The SC command, issued in the example screen shown in Figure 7, copies the buffer content to the slave CPU (or target device). The slave CPU information is also displayed on the screen. In this example, it is an MSC1210Y4 device. After the Copy command, Verify and Secure commands are issued.

The MVP board can operate standalone without the TI Downloader. When used in standalone operation, as shown in Figure 3 (page 5), the MVP board has three push-button switches (keys) for function selection:

- Key1—Copy buffer to target device
- Key2—Copy target to buffer
- Key3—Secure target device

There are two LEDs on the MVP board that for indicate the operation status:

- The green LED is on if the operation is passing.
- The red LED is on if the operation is failing.

An onboard speaker is included, which signals operation completion.

Hardware

Power Supply

The power required for the entire board during programming is ~200mA. Two REG113-5, 5V LDO regulators (U7 and U8), are used. U7 powers the slave CPU, and U8 powers the master and aux CPUs. When in parallel programming configuration, all three CPUs are active. When in serial programming configuration, only one CPU is active; thus, power consumption is lowered. Power from the ac adapter is connected through D5A and D5B to the regulators. The ac to 6VDC adapter output current must be rated at a minimum of 250mA.

Power Control for the Slave CPU

Before inserting and removing the device under programming, the master CPU shuts down the power to the slave CPU. LDO regulator U7 supplies power to the slave CPU. Signal VDDSEN of U7 enables power to the slave CPU.

When using the parallel programming configuration, the master CPU can power-down the slave CPU by driving VDDSEN low. To prepare for power-down, the master CPU turns off all control signals to the slave CPU. The master CPU program monitors the slave CPU power supply (VDDS) using AIN7 of the master CPU on-chip ADC. After the master CPU turns on VDDS, it will shut down power to the slave CPU if VDDS is below the normal operating voltage of 2.5V.

RS232 Transceiver

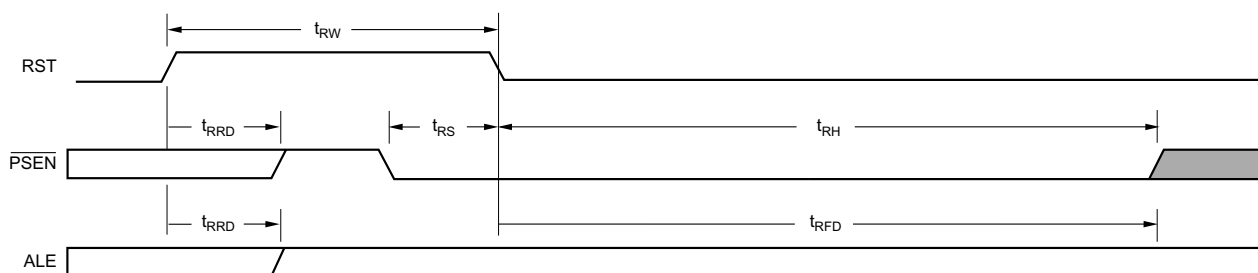
The MVP uses the MAX3223 (U9), 3V RS232 line transceiver; only one transceiver pair is used. The U9 ForceOn and ForceOff lines are configured to have the line transceiver always on for continuous operation.

The same RS232 line transceiver pair is used for PC and slave CPU communication in the serial programming configuration, and for PC and master CPU communication in the parallel programming configuration. U9 receiver line R1IN drives the RXD0 lines of both the slave CPU and the master CPU. U9 transmitter T1IN is selectable via DIP switch settings to be driven by slave CPU TXD0S (for serial programming) or master CPU TXD0M (for parallel programming).

Reset and $\overline{\text{PSEN}}$ Control

A TPS3837 reset generator (U5) with push-pull output is used with the master or slave CPU for serial or parallel programming. The CT input of U5 is connected to ground and sets the reset generator output pulse width to 10ms. An active high reset pulse is generated when the $\overline{\text{MR}}$ line is triggered. The $\overline{\text{MR}}$ line can be triggered by pushing RESET switch S1 or via software controlled transition on the RS232 RTS line. When the reset generator is set to control the master CPU reset line, master CPU I/O line MRST (selectable via DIP switch) controls the slave CPU and aux CPU reset lines.

