Application Note 615
Design Considerations for DS80C400-Based TINI Systems

www.maxim-ic.com

Introduction

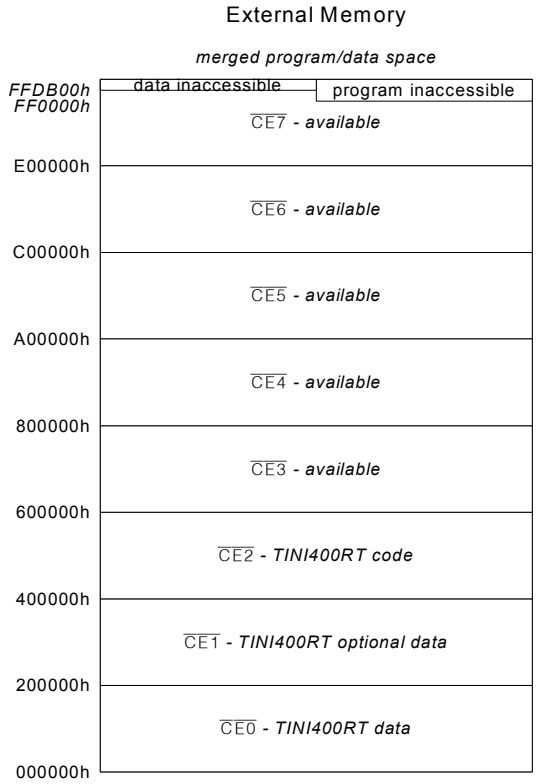
The TINI® (Tiny InterNet Interfaces) runtime environment executes Java™ byte codes on the DS80C390 and DS80C400 microcontrollers. An overview and detailed documentation for the TINI runtime and the DS80C400 microcontroller can be found on Maxim's website at www.maxim-ic.com.

Dallas Semiconductor/Maxim has published several reference designs for the DS80C400-based TINI environment, for example the TINIm400-144. However, a customer might wish to fine-tune these reference designs with regard to memory and clock speed and wish to integrate the module with the socket in order to save cost, optimize power consumption, and better solve a particular application's challenges. This application note presents design considerations that allow building of faster TINI modules with more memory, while still maintaining compatibility with the standard TINI runtime.

Beyond Reference Designs—Memory

On startup, the DS80C400 ROM boot loader configures the chip-enable ($\overline{CE}$) signals to 2MB each, i.e., 2MB in the memory map are allocated for each physical memory chip (see page 4 of this document for information on how to use 4MB per chip enable). Unlike traditional 8051 designs, the memory is configured in *von Neumann*-mode, using a unified (merged) memory map as shown in Figure 1 for program and data memory.

Figure 1. TINI400RT Memory Map



TINI is a registered trademark of Dallas Semiconductor.
Java is a trademark of Sun Microsystems.

REV: 080904

The DS80C400 TINI runtime (TINI400RT) requires

- at least 512kB of program memory (read-only or read-write) – *typically flash memory,*
- at least 512kB of data memory (read-write) – *SRAM.*

The program memory is used to store the TINI400RT code (loaded from the *tini400.tbin* file) as well as the user application byte codes. The data memory is used for the heap and the file system, which can contain additional user applications.

In order to allow for different heap/file system sizes or provide more space for user application data, several memory configurations are automatically recognized. A *master erase* is required when changing memory configurations.

Program Memory

Required: 512kB connected to $\overline{CE2}$.
Optional: Any amount up to 2MB on $\overline{CE2}$.
Maximum: Up to 2MB of program memory plus any amount on $\overline{CE3}$–$\overline{CE7}$.
Note: The TINI400RT uses only the first 512 KB of program memory on $\overline{CE2}$ (at address 400000h). Any additional memory is ignored by the TINI400RT and must be managed by the user application.

Data Memory

Required: 512kB connected to $\overline{CE0}$.
Optional: 1MB or 2MB on $\overline{CE0}$, 0/512kB/1MB/2MB on $\overline{CE1}$.
Maximum: Up to 4 MB of data memory in the configurations listed in *Table 1*.

The configurations in Table 1 are automatically recognized and the heap and the file system are sized accordingly.

Table 1. Supported TINI400RT Data Memory Configurations

$\overline{CE0}$			$\overline{CE1}$			TOTAL
512kB	**1MB**	**2MB**	**512kB**	**1MB**	**2MB**	
X						512kB
	X					1MB
X			X			1MB
	X		X			1.5MB
		X				2MB
	X			X		2MB
		X	X			2.5MB
		X		X		3MB
		X			X	4MB

Real-Time Clock, Clock Frequency, and Multiplier

The TINI400RT runs from approximately 8MHz to 75MHz. When the system starts up, the TINI400RT first enables the preconfigured clock multiplier (see below) and uses the real-time clock (RTC) to measure the CPU speed. The CPU speed is the product of the external crystal oscillator and the clock multiplier.

The RTC is optional and a default CPU speed of 29.4912MHz is assumed in the absence of an RTC.

