PROGRAMMING REFERENCE MANUAL

RFA-04-041 Rev 1.2

05/2023

Hexius Semiconductor
4640 E. Elwood Street, Suite 19
Phoenix, AZ 85040
U.S.A.

CONTENTS

FIGURES

TABLES

REVISION HISTORY

Revision	Date	Description
1.0	01/2023	Initial Release
1.1	03/2023	Revised for Firmware Version 1.8, typographical and clarity corrections
1.2	05/2023	Revised for Firmware Version 1.9, updated setup block naming, typographical and clarity corrections

1 Overview

Processor Architecture

The control unit for the TM100 and TM200 is an embedded 8-bit 8051 type microcontroller including ROM, RAM, and programmable memory. The microcontroller's main purpose is to configure, store, and adjust parameters. The embedded firmware actively tunes the crystal based on the sensed temperature.

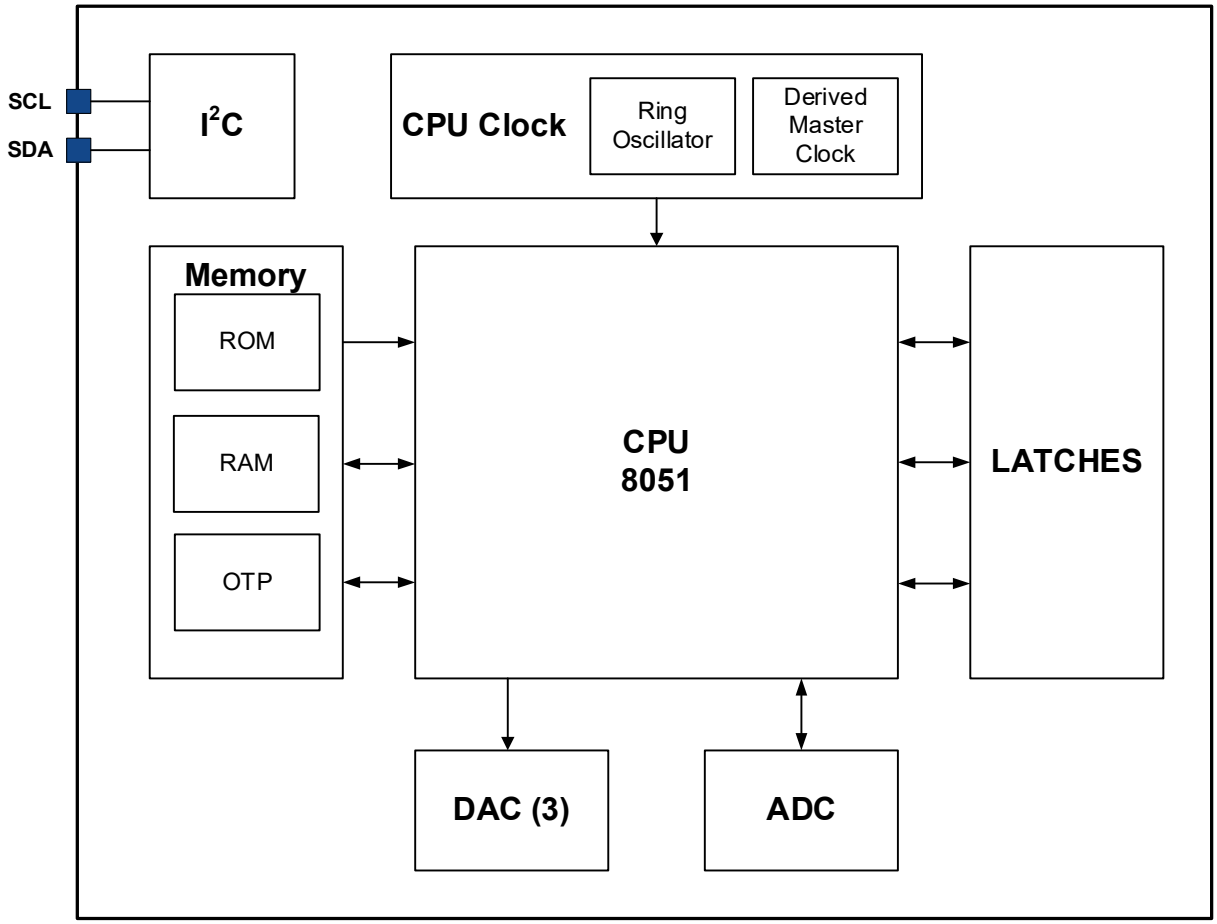

Figure 1-1 TM100/TM200 Processor Architecture

I²C Communications

The I²C interface of the TMx00 supports bi-directional communications for setup and testing. Refer to section 2 for details.

ROM Memory

A 3K ROM occupies the low end of the memory space. This bank of memory contains startup routines including one-time programmable (OTP) memory interface routines, initialization routines, and other low level system functions.

Internal and External RAM Memory

The µP architecture includes two types or RAM. 256 bytes are located internally. The external data memory contains an additional 768 bytes of storage.

One Time Programmable (OTP) Memory

32 Kbytes of OTP memory starts above the ROM. It contains program code, setup parameters, and test sequences.

Typically, test sequences and program code will be loaded into the OTP memory during production test. The setup parameters will be written during the oscillator characterization. Some of the parameters will define the general operational mode of the IC, and those will be written at the front end of the manufacturing process. Others will be written during the manufacturing test phase.

Since each memory location can be written into only one time, the parameter storage area supports multiple write cycles allowing multiple writes of a given parameter value. 16 parameter rewrites are available for the TM100 and TM200.

The OTP memory has a read temperature range of -40C to +125C, with a programming temperature range of 0C to 50C. For best programming results when writing OTP code sets, the VDDA supply voltage should be set to the maximum value, 3.465V.

Registers

The TMx00 hardware functions are controlled via registers. When the 8051 program determines that a hardware block needs to be updated, it writes the desired value to the appropriate register. That value does not need to be refreshed until the program determines a new value is needed, or the VDDA power cycles. Also, I^2C commands can be directed to specific register functions.

CPU Clock

On startup, the µP operates from an internal ring oscillator running at nominally 10MHz. For some applications, the µP clock source is switched from the internal ring oscillator from the crystal oscillator (OSC). Internal dividers allow selection of the OSC derived clock, so it remains in the range of 1MHz to 10MHz.

The OSC detector contains an integrating circuit to detect stable oscillator operation. The processor reads a flag from the detector circuit and controls the switch to the main oscillator as commanded by software. If the main oscillator fails, the µP will automatically switch back over to the internal ring oscillator.

The ring oscillator can be disabled under µP control when the main oscillator is stable. It starts up if the main oscillator fails.

ADC Clock

The A/D clock is derived from the main clock via an integer division (÷1 through 31) for operation at 2MHz or less. The A/D clock maximum rate is 2MHz for reliable sampling in the A/D core. The desired operating rate is 1MHz nominal.

POWER-ON

Supply Ramp Rate

A ROM startup routine evaluates the VDDA voltage and delays the start of OTP access until the VDDA voltage is sufficient for reliable OTP reads (2.97V). This action ensures the clean startup of OTP reads. In addition, an overall timeout counter prevents the ROM code from hanging if the supply voltage is below the ROM defined threshold. In that case the code does not jump into the OTP and responds only to commands contained in the ROM.

Power supply ramp times of up to 1 second (0V to 3.3V) are supported in the TM100 and TM200.

Startup Process

After the VDDA voltage is stable and in specification, the 8051 loads the current parameter page from OTP into RAM. The values in RAM are then used to set the latches to the desired conditions. Loading the values into RAM makes production testing easier since there is no need to write the OTP contents until the final parameter setup items are determined.

The 8051 processor then starts a set of Real Time Operating System (RTOS) tasks that support temperature correction, parameter monitoring, clock switching, and other tasks.

2 I²C COMMUNICATIONS

Table 2-1 I²C Communications Connections

Item	Data
Interface Type	I²C Client 3.3V open drain. External pullups <= 10kΩ required when active
Maximum Data Rate	100kbps
Address	5A hex
Data Pin	SDA
Clock Pin	SCL

All communications start from a master that sends the appropriate command packet to the 8051. Then after each sent packet, the 8051 initiates a readback for the corresponding number of bytes in the command. This readback checks for errors along with any information. The 8051 sends back the command byte as the first readback byte when communication occurs without errors. If there is an error, the 8051 sends back an error packet.

The I²C interface is a client, addressed at 0x5A. A checksum byte is appended after all bytes in a packet before the stop sequence. This checksum uses a single byte modular sum to ensure that there are no bit errors during communication. The checksum is computed to be the two's complement of the sum of all bytes not including the 0x5A address byte.

If a message contained data of 0x88 0x10 0xA5, the checksum would be computed as: 0x100 – 0xA5 – 0x10 – 0x88, or 0xC3. Anything above the least significant byte is discarded.

FIRMWARE VERSIONS

This document is applicable to TM100 and TM200 firmware versions 1.8 or higher.

COMMUNICATIONS COMMANDS

Packet Communication Format:

Send: Packet sent from server to client.

Recv: Packet sent back from client to server.

- The number of data bytes sent and received is limited by the data buffer size, 0x20.
- The minimum size for a read back command is 3 bytes. That ensures an error code is always available if needed.
- When requesting a read for more bytes than possible, every byte after the checksum byte will be read as 0xFF. This is due to the device shutting off communication after sending the checksum.
- The command byte of 0x00 is reserved to catch error conditions. Reading back 0x00 from either the I²C bus or the single wire bus means that there is no relevant information to send.