Another reset generator, the TPS3838 (U6) with open-drain output, is used to set up the slave or master CPUs for programming. This reset generator is controlled by the Load switch. The U6 active low RESET output line drives the master/slave CPU $\overline{\text{PSEN}}$ signals. As shown in Figure 8, if the $\overline{\text{PSEN}}$ signal is held low during master or slave CPU reset, the CPU will enter Serial Flash Programming mode.



NOTE: $\overline{\text{PSEN}}$ and ALE are internally pulled up with $\sim 9\text{k}\Omega$ during RST high.

Figure 8. Serial Flash Programming Power-On Timing (EA is ignored)

Programming the Master CPU

Please do not confuse programming the master CPU with programming the slave CPU. When the MVP board is assembled, the Flash program memory in the master CPU is not initialized. The MVP program must be downloaded from the PC using the TI-Downloader. Once the MVP program is loaded into the master CPU, the $\overline{\text{PSEN}}$ line of the master CPU must be disconnected via a DIP switch setting. Thus, the MVP code in the master CPU will not be corrupted.

To speed up project development, the TI Downloader operates in two phases. In the first phase, during MVP board preparation, the TI Downloader is used for downloading the MVP application code to the master CPU. In the second phase, during MVP operation, the TI Downloader downloads the user application code to either the slave CPU, aux CPU, or both. To the TI-Downloader program, the downloading operation is identical for both phases. During MVP operation (phase 2), the MVP imitates the responses of the MSC device being programmed for the TI-Downloader. As a result, without any extra PC software development, the MVP can use all the TI-Downloader features such as GUI, Intel hex transfer, PC COM port setting, etc.

Parallel Programming Interface

The MVP parallel programming configuration adapts the MSC1210 parallel programming protocol, as shown in Figure 9. The protocol bus has eight bidirectional data lines (Data), 16 address lines (Address), three command lines (Cmd), and three handshaking lines (Req/Ack/Pass). Data has two directions: Write Data and Read Data. The master CPU drives the Write Data bus, Address bus, Command bus, and Req handshake signal (underlined signals in Figure 9). The slave and aux CPUs drive the Read Data bus, and Ack/Pass handshaking signals (bolded signals on Figure 9). The master CPU controls both the slave CPU and the aux CPU via the parallel bus interface.

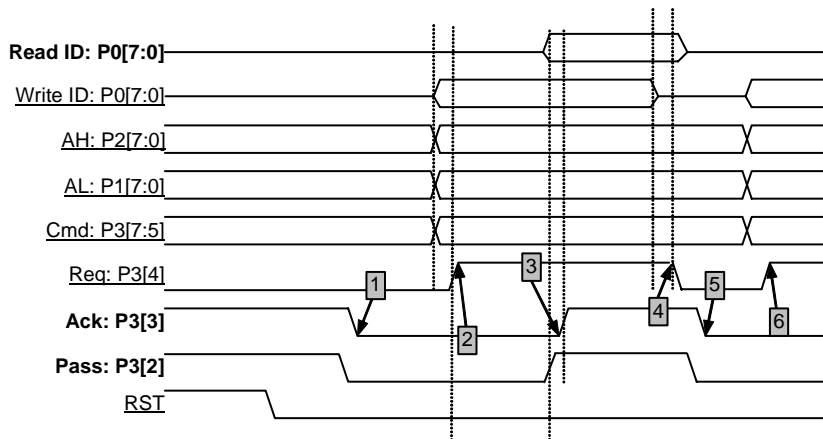


Figure 9. Parallel Programming Handshaking and Timing

Figure 9 shows the parallel programming sequence starting from power-on reset. The master CPU has separate Req/Ack/Pass lines for slave and aux CPUs. The same sequence applies to both slave and aux CPUs. The following text will apply to the slave CPU.