Default CPU Speed: 29.4912MHz
Default Clock Multiplier: 2
(Default Crystal Frequency: 14.7456MHz)

When an RTC is present, the TINI400RT automatically adjusts for different crystal frequencies. A reboot is required when changing the CPU speed (a *master erase* is not necessary, i.e., the contents of the heap and the file system are preserved).

To change the clock multiplier and the *default* CPU speed, the TINI400RT memory image has to be modified. Table 2 shows the TINI400RT header loaded into memory at address 400000h. The default CPU speed is at offset 7 (address 400007h), the default multiplier at offset 9 (address 400009h).

Table 2. TINI400RT Header Data Structure

OFFSET	LENGTH	DESCRIPTION
0	2	sjmp (80 xx)
2	4	'TINI' (54 49 4E 49)
6	1	Code Bank (40)
7	2	Default Speed
9	1	Default Multiplier

The CPU speed value is stored as CPU frequency in kHz divided by 1.2. Table 3 shows several examples for given CPU speeds.

Table 3. Example CPU Speed Values

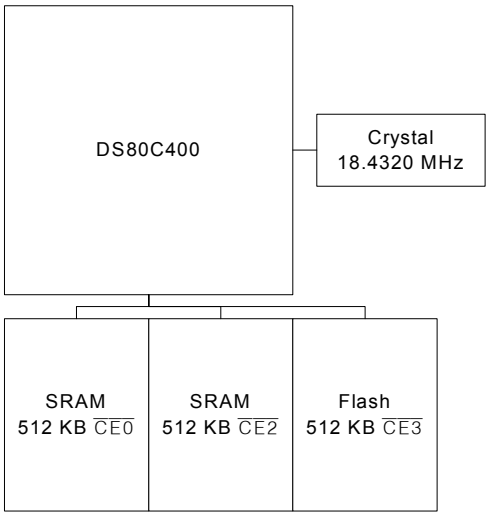
CPU SPEED (MHz)	VALUE
29.4912	24,576
36.8640	30,720
73.7280	61,440
75.0000	62,500

The multiplier is 1, 2, or 4. Optionally, you can modify the TINI400RT before loading it into memory. Please contact micro.software@dalsemi.com for a utility that modifies the TINI400RT file *tini400.tbin* for different default settings while preserving the correct CRC (see Appendix A for a description of the *tbin* file format).

Example: Copying from Flash to SRAM

Since fast flash memories are expensive, the TINI400RT image and the user application can be copied from slow inexpensive flash memory to fast SRAM on every boot before turning on the clock multiplier. The example configuration does not use a real time clock and is sketched in Figure 2.

Figure 2. Example High-Speed Configuration



The target speed of the system is 73.7280MHz (using the *x*4 multiplier).
The boot sequence is as follows:

1) The system initially starts at 18.4320MHz (*x*1).
2) The DS80C400 ROM loader starts up and initializes $\overline{CE0}$ through $\overline{CE7}$ to 2MB each.
3) The DS80C400 ROM loader scans the memory and locates the flash at $\overline{CE3}$ (address 600000h).
4) The DS80C400 ROM transfers control to the program located in flash.
5) The flash program copies the TINI400RT image to the SRAM at $\overline{CE2}$ (address 400000h).
6) The flash program sets the memory at location 400007h to 61440 and the memory at location 400009h to 4, thus configuring the multiplier and CPU speed.
7) The flash program transfers control to the TINI400RT at address 400000h.
8) The TINI400RT reads its crystal multiplier value and programs the crystal multiplier to *x*4.
9) The system now runs at 73.7280MHz and loads the TINI400 runtime and user application.

Note: A nonvolatizer for the data SRAM at $\overline{CE0}$ is optional.

Please contact micro.software@dalsemi.com if you need assistance implementing the firmware copy function (see Appendix B for an example). *Hint:* The flash copy code could also initialize the heap/file system with a preconfigured image.

Unused $\overline{CE}$ Pins

If a design does not use all $\overline{CE}$ pins ($\overline{CE0}$ through $\overline{CE7}$), the unused pins can be reconfigured to be general-purpose I/O pins. Reconfiguration and access to the I/O have to be programmed from a native library written in 8051 assembly language. The TINI firmware distribution contains native library examples; the DS80C400 datasheet and user's guide supplement describe the special function registers needed to configure the pins' alternate functions. For example, the following code sets $\overline{CE4}$ to $\overline{CE7}$ as I/O pins.

```
mov TA, #0aah              ; Timed access
mov TA, #55h
mov P6CNT, #00100000b      ; Write new configuration
```