For example, the report IC command is:

Send (to TM100/200): 0x01, 0xFF

The expected (correct) response:

Recv (from TM100/200): 0x01, 0x05, 0x5A

I²C Command Definitions

When needed, addresses are sent from the master in two bytes, addr_low followed by addr_hi. The parameter num_bytes (0x01 to 0x20) defines how many bytes to transfer in or out. Values outside the defined num_bytes range are rejected.

Relative addressing is defined as shown in the commands.

Block Read/Write Error Returns

The error function is returned instead of the normal received byte when the microcontroller detects an error in the function. For example, if the command requests or tries to set more bytes than available in the buffer (0x20), or the number of bytes is 0, the error function is invoked with a non-zero return code.

Recv: 0xFF, error_code, Checksum

Table 2-2 Error Code Returns

Error Code	Hex Value
Data Buffer is Busy	0x11
overflow Error	0x12
Command Error	0x13
Incomplete Operation Error	0x14
Checksum Error	0x15
Byte Count Error	0x16
Buffer Transfer Error	0x17
Parameter Rewrite Error	0x21
OTP Write, Parameter Write, Static Parameter Write	
Bit Error, OTP bit already 1 when program command calls out 0	0xFE
Cannot program byte after full soak pulse sequence	0xFF

General Commands

Report IC Code

This command returns the IC Code. It does not reflect the ROM code revision.

Send: 0x01, Checksum
Recv: 0x01, Code, Checksum

Table 2-3 IC Code

Value Returned	
05h	TM100/200 IC Code

Set CMOS Out

Send: 0x10, Desired State, Checksum
Recv: 0x10, 0x00 (operation successful) or 0x<XX> (operation failed), Checksum

Table 2-4 cmos_out Programming Map

Bit	Name	Description		Conditions
<0>	Output Transition Speed	**RFOUT CMOS Transition Speed**		
		0	8 ma Output Driver Active	
		1	4 ma Output Driver Active	
<1>	EN Pullup/Pulldown Sense	**EN Pullup/Pulldown**		
		0	Internal Pulldown on EN	
		1	Internal Pullup on EN	
<2>	EN Polarity	**EN Polarity**		
		0	RFOUT active when EN is low, tristate when EN is high	
		1	RFOUT active when EN is high, tristate when EN is low	
<3>	Digital Test Mode	**Digital Test Mode**		
		0	Normal, Regular signal path to RFOUT	Set to 0 for normal operation
		1	Test mode, internal test signals fed to RFOUT	
<4>	Internal Test	**Internal Test**		
		0	Leave at 0	
<7:5>	Test Signal at RFOUT pin	**Digital Test Mode Input (selected when bit 3 is "1")**		
		0	Idle (Low output)	
		1	Ring Oscillator Output	Valid when the ring oscillator is the µP clock source
		2	OCXO Divided Down clock	
		3	A/D Clock	
		4-7	Not used, do not select	

MEMORY BLOCK ACCESS COMMANDS

OTP Setup Block Commands

The TM100/200 power on device setups are stored in OTP setup blocks, each of which is 256 bytes long. A total of 16 setup spaces are available for usage. At power up, the IC reads the first non-blank setup page encountered. The page contents are moved to RAM memory storage and loaded into the relevant device setup registers as needed.

The internal RAM setup block contents can be set dynamically to allow testing of different configurations without the need for using an OTP block during oscillator setup testing. After the configuration testing is completed successfully, the internal RAM setup block can be moved to the next blank OTP setup block. At the next power-on cycle, that new setup block sets the IC parameters.

Write New OTP Setup Block

The process of writing a new OTP setup page has the following steps:

1. Fill RAM Setup Block with the needed values. The write into the OTP Setup Block will use those values.
2. Enable OTP Write into (OTP) Setup Block
3. Read OTP Setup Block Number and ensure that a blank setup page is free (with step 4).
4. Check blank setup page status. If it is blank, go to 6., otherwise go to 5.
5. Set Next OTP Setup Block.
6. Write OTP Setup Block, repeat as needed for the data set.
7. Validate proper data written into setup page by cycling power (as needed).

Enable OTP Write into Setup Block

This command enables OTP writing into the setup blocks. Once this command is issued, writing is enabled into the setup blocks until the power cycles.

Send: 0x1B, 100µs divisor low byte, 100µs divisor high byte, 400µs divisor low byte, 400µs divisor high byte, Checksum
Recv: 0x1B, 0x00, Checksum

The table below shows the OTP divisor values used for the default 10 MHz IC clocking.

Table 2-5 Default Divisor Values

OTP Divisor	Value
100µs divisor low byte	0x1A
100µs divisor high byte	0x04
400µs divisor low byte	0x68
400µs divisor low byte	0x10

Write OTP Setup Block

This command writes the OTP setup block location pointed to by the current setup block location.

Send: 0x31, offset_low, offset_high, num_bytes, DATA, Checksum
Recv: 0x31, 0x00, Checksum

If the address plus the number of bytes exceeds 256, the received data will indicate an error.

Read Current OTP Setup Block Number
This command returns the current setup block number. The possible values start at 0 and continue through 0x0F.

Send: 0x0B, Checksum
Recv: 0x0B, Number (byte), Checksum

Check OTP Blank Setup Page
This command checks the blank status of the current OTP setup block number. If the setup page is blank, the two data bytes returned are zero. Otherwise, the page already has data filling the block.

Send: 0x1D, Checksum
Recv: 0x1D, non-blank offset low, non-blank offset high, Checksum

Set Next OTP Setup Block
This command sets the OTP Setup Block pointer to the next (open) location.

Send: 0x09, Checksum
Recv: 0x09, 0x00, Checksum

Read OTP Setup Block
This command reads the OTP setup block location pointed to by the current setup block location.

Send: 0x21, offset_low, offset_high, num_bytes, Checksum
Recv: 0x21, DATA, Checksum

If the address plus the number of bytes exceeds 256, the received data will indicate an error.

RAM Setup Block Commands
See Section 3 for RAM Setup Block Commands

User Block Commands
The User Block is a 128-byte user accessible block of OTP that can be used for storing and retrieving items such as part numbers, serial numbers, operating frequencies, or other data entries. Each location can be written only one time and read as many times as needed.

The unprogrammed state of the User Block bytes are all zeros.

Read User Block
This command reads the contents of the user block in the TMx00 OTP.

Send: 0x25, offset_low, offset_high, num_bytes, Checksum
Recv: 0x25, DATA, Checksum

If the address plus the number of bytes exceeds 128, the received data will return all zeros.

Write User Block
This command writes the contents of the user block in the TMx00 RAM.

Send: 0x26, offset_low, offset_high, num_bytes, DATA, Checksum
Recv: 0x26, 0x00, Checksum

FUNCTIONAL COMMANDS

General Use Commands

Software Reset

This command forces a software reset of the processor.

Send: 0x0D, Checksum
Recv: 0x0D, 0x00, Checksum

The return may not be present since the routine executes an immediate software reset.

ADC Sample and Correction Intervals

This command sets the ADC sample and frequency correction intervals.

Each interval is measured in units of the RTOS timer ticks (10ms each) in the range of 2 to 255 for the sample interval and 3 to 255 for the frequency update interval. For example, a sample interval of 10 indicates a sampling rate of 100 per second. The freq_update_interval specifies how often the varicap or external varactor control voltage is updated to adjust the oscillator frequency.

Send: 0x11, sample_interval, freq_update_interval, Checksum
Recv: 0x11, 0x00, Checksum

The minimum value for freq_update_interval is 3. If lower values are entered, the TM100/200 will set the freq_update_interval to 3.

This command modifies the setup page variables temp_volt_sample_interval and freq_update_interval. See Section 3 for more information.

Reading ADC Channels

Start ADC

This command starts an ADC conversion.

Send: 0x14, ADC_channel, ADC_delay, Checksum
Recv: 0x14, 0x00, Checksum

ADC_channel specifies the desired ADC input to be read.

Firmware Version 1.9 and higher

See the ADC read command 0x15 for return value details.

Table 2-6 ADC Channel Programming Map

Name	Description		Conditions
ADC Channel Select	**Channel & Signal**		
	0 (0h)	Thermistor Input - THRM	A full-scale code of 4095 is equivalent to 2.9V
	1 (1h)	ADC VREF	
	2 (2h)	MUX (TM100) External Heater Feedback - HFB (TM200)	
	3 (3h)	Heater 1 Internal Heater Feedback (TM200) (Main feedback from Heater 1 sense resistor)	
	4 (4h)	Heater 2 Internal Heater Feedback (TM200) (Matching terminal of Heater 2 sense resistor)	
	5 (5h)	Internal Temperature Sensor	
	6 (6h)	External & Internal Heater Drive (TM200)	
	7 (7h)	EFC (scale factor 1, up to 2.9V)	
	8 (8h)	VDDA Supply, divided by 2	
	9 (9h)	DAC VREF, divided by 2	
	10 (Ah)	Varicap/Tune DAC	
	11 (Bh)	Temperature Set DAC (TM200)	
	12 (Ch)	Current Limit DAC (TM200)	
	13(Dh)	Digital Supply Voltage (nominally 1.8V)	
	14 (Eh)	Not used	
	15 (Fh)	No internal channels connected	
	16(10h)-255 (FFh)	Not Valid	

ADC_delay specifies the settling time for the ADC conversion, in units of 4x the ADC clock time. For example, a value of 10 (decimal) would give a settling time of 40µs assuming a default A/D clock rate of 1MHz.