After power on with $\overline{\text{PSEN}}$ and ALE, the slave CPU is setup for parallel programming, with references to the numbers in Figure 9.

1. **Power On:** Ack and Pass are pulled-up high by the slave CPU. Ack and Pass change to zero when the slave CPU is ready to accept a first or new command.
2. **Command Request:** After the master CPU detects a logic low on Ack, the master CPU sets up Addr/Data/Cmd, and raises the Req flag. For read commands (SFR Read, Flash Read), the master CPU drives Data with hi-z. For write commands, the master CPU drives Data with command data.
3. **Command Done:** After the slave CPU detects Req, it executes the requested command, and sets the Pass line if the command succeeds, or clears the Pass line if the command fails (e.g., flash write value check error). If the command is a read command, it drives the read result on the Data bus. Ack is driven high to acknowledge that the command was received and completed.
4. **Master Acknowledge:** The master CPU detects the Ack signal, drives hi-z to Data for write commands or fetches Data for read commands, and releases the Req line to signal the master CPU acknowledge command is complete. Setting up a new Addr and Cmd is optional.
5. **MSC1210 Acknowledge:** After the slave CPU detects Req release, Data is driven with hi-z for read command to avoid bus conflict, and Ack/Pass is released to prepare for a new command.
6. **New command:** Procedure repeats from (2) to (5).

The Parallel Programming Bus commands are:

- Flash Memory Read/Write/Page Erase/Mass Erase
- SFR Read/Write

Parallel Bus

The master CPU needs 48 input and output lines to implement the MVP. The master CPU has ports P0 to P3, which have only 32 I/O lines. An HC374 (U4) is added to implement the parallel programming configuration, which adds eight output lines giving a total of 40. Port use is described in Table 1.

Table 1. Port Use

Device	Signal	Destination
U4	Q [8:1]	AH[7:0]
Master CPU	P1[7:0]	AL[7:0]
Master CPU	P0[7:0]	ID[7:0]
Master CPU	P2[7:0]	Handshake signals and miscellaneous controls

To put the slave/aux CPU into Parallel Flash Programming mode, the ALE pins of the corresponding CPU must be pulled low by the master CPU. Diode D1B and D1A are pulled low through master CPU P2[0].

LED and Key I/O

Since the master CPU I/O lines are limited, the aux CPU P3[1:0] output lines are driving the red and green LEDs. The master CPU controls the aux CPU LEDs using the SFR access through the Parallel Programming Bus commands.

Key1, Key2, and Key3 are sampled using the master CPU analog input lines, AIN0 to AIN3, in order to conserve the digital I/O lines.

Slave/Aux CPU Clock

The master CPU $\overline{\text{PSEN}}$ line drives the slave/aux CPU Xin clock lines. After master CPU power-on reset, the master CPU $\overline{\text{PSEN}}$ line is 6MHz. Since the MSC1210 is a 4-cycle CPU, a 24MHz system clock gives a 6MHz $\overline{\text{PSEN}}$ signal (when the CPU is not accessing external data memory). The MSC1210 has a feature that allows $\overline{\text{PSEN}}$ to be programmed to output the same clock signal as the system clock. Therefore, the master CPU is driving 24MHz to the slave/aux CPU Xin clock lines. A 24MHz Xin for all the CPUs reduces Flash programming time at the cost of MVP system power consumption.

Slave CPU Protection

All slave CPU input and output lines are protected by 100 Ω current limiting resistors, in case of short-circuit. This resistor does not introduce noticeable delay for the Parallel Programming Bus.

As discussed before, the master CPU controls (using VDDSEN) and monitors (using AIN7) the slave CPU power supply. After each operation, the slave CPU power is shut down and the signals are released. This allows for insertion and removal of devices being programmed without powering down the MVP.