Table 4. $\overline{CE}$ Pins and Corresponding Alternate Functions

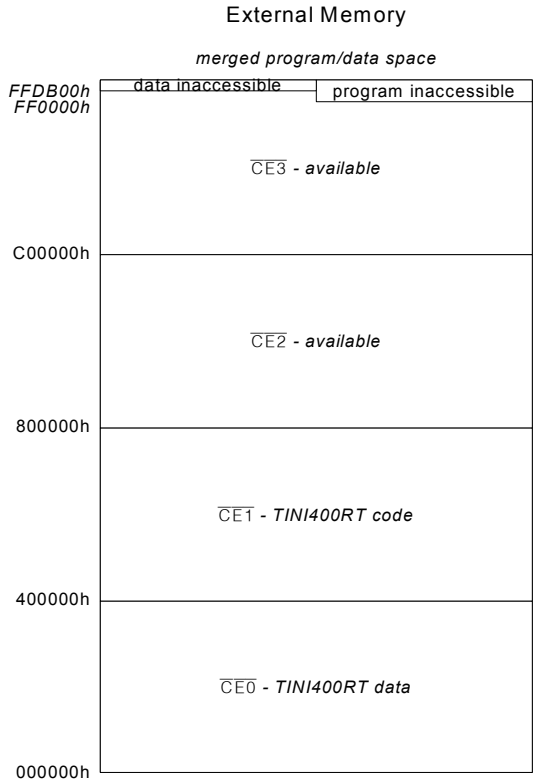
$\overline{CE}$ SIGNAL	ALTERNATE FUNCTION: PORT PIN
CE1	P4.1
CE3	P4.3
CE4	P6.0
CE5	P6.1
CE6	P6.2
CE7	P6.3

Using 4MB per Chip Enable

Although not directly supported by the ROM, 4MB per chip enable can be used by the TINI400RT. Note that the memory map changes when using 4MB per chip enable (see Figure 3) and the TINI400RT image must always be loaded at address 400000h ($\overline{CE1}$ instead of $\overline{CE2}$).

TINI400RT does not auto-detect different chip-enable sizes. A user program has to change the chip enables to 4MB (e.g., in the firmware copy function outlined on page 3).

Figure 3. TINI400RT Memory Map with 4MB per Chip Enable



More Information

Details of the TINI Platform are on the www.maxim-ic.com/TINI website. The *TINI Specification and Developer's Guide* is an invaluable resource when developing with the TINI platform, and can be downloaded from the Maxim website. Application Notes 612 and 708 are guides to getting started with the DS80C400-based TINI modules.

Maxim Integrated Products/Dallas Semiconductor Contact Information

Company Addresses:

Maxim Integrated Products, Inc.
120 San Gabriel Drive
Sunnyvale, CA 94086
Tel: 408-737-7600
Fax: 408-737-7194

Dallas Semiconductor
4401 S. Beltwood Parkway
Dallas, TX 75244
Tel: 972-371-4448
Fax: 972-371-4799

Product Literature/Samples Requests:
800-998-8800
408-737-7600

Sales and Customer Service:

Website:
www.maxim-ic.com

Product Information:
http://www.maxim-ic.com/products.cfm

Ordering Information:
http://www.maxim-ic.com/sales

FTP Site:
ftp://ftp.dalsemi.com

Appendix A: TINI400RT *tbin* File Format

A *tbin* file consists of multiple blocks that each describe up to 64 KB of data, as show in Table 5.

Table 5. tbin File Format

OFFSET	LENGTH	DESCRIPTION
0	3	Target address (LSB first) for data
3	2	Length-1 (LSB first) of data
5	*	Data (1 to 65536 bytes)
*	2	CRC-16 (LSB first) of data

Appendix B: Example Code

The following code fragment copies the TINI400RT firmware from flash to SRAM and configures the system to change the clock multiplier to *x*4. The code then jumps to the SRAM and starts executing the TINI400 runtime environment.

```
$include (ds80c400.inc)

; The desired multiplier
MULTIPLIER equ 4

; The resulting CPU speed (kHz/1.2)
SPEED equ 61440

org 600000h

; The following is the entry point for the ROM boot loader
startup:
    sjmp begin

; The following bytes are required by the ROM boot loader
    db 'TINI'
    db 60h

; This is where we begin executing
begin:
    ; First, copy the TINI400RT from flash to RAM

    mov dps, #0             ; Select dptr0
    mov dptr, #800000h      ; Source of the TINI400RT in flash (change this)
    inc dps                 ; Select dptr1
    mov dptr, #400000h      ; Target of copy operation (CE2\)

    mov dps, #30h           ; Select dptr0, auto-inc dptr, auto-inc dps
copyloop:
    movx a, @dptr           ; Get a byte from dptr0, inc dptr, inc dps
    movx @dptr, a           ; Put it at dptr1, inc dptr, inc dps
    mov a, dpx
    cjne a, #88h, copyloop  ; Copy until dptr0 points to 880000h
                            ;   which means we've copied 512KB

    mov dps, #0             ; Select dptr0

    ; Set the desired multiplier to x4 and the resulting CPU speed
    mov dptr, #400007h      ; CPU speed high byte
    mov a, #high(SPEED)
    movx @dptr, a           ; Store high byte of resulting speed
    inc dptr                ; Point to low byte of CPU speed
```

```
    mov a, #low(SPEED)
    movx @dptr, a           ; Store low byte of resulting speed
    inc dptr                ; Point to desired multiplier
    mov a, #MULTIPLIER
    movx @dptr, a           ; Store the desired multiplier

    ; Run the firmware by pushing the target address 400000h
    ;   and executing a ret
    clr a
    push acc                ; LSB byte of address
    push acc
    mov a, #40h
    push acc                ; MSB of address
    ret

end
```