Table 2-7 ADC_delay Programming Map

Name	Description		Conditions
ADC Mux Delay	**ADC Mux Settling Time**		
	0-1	Not Valid	
	2-255 (2h-FFh)	Mux and Sample Settle Time	In 4x A/D Clock Units
	2		
	..		
	..		
	255 (FFh)		

Read ADC

This command reads the results of an ADC conversion, cooperatively with the RTOS ADC interrupt routines.

Send: 0x15, Checksum
Recv: 0x15, data_low, data_high, channel_number, Checksum

The results will be a 0-4095 digital value in the two data bytes. The number of the channel measured will be in channel_number.

ADC Busy Response, Firmware Version 1.9 and higher

The microprocessor prioritizes ADC read operations that support voltage and temperature sensing to ensure valid correction data is always available. The 0x14 command ADC start may attempt to perform an ADC start when the microprocessor is busy. If so, the 0x15 read ADC command will return with 0xFF for data_low & data_high, and 0xF0 for the channel_number. In that condition, re-try the 0x14 Start ADC and 0x15 Read ADC. When the ADC is idle, the sequence will complete with valid data for data_low and data_high (0x0000-0x0FFF, 0-4095) and the channel_number will be valid (0x00-0x0F, 0-15).

ADC Reading Example

The ADC values read for channels 7 through 13 use a full-scale value of 2.9 for a digital code of 4095.

For illustration, use channel 13 (digital supply). Assuming the supply is exactly 1.8V, the reading will be:

$$ADC\ Code = \frac{1.8}{2.9} * 4095 = 2542$$

We wish to read the voltage of the digital supply (channel 13, delay of 100). Start the ADC conversion by issuing the following I^2C command:

Send to the TMx00: 0x14, 0x0D, 0x64, 0x7B
Receive from the TMx00: 0x14, 0x00, 0xEC

This starts the ADC conversion. Then read the results of the conversion, assuming the 2542 shown above is the result of the conversion:

Send to the TMx00: 0x15, 0xEB
Receive from the TMx00: 0x15, 0xEE, 0x09, 0x0D, 0xE7

0x9EE is the hex equivalent of 2542, and 0x0D is the channel number read (13, digital supply)

Oscillator Switch

This command switches between the clock modes. The clock mode returned reflects the status of the clock after the switchover. Currently the clock switchover time is up to 2 seconds.

Send: 0x16, Clock_Mode, Checksum
Recv: 0x16, Clock_Mode, Checksum

Table 2-2-8 Clock Mode Switching

Clock_Mode	Function
0x00	Ring Oscillator Clock Active
0x01	OSC Clock Active, Ring Oscillator Running
0x02	Ring Oscillator Clock Active
0x03	OSC Clock Active, Ring Oscillator Not Running

Read Revisions

This command reads the firmware revision contained in the TMx00.

Send: 0x1A, Checksum
Recv: 0x1A, ic_code, part_number, firmware_major_rev, firmware_minor_rev, Checksum

ic_code is the Silicon IC code (0x5), part_number is 1 for TM100 or 2 for TM200. The next two bytes contain the firmware major and minor revision numbers.

Read Correction Data Items

This command reads items related to the IC temperature correction operations.

Send: 0x1C, Checksum
Recv: 0x1C, Tuning DAC Value High, Tuning DAC Value Low, Temperature Sensor or THRM Average High, Temperature Sensor or THRM Average Low,
VDDA Average High, VDDA Average Low,
VDDA Voltage (4 bytes, float),
IC Temperature (4 bytes, float), Voltage Correction (4 bytes, float) Checksum

The Tuning DAC Value is the 12-bit input into the tuning DAC, 0 to 4095 range.

If the temperature sensor is active, the next 2 bytes contain the 12-bit value from the sensor. If the THRM input is active, those bytes contain the 12-bit value from the THRM input. If neither correction nor sensing modes are active, those two bytes contain 0xFFFF. In addition, the most significant bit of the THRM Average High byte is set high if the THRM correction is active. If the bit is low, the temperature sensor is the source of the data.

The VDDA voltage is the floating-point representation of the input supply voltage. The IC Temperature is the measured temperature floating-point value in degrees C.

The voltage correction is the floating-point representation of the VDDA correction. If the voltage correction order is 1 or 2, the value is the temperature code delta summed with measured temperature code value. If the voltage correction order is 4 or 5, the value is the adjusted net correction code (or voltage) for the measured supply value.

Read Internal Heater Resistor
TM200 Only

This command reads the single-precision floating point value of the internal heater resistor.

Send: 0x1E, Checksum
Recv: 0x1E, Heater Resistor Value (4 bytes, float), Checksum

The nominal value of the heater resistor is 26 Ω.

Read Temperature Sensor Scale and Offset
This command reads the temperature sensor scale and offset values defined at IC test. These values are used for IC temperature calculation when the matching setup page variables are 0.0.

It also reads the VDDA code that represents 3.3 V supply as measured at IC test.

Send: 0x1F, Checksum
Recv: 0x1F, Temperature Sensor Scale (4 bytes, float), Temperature Sensor Offset (4 bytes, float), VDDA test value (2 bytes, unsigned integer), Checksum

The detected IC temperature in degrees C is given by:

$$degreesC = (Temp\ Code * Temperature\ Sensor\ Scale) + Temperature\ Sensor\ Offset$$

The resultant value is not used for internal firmware calculations. It is an indicator for the TMx00 user.

Latch Based Commands

Read Latch
This command reads a byte from the defined latch as shown in the table.

Send: 0x02, Index, Checksum
Recv: 0x02, DATA, Checksum

Table 2-9 Latch Read

Name	Index	Notes
clk_sel_latch	0x09	

Write Latch
The write latch command allows the user to write latches in the TM100/200 microcontroller.

Send: 0x03, index, data_low, data_high, checksum
Recv: 0x03, 0x00, checksum

These commands expect sixteen bits of data, regardless of the latch width.

Table 2-10 Write Latch

Name	index	Notes
osc0_latch	0x00	
osc1_latch	0x01	
osc2_latch	0x02	
analog0_latch	0x03	
therm_latch	0x04	Not used for TM100, enabled for TM200.

temp_latch	0x05	Not used for TM100, enabled for TM200.
tuning_latch	0x06	
bias_latch	0x07	
cmos_out_latch	0x0A	
osc_div_latch	0x0D	
clk_sel_latch	0x0E	
–	0x0F	Not used for TM100 or TM200
adc_div_latch	0x10	

osc0_latch Programming Definitions

Table 2-11 osc0_latch Programming Map

Bit	Name	Description		Conditions
<1:0>	Thermistor Range Set Trim	**Set Temperature Range Trim Bits**		
		0	9 KΩ || 14.2 KΩ	**TM200 Only, N/A for TM100** These bits select resistors parallel to the ones selected by therm_latch<6:4>
		1	14.2 KΩ	
		2	9 KΩ	
		3	Open	
<8:2>	CF Capacitor Control Bits	**Set CF Capacitance Value**		
		0	Stray Capacitance (~ 5 pF)	Bit 0: 2 pF Bit 1: 4pF Bit 2: 8 pF Bit 3: 16 pF Bit 4: 32 pF Bit 5: 64 pF Bit 6: 64 pF
		1	2 pF + Stray	
		2	4 pF + Stray	
		…		
		127	190 pF + Stray	
<13:9>	CV Capacitor Control Bits	**Set CV Capacitance Value**		
		0	Stray Capacitance (~2 pF)	Bit 8: 1 pF Bit 9: 2 pF Bit 10: 4 pF Bit 11: 8 pF Bit 12: 16 pF
		1	1 pf + Stray	
		2	2 pF + Stray	
		…		
		31	31 pF + Stray	
<15:14>	N/A	**Not Used**		

osc1_latch Programming Definitions

Table 2-12 osc1_latch Programming Map

Bit	Name	Description		Conditions
<3:0>	Pierce CA Capacitor Control Bits	**CA Capacitance Value**		Bit 0: 5 pF Bit 1: 10 pF Bit 2: 15 pF Bit 3: 20 pF
		0	Stray (~2 pF) capacitance	
		1	5 pF + Stray	
		2	10 pF + Stray	
		…		
		15	50 pF + Stray	
<7:4>	Pierce CB Capacitor Control Bits	**CB Capacitance Value**		
		0	Stray (~2 pF) capacitance	Bit 4: 5 pF Bit 5: 10 pF Bit 6: 15 pF Bit 7: 30 pF
		1	5 pF + Stray	
		2	10 pF + Stray	
		…		
		15	60 pF + Stray	
<10:8>	Pierce RF Resistor Control Bits	**Pierce Feedback Resistor (RF) value**		
		0	100KΩ	
		1	1.6KΩ	
		2	2.0KΩ	
		3	2.6KΩ	
		4	3.4KΩ	
		5	4.5KΩ	
		6	Open	Use External RF
		7	Open	
<13:11>	Pierce RD Resistor Control Bits	**Pierce Series Phase Shift (RD) Resistor**		
		0	25Ω	
		1	50Ω	
		2	100Ω	
		3	150Ω	
		4	open	Use External RD
		5	1000Ω	
		6	400Ω	
		7	500Ω	
<15:14>	N/A	**Not Used**		

osc2_latch Programming Definitions

osc2_latch controls the EFC/Tuning DAC/Varicap/XTUNE steering and bias circuitry.