Firmware

The TI-Downloader software was developed to download Intel hex format files to the MSC1210, and is plugged directly in to the MVP user interface.

MVP firmware runs with a 24MHz system clock. With a 24MHz master CPU clock, most firmware is developed in C without any noticeable performance degradation. The C language tool used is from Raisonance RIDE [3].

Out of the 32KB available code space from the MSC1210 master CPU, 10KB total code space is used. 120 bytes of IDATA and 32 bytes of XDATA are used from the internal 256 bytes IDATA and 1KB XDATA SRAM.

Interrupt Service Routines

List 1 shows four tasks that are programmed with interrupt service routines (ISRs): Timer, Tone, ADC, and Serial port. Timer/Tone/ADC are serviced by aux_int_isr() and are run at a 2ms rate.

```
void aux_int_isr(void) interrupt 6 {
    if (PAI==5) { // MSINT
        PAI=MSINT; //Dummy read (PAI is read only)
        if (t0!=0) t0--; //GP Timer
        if (t1!=0) {t1--; if(PWM!=0) PWM -= 1;} else PWMCON=0;
        if (t2!=0) t2--; //Tone Timer
    }
    if (PAI==6) { // ADCINT
        if (adc_delay==3){
            if (ADMUX == 0x08) {k1=ADRESH; ADMUX=0x18; }
            else if (ADMUX == 0x18) {k2=ADRESH; ADMUX=0x28; }
            else if (ADMUX == 0x28) {k3=ADRESH; ADMUX=0x78; }
            else if (ADMUX == 0x78) {vdds=ADRESH; ADMUX=0x08; }
            adc_delay=0;
        } else adc_delay++;
        PAI=ADRESL; //Dummy read
    }
    AI=0;
}
```

List 1. Timer/Tone/ADC ISR

Every 2ms, aux_int_isr() will check if the general-purpose times (t0,t1,t2) and the preset tone generator expire. Aux_int_isr() scans Key1~3 and slave CPU power-supply voltage VDDS using the internal ADC. Only an 8-bit result from the 24-bit ADC is used.

The serial port is serviced by serial0_isr() (List 2). RI and TI flags are set when the serial port receive buffer is full or the transmit buffer is empty, respectively. As shown in Figure 10, two 16-byte FIFO buffers at the internal 1KB XDATA memory are assigned for Tx and Rx each. The RI event advances the Rx head pointer (rx_hdptr) and receives a value from the serial port Rx buffer to fill the Rx FIFO. The receiving routine rx_byte() advances the Rx tail pointer (tx_tailptr) to offload the FIFO. The rxfull and rxmt flags monitor if the Rx FIFO is full or empty. The transmitting routine tx_byte() advances the Tx head pointer (tx_hdptr). The TI event advances the Tx tail pointer (tx_tailptr) and sends a value from the Tx FIFO to the Tx buffer. The txfull and txmt flags monitor if the Tx FIFO is full or empty.

```
void serial0_isr(void) interrupt 4
{
    if (RI) {
        *rx_hdptr=SBUF0;
        if (!rxfull) rx_hdptr++;
        if (rx_hdptr==RX_END) rx_hdptr=RX_BEG;
        if (rx_hdptr==rx_tailptr) {rxfull=1;}
        rxmt=0;
        RI=0;
    }
    if (TI) {
        if (tx_tailptr!=tx_hdptr || txfull){ // Send FIFO
            SBUF0=*tx_tailptr;
            tx_tailptr++;
            if (tx_tailptr==TX_END) tx_tailptr=TX_BEG;
            txfull=0;
        } else txmt=1; // FIFO empty
        TI=0;
    }
}
```

