How to Plug a TSC80251G1 Step A in a C51 Board ?

Introduction

This application note assumes that the user is familiar with C51 microcontrollers.

The purpose of this application note is to address the software and the hardware design considerations when migrating from C51 microcontrollers to the first general purpose TEMIC C251 microcontroller, the TSC80251G1 Step A.

Without changing the code, the average performance will increase by a factor from 2 to 5 (See “Speed Increase” part). If the new instruction set is used, the performance may increase up to a factor of 15.

To plug a TSC80251G1 into a C51 socket, the user has to consider three points:

- the pin–to–pin replacement,
- the programming of the two configuration bytes,
- the speed increase.

Caution:

If the chip is a TSC80251G1 –B (See Ordering Information in Product Datasheet), the microcontroller is already programmed to be C51 compatible and the user is not concerned with programming the configuration bytes.

Pin–to–Pin Replacement

The C51 microcontrollers have 4 NC (Not Connected) pins which are used on TSC80251G1 microcontrollers.
In Step A, these pins must not be floating, so please connect VSS1, VSS2, NMI (respectively pins #1, #23 and #34 in PLCC44) to the Ground and WAIT# (pin #12 in PLCC44) to VCC.
However, the TSC80251G1 Step C will provide a pin–to–pin replacement of C51 microcontrollers without these constraints.

Configuration Bytes

The TSC80251G1 Step A provides a variety of features and operating modes by programming two configuration bytes CONFIG0 and CONFIG1 (See Figures 1 and 2). These bytes are read from specific registers (See Table 1) during the reset of the chip.

Table 1. Configuration Bytes Location

Microcontrollers	Location of registers (⋆)	Programming
TSC87251G1 EPROM	On–chip EPROM	by user or ask factory
TSC87251G1 OTPROM	On–chip OTPROM	by user or ask factory
TSC83251G1 MaskROM	Metal Mask	in factory
TSC80251G1 ROMless	Metal Mask	in factory

(⋆) In Step C, user configuration bytes (UCONFIG0 and UCONFIG1) are used instead of CONFIG0 and CONFIG1. In all cases (EPROM, OTPROM, MaskROM, ROMless, ...), the user is able to program these bytes. He must do it unless it has been ordered to factory (EPROM, OTPROM, MaskROM).

CONFIG0 (80h)
Configuration byte 0

–	–	WSA	XALE	RD1	RD0	PAGE	SRC
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**

Bit Number	Bit Mnemonic	Description
7	–	**Reserved** Set this bit when writing to CONFIG0.
6	–	**Reserved** Set this bit when writing to CONFIG0.
5	WSA	**Wait State A bit** ⇒ Clear to generate one wait state for memory regions 00:, FE: and FF:. Set for no wait states for regions 00:, FE: and FF:.
4	XALE	**Extend ALE bit** Clear to extend the time of the ALE pules from T_{OSC} to 3.$T_{OSC.}$ ⇒ Set to keep the time of the ALE pulse to T_{OSC}.
3	RD1	**Memory Signal Selection bits** Codes specify a 17–bit or 16–bit external address bus and address ranges for RD#, WR# and PSEN# signals.
2	RD0	

	RD1	RD0	RD#	P1.7	PSEN# Range
	0	0	A16	A17	PSEN# is the read signal for both external data and program address spaces (256 Kbytes).
	0	1	A16	I/O pin	PSEN# is the read signal for both external data and program address spaces (128 Kbytes).
	1	0	I/O pin	I/O pin	PSEN# is the read signal for both external data and program address spaces (64 Kbytes).
⇒	1	1	RD#	I/O pin	PSEN# is the read signal for the external program address spaces (64 Kbytes) and RD# is the read signal for the external data address space (64 Kbytes).

1	PAGE	**Page Mode Select bit** Clear for page mode with A15:8/D7:0 on Port 2 and A7:0 on Port 0. ⇒ Set for non–page mode with A15:8 on Port 2 and A7:0/D7:0 on Port 0.
0	SRC	**Source Mode/Binary Mode Select bit** ⇒ Clear for binary mode. Set for source mode.

Note:
The arrow (⇒) shows the recommended configuration for C51 compatibility.

Figure 1. Configuration Byte 0

CONFIG1 (81h)
Configuration Byte 1

–	–	–	INTR	WSB	–	–	EMAP
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**

Bit Number	Bit Mnemonic	Description
7	–	**Reserved** Set this bit when writing to CONFIG1.
6	–	**Reserved** Set this bit when writing to CONFIG1.
5	–	**Reserved** Set this bit when writing to CONFIG1.
4	INTR	**Interrupt Mode bit** ⇒ Clear so that the interrupts push 2 bytes onto the stack (the 2 lower bytes of the PC register). Set so that the interrupts push 4 bytes onto the stack (the 3 bytes of the PC register and the PSW1 register).
3	WSB	**Wait State B** ⇒ Clear to generate one wait state for memory region 01:. Set for no wait states for region 01:.
2	–	**Reserved** Set this bit when writing to CONFIG1.
1	–	**Reserved** Set this bit when writing to CONFIG1.
0	EMAP	**EPROM Map bit** Clear to map the upper 8 Kbytes of on–chip code memory (FF:2000h–FF:3FFFh) to 00:E000h–00:FFFFh. ⇒ Set to not map the upper 8 Kbytes of on–chip code memory (FF:2000h–FF:3FFFh)

Note:
The arrow (⇒) shows the recommended configuration for C51 compatibility.