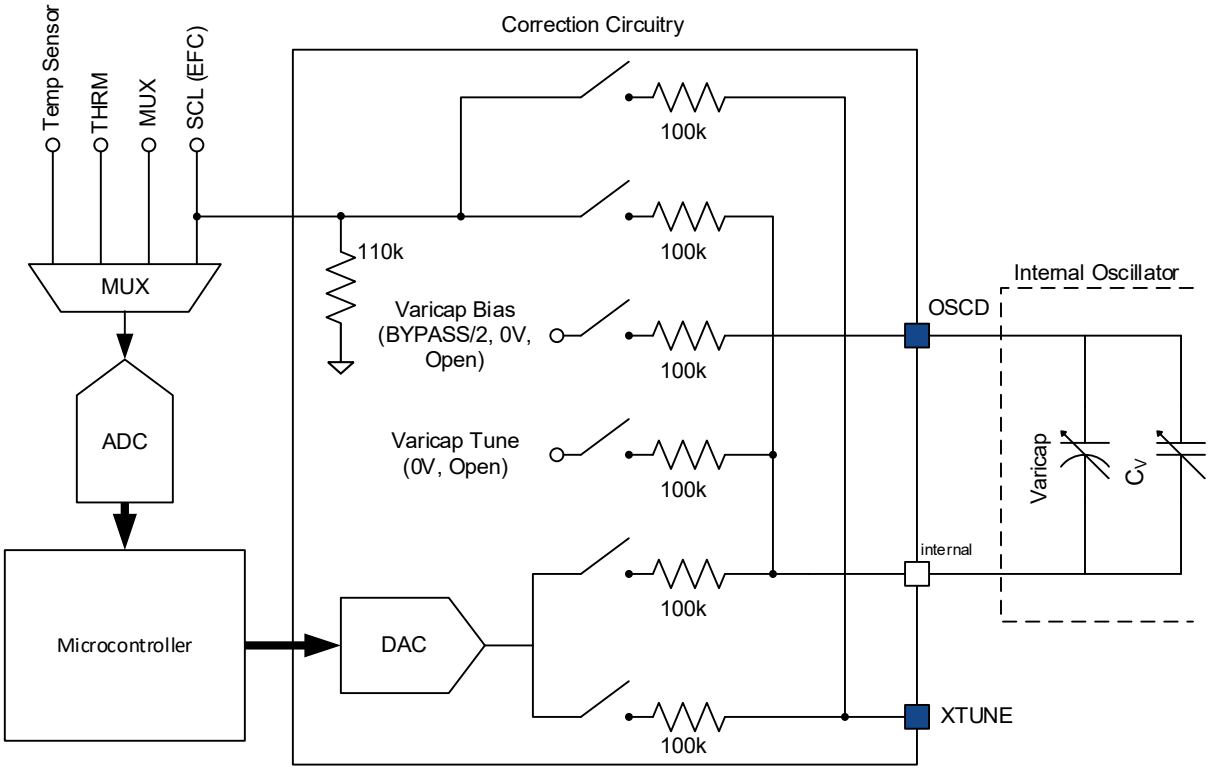

Figure 2-1 EFC/Tuning DAC/Varicap/XTUNE Circuit Block Diagram

The TM200 has only one MUX input, called “THRM”.

Table 2-13 osc2_latch Programming Map

Bit	Name	Description		Conditions
<0>	EFC to XTUNE	**EFC - XTUNE**		
		0	EFC to XTUNE disconnected	Varicap minimum capacitance occurs when Varicap Tune is 1.45 V more positive than Varicap Bias
		1	EFC connected to XTUNE via a 100KΩ resistor	
<1>	DAC to XTUNE	**DAC - XTUNE**		
		0	DAC to XTUNE disconnected	
		1	DAC connected to XTUNE via a 100KΩ resistor	
<2>	EFC to Varicap Tune	**EFC - Varicap**		
		0	EFC to Varicap disconnected	Varicap minimum capacitance occurs when Varicap Tune is 1.45 V more positive than Varicap Bias
		1	EFC connected to Varicap Tune via a 100KΩ resistor	
<3>	DAC to Varicap Tune	**DAC - Varicap**		
		0	DAC to Varicap disconnected	Varicap minimum capacitance occurs when Varicap Tune is 1.45 V more positive than Varicap Bias
		1	DAC connected to Varicap Tune via a 100KΩ resistor	
<4>	Varicap Bias Enable	**Varicap Bias Enable**		
		0	Varicap Bias = $V_{BYPASS}/2$	1.45V for internal BYPASS regulator active
		1	Varicap Bias not fed (floating)	
<5>	Varicap Zero Bias	**Varicap Zero Bias**		
		0	Normal Varicap Bias as set by other bits	
		1	Varicap Bias and Varicap Tune voltages set to 0V	Bit 4 (Varicap Bias Enable), bits 0-3 should be zero when this bit is set
<15:6>	N/A	**Not Used**		

Table 2-14 XTUNE Signal Sources

osc2_latch Bit 1	osc2_latch Bit 0	XTUNE Signal	Conditions
0	0	Open	
0	1	EFC→XTUNE	Connection via 100 KΩ resistor
1	0	DAC→XTUNE	Connection via 100 KΩ resistor
1	1	EFC & DAC→XTUNE	Each connected via 100 KΩ resistor

Table 2-15 Varicap Biasing Voltages

osc2_latch Bit 5	osc2_latch Bit 4	Varicap Bias	Conditions
0	0	$V_{BYPASS}/2$	1.45V when V_{BYPASS} is 2.9V, $V_{BYPASS}/2$ otherwise
0	1	Varicap Bias Not Fed (Floating)	
1	1	Varicap Bias (and Varicap Tune) Set to 0V	
1	0	Varicap Bias (and Varicap Tune) Set to 0V	Do not use, extra 0.3 ma current drain

Table 2-16 Varicap Tuning Voltages

osc2_latch Bit 5	osc2_latch Bit 3	osc2_latch Bit 2	Varicap Tune	Conditions
0	0	0	EFC→Varicap Tune	Connection via 100 KΩ resistor
0	0	1	DAC→Varicap Tune	Connection via 100 KΩ resistor
0	1	1	EFC & DAC→Varicap Tune	Each connected via 100 KΩ resistor
1	0	0	Varicap Tune (and Varicap Bias) Set to 0V	
1	0	1	Varicap Tune (and Varicap Bias) Set to 0V	Do not use, loads DAC output
1	1	0	Varicap Tune (and Varicap Bias) Set to 0V	Do not use, loads EFC output
1	1	0	Varicap Tune (and Varicap Bias) Set to 0V	Do not use, loads DAC & EFC output

analog0_latch Programming Definitions

Table 2-17 analog0_latch Programming Map

Bit	Name	Description		CONDITIONS
<1:0>	Output Divider	**Division Value**		
		0	÷1	
		1	÷2	
		2	÷4	
		3	÷8	
<4:2>	Duty Cycle Trim Value	**RFIN Bias Voltage**		
		0	1.45	
		1	1.50V	
		2	1.55V	
		3	1.60V	
		4	1.65V	
		5	1.70V	
		6	1.75V	
		7	1.8V	
<5>	Pre-driver Enable	**Sets Driver to defined off state, enables/disables RFIN pre-driver (Path to RFOUT is non-inverting)**		
		0	Input passes through to output	
		1	Pre-driver output set low, RFIN input is disabled (~ 25kΩ to ground)	
<6>	Thermal Controller Maximum Drive Current	**Selects Maximum Drive Current for External Heater Drive**		
		0	Bipolar Drive Mode: Maximum Drive Current 8.2 ma nominal (7 ma minimum)	**TM200 Only**
		1	FET Drive Mode: Maximum Drive Current 500µA nominal (420 µA minimum)	
<7>	OSC Clock Switchback	**Controls Automatic OSC to Ring Oscillator Switchback**		
		0	Automatic Switchback from OSC µP to Ring Oscillator when OSC fails	
		1	Disable Automatic Switchback from OSC µP to Ring Oscillator when OSC fails	

therm_latch Programming Definitions

TM200 Only

Table 2-18 therm_latch Programming Map

Bit	Name	Description		CONDITIONS
<2:0>	Thermal Controller Feedback Gain	**Feedback Value ("L" network)**		
		0	10KΩ Series, 562Ω to ground	
		1	7.5KΩ Series, 562Ω to ground	
		2	5.0KΩ Series, 562Ω to ground	
		3	2.5KΩ Series, 562Ω to ground	
		4	1KΩ Series, 562Ω to ground	
		5	500Ω Series, 562Ω to ground	
		6	200Ω Series, 562Ω to ground	
		7	22 Ω Series, 562Ω to ground	
<3>	Enable Thermal Controller	**Thermal Controller**		
		0	Thermal Controller disabled	
		1	Thermal Controller enabled	Bit 3 of bias_latch (BYPASS regulator) must be set, or an external voltage source connected to BYPASS to allow Thermal Controller operation
<6:4>	Thermistor Range Set	**Sets temperature range for thermistor operation**		
		0	Open (for use with an external resistor between BYPASS and HTHERM, or only the trim resistors)	osc0_latch <1:0> control trim resistors in parallel with these units
		1	Open (for use with an external resistor between BYPASS and HTHERM, or only the trim resistors)	
		2	6.9kΩ	
		3	5.8kΩ	
		4	5.0kΩ	
		5	4.4kΩ	
		6	4.1kΩ	
		7	3.7kΩ	
<7>	Internal/External Heater Select	**External/Internal Heater**		
		0	External Heater and Drive Transistors Selected	
		1	Internal Heater and Drive Transistors Selected	
<8>	Temp Drive Opamp (Temperature Control DAC) Enable	**Opamp (Temperature Control DAC) Enable**		The opamp is the final stage of the Temperature Control DAC function
		0	Temp drive opamp (Temperature Control DAC) disabled	
		1	Temp drive opamp (Temperature Control DAC) enabled	
<15:9>	Current Limit DAC	**Sets driver stage current limit**		**Controls voltage into current limit amplifier**

		0	0	For external heaters, the useful range is 0 - 0.4V For internal heaters, the useful range is 0 – 0.65V
		1		
		…		
		127	Full scale (2.9 V)	

temp_latch Programming Definitions

TM200 Only

Table 2-19 temp_latch Programming Map

Bit	Name	Description		Conditions
<11:0>	Temperature Control DAC	**D/A value**		
		0	0V	
		…	…	
		…	…	
		4095	2.9V full scale	
<15:12>	Not Used	**Not Used**		

tuning_latch Programming Definitions

This latch sets the input for the frequency set DAC, fed to either the internal Varicap or an external varactor.