List 2. Serial0 ISR

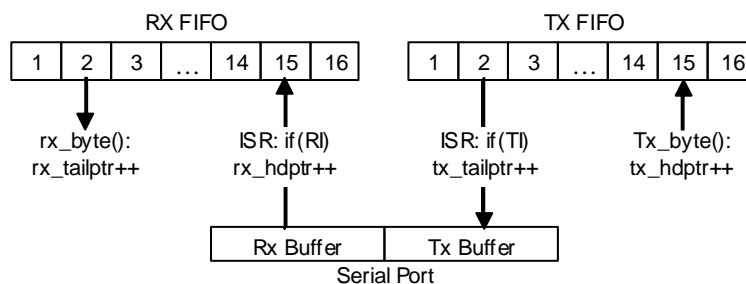


Figure 10. Tx and Rx FIFO

Command Parser

Low-level operations are implemented with some basic commands such as:

- reset the slave/aux CPU
- read/write/mass erase/page erase Flash memory.

Basic commands are used for the TI-Downloader interface and user interface. Users may issue an SC command to copy buffer content to the slave CPU. The SC command detects the installed target part and extracts part info, mass erases the part, and performs a Flash write to the part according to the memory size of the target part. The TI-Downloader may issue an 'L' command that will start the Intel hex data format parser, and random write access of Flash. The TI-Downloader related commands are:

- **M**—mass erase
- **RR/RW**—register read/write
- **E**—echo on/off

Some user interface commands are:

- **V**—verify
- **B**—blank check
- **K**—erase
- **X**—secure

There are extra commands for system testing.

Parallel Access Function

List 3 lists the function `fpm()` that implements slave/aux CPU parallel register/Flash access timing, as shown in Figure 9. Since the slave and aux CPUs have dedicated handshaking lines (Reqs/Acks and Reqa/Acka), the master CPU can send individual commands to each CPU or broadcast commands to both CPUs together. The SFR read/write and Flash read command access times are $\sim 23\mu\text{s}$, the Flash write command is $\sim 140\mu\text{s}$, and the Flash erase command access time is $\sim 12\text{ms}$. Overall programming time is listed in the Features section.

```
unsigned char fpm(unsigned char cmd, unsigned int a, unsigned char pd)
{ unsigned char c,cpu, cmdx,rdcmd;
  // Warning: issues SRD/FRD to both CPUA & CPUS will cause bus conflict!!
  cpu=cmd & 0xf0; cmdx=cmd & 0x0f;
  if (cmdx== SRD || cmdx==FRD) {rdcmd=1; P0DDR=P0IDLE; id=0xff;}
  else {P0DDR=P0OUT; id = pd; rdcmd=0;}
  ah= (unsigned char)(a>>8); CMD0 = cmdx & 0x01; CP=0; CP=1; CMD1= cmdx & 0x02;
  al=(unsigned char)a; CMD2= cmdx & 0x04;
  if (cpu==CPUA) { // Aux CPU command
    REQA=1; while(ACKA==0){}; c=(unsigned char)!PASSA;
    if (rdcmd!=0) c=id; else { P0DDR=P0IDLE; id=0xff; }
    REQA=0; while(ACKA==1){};
  } else if (cpu==CPUS) { //Slave CPU command
    REQS=1; while(ACKS==0){}; c=(unsigned char)!PASSS;
    if (rdcmd!=0) c=id; else { P0DDR=P0IDLE; id=0xff; }
    REQS=0; while(ACKS==1){};
  } else if (cpu==CPUBOTH) { //Slave & Aux CPU command
    REQS=1; REQA=1; while(ACKS==0 || ACKA==0){};
    c=(unsigned char)!(PASSS & PASSA); P0DDR=P0IDLE; id=0xff;
    REQS=0; REQA=0; while(ACKS==1 || ACKA==1){};
    return(c);
  }
}
```

List 3. Parallel Access Function `fpm()`

System Testing

The serial programming configuration does not need the master and aux CPUs and their related parts. After assembling the board, measure the power-supply output lines (VDD/VDDS/VDDM) and make sure they are not short to ground. Apply 6VDC to the power jack, and measure VDDS and VDDM with the reset key pressed; they must be 5V. Measure pin 3 and 7 of the MAX3223; they should be 7V and -6V. Set the DIP switch to the Master CPU Setup setting shown in Table 2. You are ready to download a program to the master CPU. Table 2 also shows the DIP switch setting for parallel and serial programming configurations.