Figure 2. Configuration Byte 1

The default configuration for OTPROM/EPROM microcontrollers is CONFIG0 = 1111 1111b and CONFIG1 = 1111 1111b.

The configuration recommended for C51 compatibility (hardware and software) is CONFIG0 = 1101 1110b and CONFIG1 = 1110 0111b.
In details:

- SRC = 0 for binary mode.
- PAGE = 1 for non–page mode.
- INTR = 0 to push 2 bytes onto the stack.
- RD1 = 1 and RD0 = 1 for memory signals.
- XALE = 0 to keep the ALE pulse width (0 wait state).
- WSA = 0 and WSB = 0 to add 1 wait state for all external memory regions.
- EMAP = 1.

The first four points are mandatory to follow for C51 compability while the others are optionnal. In fact, the internal code memory of the TSC80251G1 is faster than most of the external memory and peripheral device available. When

interfacing the TSC80251G1 with slower devices (used on C51 board), a wait state can be used to extend the external system bus cycle. The TSC80251G1 Step A can be configured to generate 0 or 1 wait state for ALE, PSEN#, RD# and WR# signals. With 0 wait state, the pulse of these signals is 1 oscillator period. With 1 wait state, the pulse widths are extended to 3 oscillator periods.

The user may need to check the timing specifications for the TSC80251G1 and the external devices to ensure that the TSC80251G1 will be able to interface successfully with external devices.

Speed Increase

The TSC80251G1 uses the new C251 microcontroller's core that is different from the traditional C51's one. It is designed based on a pipelined architecture and a register–based machine. Therefore, the execution time decreases and the user must consider changing the the timing loops or sequences of C51 code running on TSC80251G1 when he relies on C51 execution time.

For example, a delay time is used to provide a 13 µs delay in a 12 MHz 80C51 application.The code and the taken time are shown on Listing 1.

```
        ; Internal or External Code Execution          Taken time in number of cycles

              MOV   R0,#06h                            ; 1
LOOP    :     DJNZ  R0,LOOP                            ; 2

        ; 1 cycle = 12 oscillator periods
        ; Total oscillator periods = [ 1 + ( 6 x 2 ) ] x 12 = 156
        ; Total time taken in the timing loop = 156 x 83.33 ns = 13 µs
```

Listing 1. Loop executed at 12 MHz on 80C51

In the same application, the 80C51 microcontroller is replaced by a 12 MHz TSC80251G1 programmed using the recommanded configuration for C51 compatibility in binary mode. No modification is made to the code shown on Listing 1. But to simplify timing loop calculation, the code is executed in internal memory. The code and the taken time is shown on Listing 2.

```
        ; Internal Binary Code Execution               Taken time in number of states

              MOV   R0,#06h                            ; 1
LOOP:         DJNZ  R0, LOOP                           ; 5 and 2 for exiting the loop

        ; 1 state =  2 oscillator periods
        ; Total oscillator clock periods = [ 1 + ( 5 x 5 + 2 ) ] x 2 = 56
        ; Total time taken in the timing loop = 56 x 83.33 ns = 4.66 µs
```

Listing 2. Loop executed at 12 MHz on TSC80251G1 without code modification

To maintain the same delay time, some changes in the assembly code are needed. The new code and the taken time are shown on Listing 3.

```
        ; Internal Binary Code Execution          Taken time in number of states

              MOV   R0,#0Fh                       ; 1
LOOP:         DJNZ  R0, LOOP                      ; 5 and 2 for exiting the loop

        ; 1 state =  2 oscillator periods
        ; Total oscillator clock periods = [ 1 + ( 15 x 5 + 2 ) ] x 2 = 156
        ; Total time taken in the timing loop = 156 x 83.33 ns = 13 µs
```

Listing 3. Loop executed at 12 MHz on TSC80251G1 with code modification

Calculating execution time for TSC80251G1 is not as easy as for 80C51. The execution time depends on whether the code is fetched: from internal or external memory. When bytes are fetched in external memory in non–page mode and with one wait state, each fetch takes 6 oscillator periods (3 states) and the corresponding speed increase factor is 2. When no wait states are added, this factor becomes 3. When bytes are fetched in external memory in page mode, each fetch requires only 2 oscillator periods (1 state) and this factor becomes 6.
It is the same factor for internal code execution, which is the fastest case. But the average speed increase factor is 5 since all instructions do not take the same execution time.
The user may note that, on Listing 2, the execution time is only divided by a factor of 2.7 because DJNZ is one of the slowest instruction.

Caution:

To execute code in page–mode, an address latch must be connected to Port 2 instead of Port 0 and the hardware of 80C51 applications must be changed.
To use existing 80C51 applications without changing your hardware, please execute code in non–page mode.