Table 2-20 tuning_latch Programming Map

Bit	Name	Description		Conditions
<11:0>	Tuning DAC Voltage	**Output Voltage**		
		0	0V	
		1		
		…		
		4095	V_{BYPASS}	2.9V for internal regulator
<15:12>	N/A	**Not Used**		

bias_latch Programming Definitions

Table 2-21 bias_latch Programming Map

Bit	Name	Description		Conditions
<0>	Enable DAC Reference	**DAC Reference**		
		0	Disabled	Must be enabled for thermal controller operation
		1	Enabled	
<1>	Enable ADC Reference	**ADC Reference**		
		0	ADC Reference disabled	
		1	ADC Reference enabled	
<2>	Enable ZTC Current References	**ZTC Current Sources (for ADC operation)**		
		0	ZTC disabled	Must be enabled for ADC operation
		1	ZTC enabled	
<3>	Enable Oscillator (2.9V BYPASS) Regulator	**Oscillator Regulator (BYPASS)**		
		0	BYPASS Regulator disabled	Disable this regulator if an external reference provides the 2.9V BYPASS voltage. The 2.9V internal or external regulator is also used by the thermal controller.
		1	BYPASS Regulator enabled	
<15:4>	Not Used	**Not Used**		

cmos_out_latch Programming Definitions

Table 2-22 cmos_out_latch Programming Map

Bit	Name	Description		Conditions
<0>	Output Transition Speed	**RFOUT CMOS Transition Speed**		
		0	Higher Drive (8 ma) RFOUT	
		1	Lower Drive (4 ma) RFOUT	
<1>	EN Pullup/Pulldown Sense	**EN Pullup/Pulldown**		
		0	Internal Pulldown on EN	
		1	Internal Pullup on EN	
<2>	EN Polarity	**EN Polarity**		
		0	RFOUT active when EN is low, tristate when EN is high	
		1	RFOUT active when EN is high, tristate when EN is low	
<3>	Digital Test Mode	**Digital Test Mode**		
		0	Normal, Regular signal path to RFOUT	Leave bit at 0 for RFOUT operation
		1	Test mode, internal test signals fed to RFOUT	If RFOUT needs to be disabled, set this bit to 1 and set Digital Test Signal to 0x0.
<4>	Internal Test	**Internal Test**		
		0	Internal Test	Leave bit at 0, do not set to 1.
<7:5>	Digital Test Signal	**Digital Test Mode Input (selected when bit 3 is “1”)**		
		0	Idle (Low output)	
		1	Ring Oscillator Output	Valid when the ring oscillator is the µP clock source
		2	OSC Divided Down clock	
		3	A/D Clock	
		4-7	N/A	

CLOCK SOURCE SWITCHING AND OSC DIVISION LATCHES

The CPU operates from two clock sources. When the unit starts, an internal ring oscillator operating at ~10MHz provides the initial clock signal to the CPU. When the OSC is stable, the processor can command a clock switchover to the OSC clock.

The OSC could operate as high as 100MHz, while the CPU design supports up to 10MHz functionality. An internal divider reduces the frequency as needed under processor control with division rates of 1 to 31. The divider is synchronous so all positive going CPU clocking transitions occur on the positive going edge of the OSC clock. When the divisor is odd, a control bit allows setting the low going CPU clock transitions to the low going transition of the OSC clock, or to the immediately following positive OSC transition.

The clock switching uses digital synchronizers and will always occur glitch-free.

osc_div_latch Programming Definitions

A write sets the division value and the low going transition flag. This register does not change the CPU clock source.

Table 2-23 osc_div_latch Programming Map

Bit	Name	Description		Conditions
<4:0>	OSC Division	**Divisor for OSC to CPU clock**		
		0	Not valid	
		1	÷1	
		2	÷2	
		…		
		31	÷31	
<5>	Odd Divisor Negative Edge	**CPU H-L Transition When Divisor is Odd**		
		0	H-L Output Transition coincident with L-H OSC Transition	
		1	H-L Output Transition coincident with H-L OSC Transition	
<7:6>	Not Used	**Not Used**		

clk_sel_latch Programming Definitions

A write to this latch controls the CPU clock source. The OSC cannot be selected as a clock source when the OSC source detector indicates no signal is present. If the OSC detector toggles to indicate no OSC clock present, the CPU will operate from the ring oscillator.

A read of this latch will show the status of the CPU clock source, bit 3 – bit0.

Bit 7 of the analog0 register controls enabling and disabling automatic switchback when the OSC clock source is selected and the OSC clock fails.

clk_sel_latch is not stored in the internal setup block or the OTP setup block.

Table 2-24 CPU Clock Source Selection

Bit	Name	Description		Conditions
<0>	Commanded CPU Clock Source	**Commanded OSC/Ring Oscillator Select**		This bit represents the state of the last register write command. However, if the OSC signal goes away or is not present, the read value will revert to 0 and the µP clock will be driven from the ring oscillator
		0	Select Ring Oscillator	
		1	Select OSC	
		In Read mode	0 if the Ring oscillator is selected, 1 if the OSC is selected	
<1>	OSC Detect	**OSC Detector, latched (only in read mode)**		
		0	OSC not active, or OSC has failed and switched back to ring oscillator	
		1	OSC active (after commanded by SFR write with bit 0 high	
<2>	Active CPU Clock Source	**OSC/Ring Oscillator Active as CPU clock (only in read mode)**		This bit shows the CPU active clock source
		0	Ring Oscillator Clock	
		1	OSC Active	
<3>	Unlatched OSC Detect	**Unlatched OSC detect (only in read mode)**		
		0	OSC not active (independent of latch state)	
		1	OSC Active	
<4>	Ring Oscillator Disable	**Ring Oscillator Enable/Disable**		This function enables or disables the ring oscillator without switching the µP clock. If the OSC is not active, the ring oscillator cannot be disabled.
		0	Ring Oscillator Enable	
		1	Ring Oscillator Disable	
<7:5>	N/A	**Not Used**		

adc_div_latch Programming Definitions

The A/D converter typically operates from a 1MHz sampling clock. This register sets the divide value from the CPU clock to the A/D clock.

Table 2-25 A/D Clock Division Programming Map

Bit	Name	Description		Conditions
<4:0>	A/D Division	**Divisor for CPU to A/D clock**		
		0	Not valid	
		1	÷1	
		2	÷2	
		…		
		31	÷31	
<5>	Odd Divisor Negative Edge	**A/D Clock H-L Transition When Divisor is Odd**		
		0	H-L Output Transition coincident with H-L CPU clock Transition	
		1	H-L Output Transition coincident with L-H CPU clock Transition	
<7:6>	Not Used	**Not Used**		

3 FIRMWARE/HARDWARE FUNCTIONS

INTERNAL SETUP STORAGE MEMORY

At power up, the 8051 loads the current parameter page from OTP into a setup page block in RAM. The values in RAM set latches and other items to the desired conditions. Loading the values into RAM makes production testing easier since there is no need to write the OTP contents until the final parameter setup items are determined.

Read RAM Setup Block Command

This command reads the contents of the setup block in the TMx00 RAM.

Send: 0x18, offset_low, offset_high, num_bytes, Checksum
Recv: 0x18, DATA, Checksum

The offset in the table below is the value used in the command. If the offset plus the number of bytes exceeds 255 (0xFF), the received data will indicate an error.

Write RAM Setup Block Command

This command writes the contents of the setup block in the TMx00 RAM.

Send: 0x19, offset_low, offset_high, num_bytes, DATA, Checksum
Recv: 0x19, 0x00, Checksum

The offset in the table below is the value used in the command. If the offset plus the number of bytes exceeds 255 (0xFF), the return value will indicate an error and the values will not be written into the setup block.

The latch definitions in the table are found in Latch Based Commands (above). The other items are defined in the following sections.