Table 2. DIP Switch Settings

	DIP Switch S1							
	1	2	3	4	5	6	7	8
MVP Parallel Programming Configuration	OFF	OFF	ON	ON	OFF	OFF	OFF	OFF
MVP Serial Programming Configuration ⁽¹⁾	OFF	ON	OFF	OFF	ON	ON	ON	ON
Master CPU Setup	ON	OFF	ON	ON	OFF	OFF	OFF	OFF
NOTE: (1) The following components are not needed for MVP serial programming configuration: U1, U2, and U4. If they are installed onboard, cut the following onboard resistor-jumpers: R11, R12, and R13.								

Short-circuits on the bus signals are a common problem when assembling the board. If master CPU Flash hardware configuration memory address 8003h is cleared to 00h, the MVP code will toggle the AH/AL/Cmd/Reqa/Reqs lines. Measure the lines and make sure there is no open-circuit or short-circuit on the bus from the master CPU to the slave and aux CPUs.

Some extra commands are included to further test the board. These commands are shown in Table 3. Issue the commands from the MVP application prompt.

A file (AA32KB.Hex) that contains 32KB of all hex value AA is attached in the project directory for download testing.

Table 3. Board Checkout Commands

Command	Descriptions
U0C	Slave retention check
U0D	Slave retention check and march test
U0E	Write to whole memory with incrementing numbers and check results
U0F	Program slave CPU with AA and check results
U10	LED/Beeper test
U11	Key on/off test
U12	Key voltage level test

Other Supported Configurations

ISP Configurations

Many applications do not need an RS232 port; therefore, no RS232 level shifter is on the system board. The MVP provides ISP configuration, as shown in Figure 11. The MVP includes a serial interface connector, as shown in Figure 12. In order to communicate with the TI Downloader, an RS232 transceiver is needed on the target system. Also needed are RST and PSEN delay devices, which configure the target systems to Serial Programming mode. For systems without these devices, and to save on system cost, the ISP connector provides access to the functionality of these MVP devices. The 6VDC and ground pins of the ISP connector can be used to drive an external circuit, or be connected to a power bus that drives multiple MVP boards.

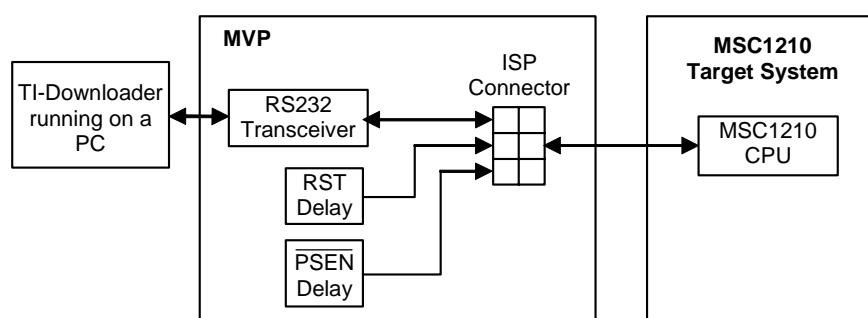


Figure 11. MVP ISP Configuration

MVP RXD Input	Pin 1	Pin 2	MVP TXD Output
Ground	Pin 3	Pin 4	6VDC VDD I/O
MVP RST Output	Pin 5	Pin 6	MVP PSEN Output

Figure 12. ISP Configuration Connector

Gang Serial Configuration

The lower-cost serial programming configuration does not need the master and aux CPUs. A PC with multiple serial ports allows multiple devices to be programmed concurrently. Users can spawn multiple TI-Downloader programs to program the parts.