Storage size definitions:

uint: 16-bit unsigned integer
uchar: 8-bit unsigned character
float: 32-bit single precision floating point, IEEE 754 format

Table 3-1 Internal Setup Block Contents

Name	Setup Page Offset	Data type	Notes
osc0_latch	0x0000	uint	
osc1_latch	0x0002	uint	
osc2_latch	0x0004	uint	
analog0_latch	0x0006	uint	
therm_latch	0x0008	uint	**TM200 only**
temp_latch	0x000A	uint	**TM200 only.**
tuning_latch	0x000C	uint	
bias_latch	0x000E	uint	
cmos_out_latch	0x0010	uint	
adc_conv_ch_latch	0x0012	uint	
clk_setup_latch	0x0014	uint	
osc_div_latch	0x0016	uint	
adc_div_latch	0x0018	uint	
div_100us_latch	0x001A	uint	
div_400us_latch	0x001C	uint	
rtos_timer	0x001E	uint	
corr_order	0x0020	uchar	
corr_a0	0x0021	float	
corr_a1	0x0025	float	
corr_a2	0x0029	float	
corr_a3	0x002D	float	
corr_a4	0x0031	float	
corr_a5	0x0035	float	
corr_a6	0x0039	float	
corr_a7	0x003D	float	
corr_a8	0x0041	float	
corr_a9	0x0045	float	
temp_volt_sample_interval	0x0049	uchar	
freq_update_interval	0x004A	uchar	
osc_freq	0x004B	float	
corr_mean	0x004F	float	
corr_std	0x0053	float	
temp_sensor_scale	0x0057	float	
temp_sensor_offset	0x005B	float	
corr_order2	0x005F	uchar	
corr_c1	0x0060	float	c_1
corr_b1	0x0064	float	b_1
corr_b2	0x0068	float	b_2
corr_b3	0x006C	float	b_3
corr_b_mean	0x0070	uint	b_mean
corr_c2	0x0074	float	c_2
heater_startup_delay	0x0078	uchar	**TM200 only, N/A for TM100**
heater_soft_start	0x0079	uchar	**TM200 only, N/A for TM100**
lookup_table	0x007A	80 (decimal) locations of 3 x uchar	**Bit Mapping** Ttttttttt ttttDddd dddddddd T – MSBit of temp code t – less significant bits of temp code D – MSBit of DAC code d – less significant bits of DAC code
corr_c3	0x016A	float	c_3
ext_byp_volt	0x16E	float	

Setup Item Definitions

clk_setup

The clk_setup_latch controls the switchover to the OSC based clock after it is powered on. It does not affect clock generation at any other times.

The TMx00 always uses the internal ring oscillator at power up. If bit 0 is zero, the TMx00 remains on the ring oscillator. If bit 0 is one, the TMx00 will switch to the OSC based CPU as specified by the upper 12 bits of the clk_setup_latch. Those bits count in RTOS time increments of 10 ms. For example, a 2.5 second delay would be represented by 250 (0xFA) in the upper 12 bits.

rtos_timer

The CPU Real Time Operating System (RTOS) operates using a time interval of 10 ms. This time value is found by dividing the CPU clock frequency using the formula:

$$rtos\_timer = 0.01/(12.0 * (\frac{1e^{-6}}{CPU\ Clock\ Frequency\ (MHz)})$$

The internal ring oscillator operates at 10 MHz, and the default *rtos_timer* value is 8,333. This value needs to be updated if the CPU clock is derived from the OSC clock.

corr_order

The bottom four bits of this variable define the Temperature Correction order/mode. Polynomial correction values are 0 to 9, indicating the correction order chosen. If the bottom four bits (bits 3 – 0) are 10 (0xA), RAM-based table lookup replaces the polynomial correction.

If bit 4 is zero, the correction algorithms are disabled, and the temperature DAC code comes from the value in *tuning_latch*. The correction routines are active when bit 4 is one.

Testing Assist

If the bottom 4 bits of *corr_order* are 11 (0x0B), the temperature sensor inputs and VDDA sensor inputs are averaged to support more accurate measurements during the TMx00 oscillator characterization phase.

Voltage Correction

Table 3-2 Voltage Correction

Bits 7-5 (corr_order)	Correction or temp code?	Notes
0b000 (0)	–	No voltage/temperature correction
0b001 (1)	$Temp_{ADJ}(\Delta VDDA)$	$Temp_{ADJ}(\Delta VDDA) = \Delta_{VDDA} * corr\_b1$
0b010 (2)	$Temp_{ADJ}(\Delta VDDA)$	$Temp_{ADJ}(\Delta VDDA) = \Delta_{VDDA} * corr\_b1 + \Delta_{VDDA}^2 * corr\_b2 + \Delta_{VDDA}^3 * corr\_b3$
0b011 (3)	$Temp_{ADJ}(\Delta VDDA)$	$Temp_{ADJ}(\Delta VDDA) = \Delta_{VDDA} * corr\_b1 + \Delta_{VDDA}^2 * corr\_b2 + \Delta_{VDDA}^3 * corr\_b3 + \Delta_{VDDA} * tempCode_{norm} * corr\_c1 + \Delta_{VDDA}^2 * tempCode_{norm} * corr\_c2 + \Delta_{VDDA} * tempCode_{norm}^2 * corr\_c3$
0b100 (4)	–	No voltage/temperature correction
0b101 (5)	$CorrDAC_{ADJ}(\Delta VDDA)$	$CorrDAC_{ADJ}(\Delta VDDA) = \Delta_{VDDA} * corr\_b1$
0b110 (6)	$CorrDAC_{ADJ}(\Delta VDDA)$	$CorrDAC_{ADJ}(\Delta VDDA) = \Delta_{VDDA} * corr\_b1 + \Delta_{VDDA}^2 * corr\_b2 + \Delta_{VDDA}^3 * corr\_b3$
0b111 (7)	$CorrDAC_{ADJ}(\Delta VDDA)$	$CorrDAC_{ADJ}(\Delta VDDA) = \Delta_{VDDA} * corr\_b1 + \Delta_{VDDA}^2 * corr\_b2 + \Delta_{VDDA}^3 * corr\_b3 + \Delta_{VDDA} * tempCode_{norm} * corr\_c1 + \Delta_{VDDA}^2 * tempCode_{norm} * corr\_c2 + \Delta_{VDDA} * tempCode_{norm}^2 * corr\_c3$

When bits 6 – 5 equal zero, voltage correction is not active. When bits 6 – 5 equal 1, the corr_b1 value is used. When bits 6 – 5 equal 2, corr_b1, corr_b2, and corr_b3 are used in the calculations. When bits 6 – 5 equal 3, corr_b1, corr_b2, corr_b3, corr_c1, corr_c2, and corr_c3 are included in the trim calculations.

$$\Delta_{VDDA} = Normalized\ VDDACode_{Meas}\ with\ respect\ to\ VDDACode_{Nom}$$

$$VDDACode_{Nom} = VDDACode\ measured\ by\ IC\ when\ 3.3\ V\ is\ applied\ to\ VDDA$$

$$VDDA_{SD} = Standard\ deviation\ of\ VDDA\ set\ so\ that\ \pm 0.05\ wrt\ VDDACode_{3.3V}\ produces\ a\ \sigma\ of\ \pm 3$$

$$VDDA_{SD} = VDDACode_{NOM}\left(\frac{0.05}{3}\right)$$

$$\Delta_{VDDA} = \frac{VDDACode_{Meas} - VDDACode_{Nom}}{VDDA_{SD}}$$

$$TempCode_{norm} = temperature\ code\ normalized\ using\ corr\_mean\ and\ corr\_std$$

$$Temp_{adj} = temp\ code\ adjustment\ based\ voltage\ \&\ temperature\ measurements$$

$$\Delta TempCode = TempCode_{norm} + Temp_{adj}(\Delta VDDA)$$

$$\Delta CorrDAC\ Code = CorrDAC\ Code + CorrDAC_{ADJ}(VDDA)\ (Lookup\ Table)$$

$$\Delta CorrDAC\ Voltage = CorrDAC\ Voltage + CorrDAC_{ADJ}(VDDA)\ (Polynomial\ Correction)$$

Temperature Sensing

The crystal temperature is sensed with the internal TMx00 thermal sensor, or an external thermistor connected to the THRM pin. The input selection is controlled by bit 7 of *corr_order2*. If that bit is zero, the internal temperature sensor is used, and the thermistor is selected when bit 7 is one.

corr_a0 - corr_a9
These variables are single precision floating point values representing the desired polynomial temperature correction.

temp_volt_sample_interval
This variable defines the sampling interval for temperature and voltage readings, in units of the RTOS interval (10 ms). The minimum value is 2, or 20 ms equivalent.

freq_update_interval
The lower 4 bits define the update interval for tuning (frequency) correction, in units of the RTOS interval (10 ms) with a minimum of 3 (30 ms). The maximum value is 15 (0xF), or 150 ms.

osc_freq
This value is the approximate single precision floating point value of the oscillator. It is used indirectly by the control software to determine divide values and other parameters.

corr_mean
This value is the mean (single precision floating point) of the temperature readings determined during the oscillator testing. The temperature readings are 12-bit unsigned numbers (temperature code) read via the ADC.

Assume five temperature points with the values 2512, 2698, 2801, 2830, and 2950. The mean would be:

$$sample_i = each\ sample$$

$$N = number\ of\ samples$$

$$Mean = \frac{\Sigma(sample_i)}{N} = 2758.2$$

This value and *corr_std* together may improve the precision of the polynomial correction algorithm.

corr_std
This value is the standard deviation (single precision floating point) of the temperature readings determined during the oscillator testing.

The standard deviation is computed using the formula:

$$\sigma = \sqrt{\frac{\Sigma(sample_i - mean)^2}{N}}$$

Using the same values as shown above:

$$\sigma = 144.52$$

temp_sensor_scale
This single precision floating point value indicates the scale factor for converting the temperature sensor reading into a degree C indication. It is not used in any TMx00 correction calculations.

temp_sensor_offset
This single precision floating value indicates the offset for converting the temperature sensor reading into a degree C indication. It is not used in any TMx00 correction calculations.

corr_order2
The bottom 4 bits of this variable define the number of tuning correction updates combined into a rotating average. 0 indicates no averaging (one sample averaging). 16 is the maximum number of averages, represented by 0xF.