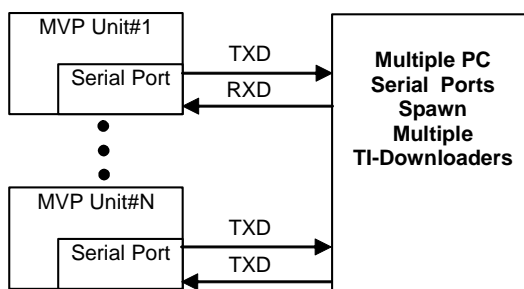


Figure 13. MVP Gang Serial Configuration

Gang Parallel Programming Configuration

The high-speed parallel programming configuration can extend to a gang parallel programming configuration that programs multiple devices concurrently. Since each MVP board has its own LED indicators and program switches, a PC terminal program or TI-Downloader is not needed for mass production. To setup the gang parallel programming configuration, use an MSC device with the master code already programmed, place it into the TQFP (ZIF) socket on the MVP, and press Key2; that copies the master code to the buffer of the MVP. Repeat this process for each MVP device connected to the gang system. Use Key1 to copy the code from the MVP to a blank MSC device. Since it only takes 10 seconds to duplicate one part, five MVP boards in a gang parallel configuration are enough to maintain continuous programming with waiting time between programming. Figure 15 shows a prototype of a five-MVP gang parallel system.