Temperature Sensing

The crystal temperature is sensed with the internal TMx00 thermal sensor, or an external thermistor connected to the THRM pin. The input selection is controlled by bit 7 of *corr_order2*. If that bit is zero, the internal temperature sensor is used, and the thermistor is selected when bit 7 is one.

If bit 7 is a 0, the correction algorithm input is the internal IC temperature sensor. If bit 7 is a 1, the correction algorithm input is the THRM pin. The THRM pin has an input range of 0V (0 digital code) to 2.9V (4095 digital code).

lookup_table
The *lookup_table* variable contains up to 80 pairs of temperature code readings for RAM-based table lookup and tuning DAC code values, 3 bytes per pair.

The first 1.5 bytes in each entry contain the Temperature Code, and the last 1.5 bytes contain the tuning DAC value. The most significant bit of the Temperature Code is in the most significant bit of the first byte. The most significant bit of the tuning DAC code starts at the most significant bit of the bottom nibble in the middle byte. The Temperature Code readings must be monotonic from low to high. The tuning DAC values do not have that restriction.

The end of the table occurs when either 1) both the Temperature Code and the tuning DAC values for an entry are zero, or 2) all 80 table entries are filled.

If the Temperature Code value is below the minimum value, the tuning DAC value is set to the DAC value in the minimum code entry. If the Temperature Code is above the maximum value, the tuning DAC value is set to the DAC value in the maximum code entry.

If the temperature sensor reads the exact value of a temperature code, the tuning DAC value is used for the correction. If not, the tuning DAC values are linearly interpolated for temperature sensor readings between two table entries.

corr_b1-b3, corr_c1-c3
These values are single precision floating point parameters that specify the voltage correction as defined in the *corr_order* section.

corr_b_mean
The value is the ADC reading that represents the nominal VDDA supply of 3.3 Volts. The VDDA ADC reading value is ½ of the VDDA value, for 12-bit conversion with a 2.9V reference voltage.

If *corr_b_mean* is 0 (0x0000), the ADC reading measured during TMx00 final test is used.

The standard deviation of the VDDA supply reading is internally computed as (0.05 * *corr_b_mean*)/3, for ± 3 σ at the ± 5% supply values.

heater_startup_delay
TM200 Only

This value is the startup delay for the heater turn-on from power up, measured in units of 2x the RTOS interval. For example, a value of 5 would generate a heater delay of 100 ms assuming a 10 ms RTOS interval.

heater_soft_start
TM200 Only

This value defines soft start for the heater. The current limit of the heater increases from 0 to the value specified by the current limit over the time interval time (*heater_soft_start* * 10 ms), assuming 10 ms RTOS time intervals. If *heater_soft_start* is zero, the current limit of the heater is immediately set to the bits 15 – 9 values of *therm_latch*.

ext_reg_volt
This value is a floating-point of the voltage of an external supply connected to BYPASS when the internal regulator is disabled.

FREQUENCY TRIMMING

The TM100 and TM200 support three correction algorithms as part of a temperature compensation network:

1. **Linear Lookup Table** – Temperature correction algorithm
2. **Temperature Polynomial Curve Fit** – Temperature correction algorithm
3. **Supply Voltage Polynomial Curve Fit** – Voltage correction algorithm

The TMx00 correction algorithms use integer numbers (digital codes) that represent temperatures and voltages to make compensation adjustments. The ADC converts analog signals into the digital domain for the MCU to calculate the appropriate Correction DAC input code and produce an analog correction voltage.

The first two algorithms use the IC internal temperature sensor or an external thermistor (via the THRM pin) to produce a *Temp Code* which then generates a *CorrDAC Code* from the Correction DAC output to correct frequency variations over temperature. Only one of the temperature correction algorithms may be used during operation.

The *Temp Code* value is an integer with the range of 0 to 4095 and corresponds to the temperature being measured and digitized through the ADC. A -40C to 90C temperature range will utilize a *Temp Code* range of approximately 2000 to 3200.

The *CorrDAC Code* value is an integer with the range of 0 to 4095 and corresponds to the DAC input code needed to vary the capacitance across an external varactor or the internal TM100 Varicap to correct the frequency variation for a given temperature.

The *CorrDAC Voltage* is resulting Correction DAC output voltage for a given *CorrDAC Code*. It is calculated by multiplying the Correction DAC's reference voltage by the ratio of the *CorrDAC Code* to the Correction DAC's full-scale code (4095).

$$CorrDAC\ Voltage = \frac{CorrDAC\ Code}{4095} * BYPASS\ Pin\ Voltage$$

As a brief relationship example, a measured temperature of 25C may produce a *Temp Code* of 2541 and results in a *CorrDAC Code* of 1983 and a *CorrDAC Voltage* of 1.4043V for the correct center frequency of a unique TCXO assembly.

Adjustable Timing Parameters

The TMx00 correction mechanism is a hybrid analog/digital implementation and has several adjustable timing parameters that allow customers to tune timing modules to the time constants of the intended applications. The adjustments also implement low pass filters to reduce crystal adjustment transients.

Supported adjustable timing parameters include:

Temperature Sample Interval – This sets the time interval (in ms) for measuring the temperature sensor and VDDA value. The interval is between 20ms and 2550ms. Both the temperature and VDDA readings are averaged over 8 sample intervals, updated at each sample interval (rolling average).

Frequency Correction Interval - This sets the time interval (in ms) for updating the Correction DAC value applied to the Varicap or external varactor. The interval is between 20ms and 2550ms.

Averaging Intervals - This sets the number of Frequency Correction Intervals used to calculate an average DAC correction value, updated each correction interval (rolling average). The averaging is between 1 and 16.

Linear Lookup Table Curve Fit

The Linear Lookup Table correction technique uses the measured reading from the IC temperature sensor or external thermistor (*Temp Code*) and generates a *CorrDAC Code* via a lookup table.

The lookup table is arranged from the lowest *Temp Code* to the highest *Temp Code* with up to 80 defined points. The spacing between the points is user defined to account for changing crystal temperature coefficient slopes. The appropriate DAC correction code (*CorrDac Code*) between the defined *Temp Code* points is calculated by linear interpolation.

Temperature Correction Polynomial Curve Fit

The temperature polynomial curve fit correction technique uses the measured temperature reading from the IC sensor or external thermistor (*TempCode*) and calculates a *CorrDAC Voltage* via polynomial curve fit using the correction order coefficients, a_0 - a_9.

The correction process computes the Correction DAC voltage (0V to 2.9V) needed for best correction. The correction value is converted into a 12-bit DAC input digital code (*CorrDAC Code*), 0 for 0V, and 4095 for the max BYPASS Voltage.

$$Corr_{temp}(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5 + a_6x^6 + a_7x^7 + a_8x^8 + a_9x^9$$

$$x = TempCodeNormalized: TempCode\ normalized\ for\ Mean\ and\ Standard\ Deviation$$

$$Corr_{temp}(x) = Correction\ DAC\ output\ voltage$$

To use the polynomial curve fit algorithm, the following data needs to be selected or entered:

1. Desired correction order: For some applications 3rd or 5th order corrections may produce better results over temperature than a 7th or 9th order correction. This item can be set as needed for each oscillator application.

2. *Temp Code* Mean and *Temp Code* Std Dev: Measure the module over temperature and capture the *Temp Code* for each measured temperature point. Compute the mean and standard deviation of the *Temp Code* values. Then scale the *Temp Code* values by the mean and standard deviation generating *x (TempCodeNormalized)* for the above equation.

3. CorrDAC Codes over Temperature: As part of the module measurements over temperature, determine the *CorrDAC Code* that produces the least error. Convert the *CorrDAC Code* to *CorrDAC Voltage* by scaling it, 0 to 4095 for the voltage range 0 to the BYPASS pin Voltage. *CorrDAC Voltage* is *f(x)* in the equation above.

4. Use polynomial curve fit software to find the values of a_0 to a_n based on x and $Corr_{temp}(x)$.

Enter the coefficients, a_0 - a_9, the *Temp Code* mean, and the *Temp Code* Std Dev for the respective correction order. The coefficients are single precision floating point numbers stored in the microcontroller. Single precision floating point numbers have 7-8 decimal digits of precision and ensure that all significant digits are entered into the microcontroller.

Supply Voltage Correction Trim

The Supply Voltage Correction Trim algorithm is an optional enhanced correction technique that generates an adjustment term $\Delta tempCode_{adj}$ or $\Delta correction$ to compensate for variations over supply voltage (VDDA). The *CorrDACVoltage* values are not overly sensitive to supply variation, but the *TempCodeNormalized* values have a small dependency on supply voltage.