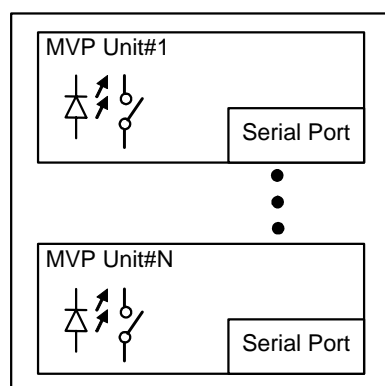


Figure 14. MVP Gang Parallel Configuration

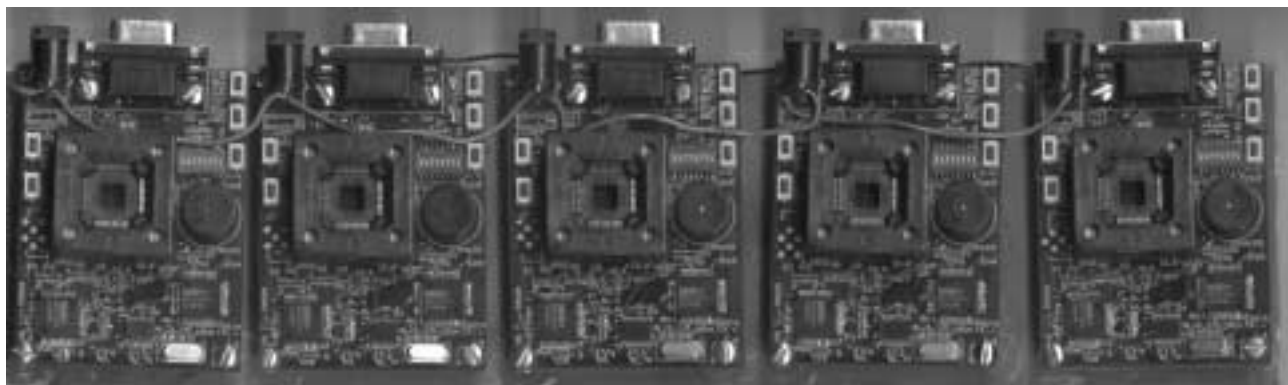
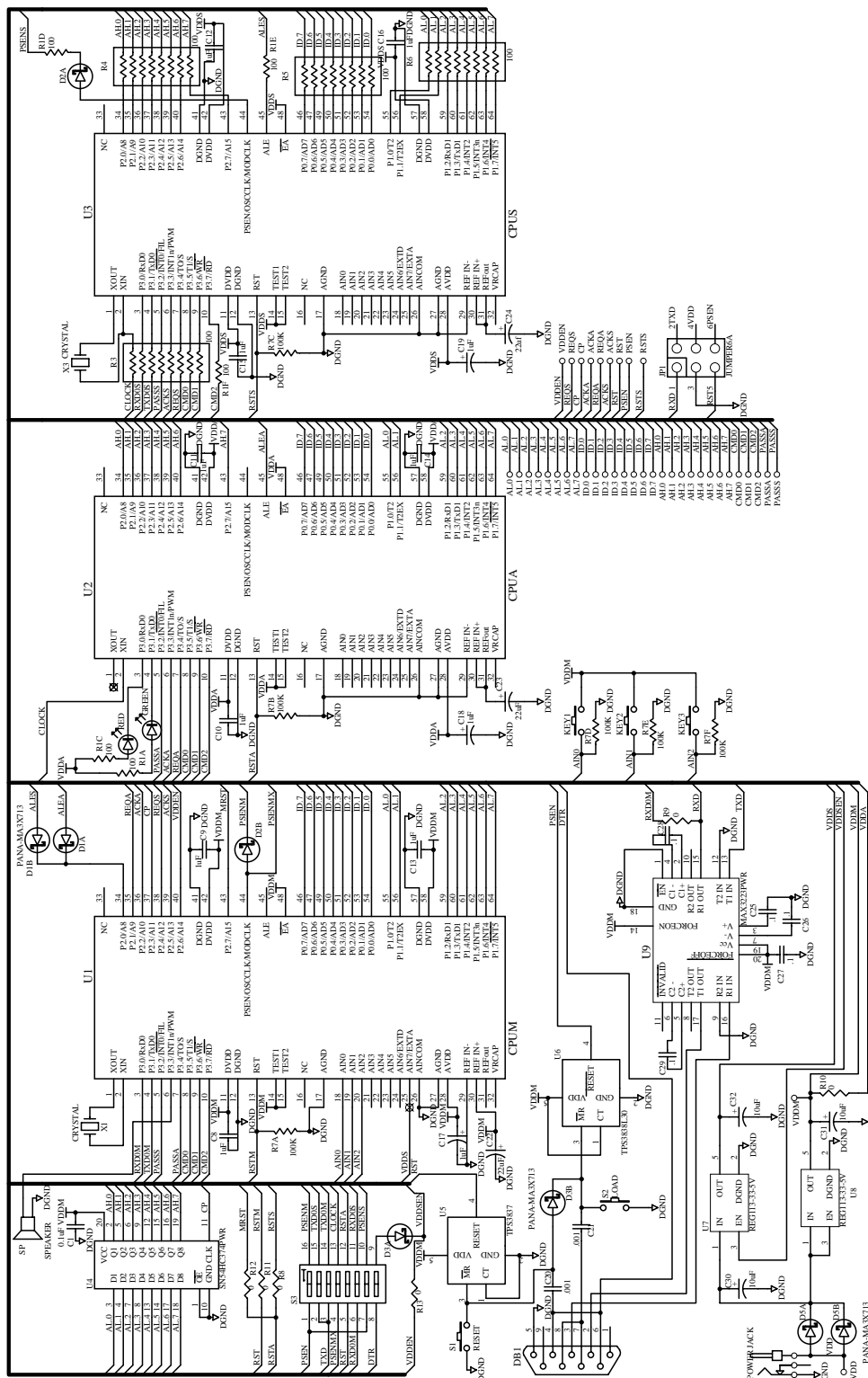


Figure 15. MVP Gang Parallel System with Five MVP Boards

Appendix A: MSC1210 Versatile Programmer Schematic



References

- [1] MSC1210 Data Sheet MSC1210.pdf (766KB, Rev.A) (Updated: 03/26/2002)
- [2] Programming the MSC1210 (Rev. A) (SBAA076A - Updated: 04/17/2002)
- [3] Raisonance RIDE: www.raisonance.com

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265