Table 3-3 Voltage Correction

Bits 7-5	Correction or temp code?	Notes
0b000 (0)	–	No voltage/temperature correction
0b001 (1)	$\Delta tempCode_{adj}$	$\Delta temp\ code = \Delta_{VDDA}*corr\_b1$
0b010 (2)	$\Delta\ tempCode_{adj}$	$\Delta temp\ code = \Delta_{VDDA}*corr\_b1+\Delta_{VDDA}^2*corr\_b2 +\Delta_{VDDA}^3*corr\_b3$
0b011 (3)	$\Delta\ tempCode_{adj}$	$\Delta temp\ code = \Delta_{VDDA}*corr\_b1+\Delta_{VDDA}^2*corr\_b2+\Delta_{VDDA}^3*corr\_b3 +\Delta_{VDDA}*tempCode_{norm}*corr\_c1 +\Delta_{VDDA}^2*tempCode_{norm}*corr\_c2 +\Delta_{VDDA}*tempCode_{norm}^2*corr\_c3$
0b100 (4)	–	No voltage/temperature correction
0b101 (5)	$CorrDAC_{ADJ}(\Delta VDDA)$	$CorrDAC_{ADJ}(\Delta VDDA) = \Delta_{VDDA}*corr\_b1$
0b110 (6)	$CorrDAC_{ADJ}(\Delta VDDA)$	$CorrDAC_{ADJ}(\Delta VDDA) = \Delta_{VDDA}*corr\_b1+\Delta_{VDDA}^2*corr\_b2 +\Delta_{VDDA}^3*corr\_b3$
0b111 (7)	$CorrDAC_{ADJ}(\Delta VDDA)$	$CorrDAC_{ADJ}(\Delta VDDA) = \Delta_{VDDA}*corr\_b1+\Delta_{VDDA}^2*corr\_b2+\Delta_{VDDA}^3*corr\_b3 +\Delta_{VDDA}*tempCode_{norm}*corr\_c1 +\Delta_{VDDA}^2*tempCode_{norm}*corr\_c2 +\Delta_{VDDA}*tempCode_{norm}^2*corr\_c3$

When bits 6 – 5 equal zero, voltage correction is not active. When bits 6 – 5 equal 1, the corr_b1 value is used. When bits 6 – 5 equal 2, corr_b1, corr_b2, and corr_b3 are used in the calculations. When bits 6 – 5 equal 3, corr_b1, corr_b2, corr_b3, corr_c1, corr_c2, and corr_c3 are included in the trim calculations.

$$\Delta_{VDDA} = Normalized\ VDDACode_{Meas}\ with\ respect\ to\ VDDACode_{Nom}$$

$$VDDACode_{Nom} = VDDACode\ measured\ by\ IC\ when\ 3.3\ V\ is\ applied\ to\ VDDA$$

$$VDDA_{SD} = Standard\ deviation\ of\ VDDA\ set\ so\ that\ \pm 0.05\ wrt\ VDDACode_{3.3V}\ produces\ a\ \sigma\ of\ \pm 3$$

$$VDDA_{SD} = VDDACode_{NOM}\left(\frac{0.05}{3}\right)$$

$$\Delta_{VDDA} = \frac{VDDACode_{Meas} - VDDACode_{Nom}}{VDDA_{SD}}$$

$$TempCode_{norm} = temperature\ code\ normalized\ using\ corr\_mean\ and\ corr\_std$$

$$Temp_{adj} = temp\ code\ adjustment\ based\ voltage\ \&\ temperature\ measurements$$

$$\Delta TempCode = TempCode_{norm} + Temp_{adj}(\Delta VDDA)$$

$$\Delta CorrDAC\ Code = CorrDAC\ Code + CorrDAC_{ADJ}(VDDA)\ (Lookup\ Table)$$

$$\Delta CorrDAC\ Voltage = CorrDAC\ Voltage + CorrDAC_{ADJ}(VDDA)\ (Polynomial\ Correction)$$

TM200 THERMAL CONTROLLER

The Thermal Controller is used to control the thermal characteristics of the module being constructed. The controller uses a driver to control transistors with resistive loads to dissipate power and cause the cavity temperature to rise. An internal temperature sensor and/or external thermistor provides data input into the feedback network to adjust and stabilize the cavity temperature. It is designed with low Allan deviation (ADEV) in mind.

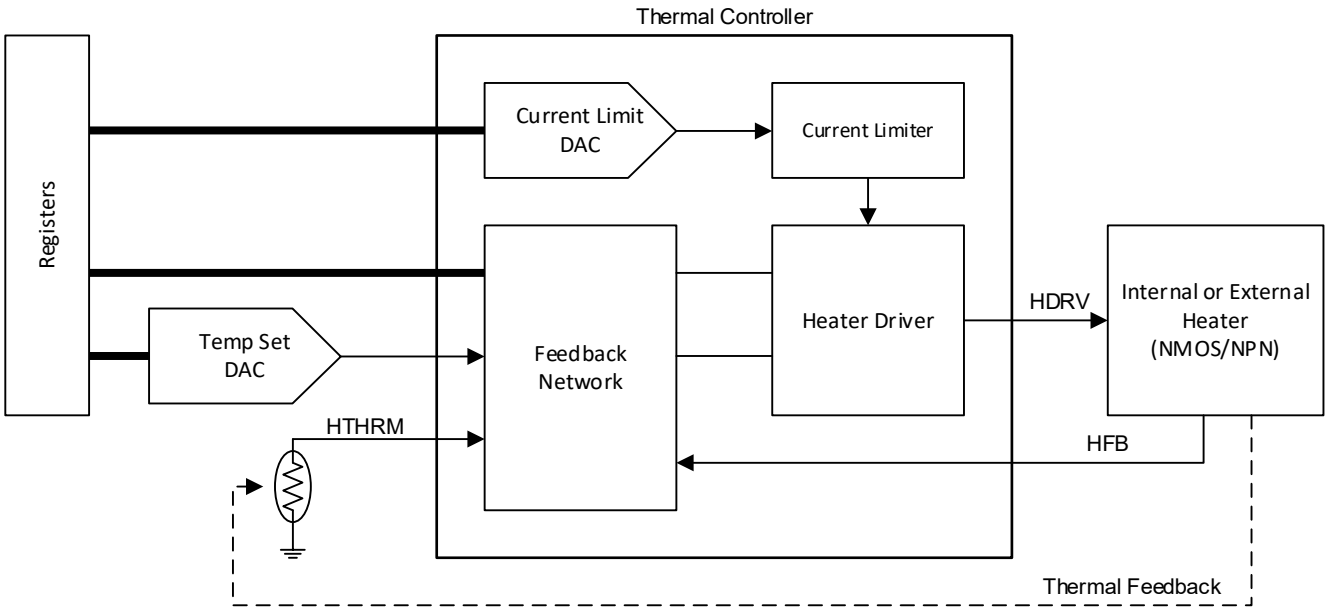

Figure 3-1 TM200 Thermal Controller

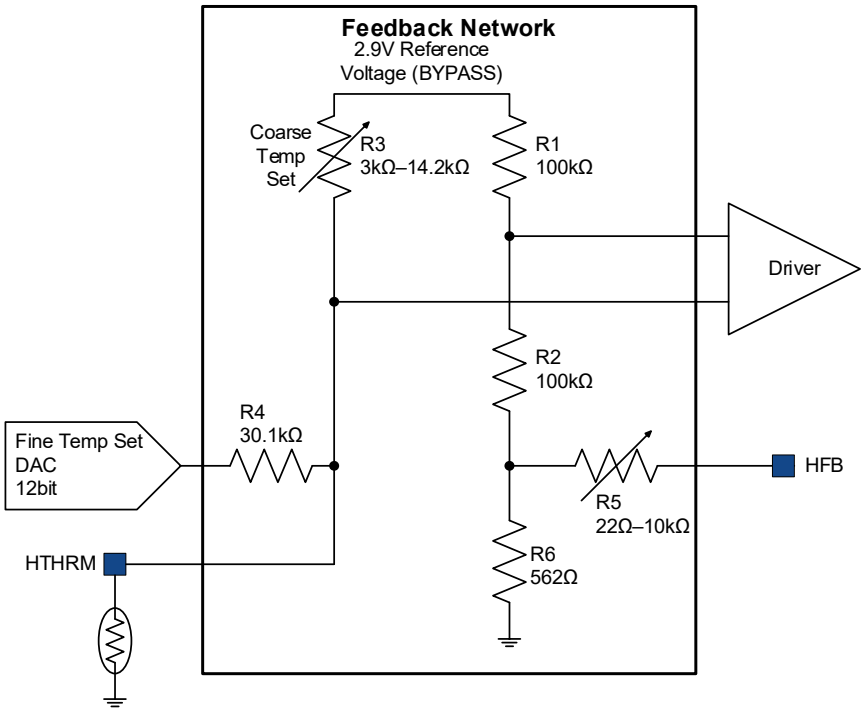

Figure 3-2 TM200 Thermal Controller Feedback Network

Temperature Set Point

Two elements set the controller's settling temperature. R3 is a thermal range set resistor located in the feedback network that selects an approximate temperature range between 70C and 110C based on the resistance of the negative temperature coefficient (NTC) thermistor (100kΩ room temp nominal) curve. The 12-bit Temperature DAC provides trim current up to 50-100µA for the fine temperature setting.

Typical applications use a 100kΩ NTC (Negative Temperature Coefficient) bead thermistor with tight thermal coupling to the crystal. The variation of the resistance over temperature is given by the beta equation or Steinhart-Hart equation.

The beta equation is:

$$R_{T1} = R_{T0} e^{\beta(\frac{1}{T1} - \frac{1}{T0})}$$

$$T0 = Reference\ Temperature\ (25C) in\ °K = 298.15$$

$$T1 = Test\ Temperature\ in\ °K$$

$$R_{T0} = Resistance\ at\ Reference\ Temperature = 100k\Omega\ typical$$

$$\beta = Parameter\ from\ vendor\ that\ illustrates\ the\ value\ shift\ over\ temperature, typically\ \sim 4200$$

Extended Temperature Range

The temperature range of the thermal controller can be moved downward to as low as -20C by replacing the 100kΩ NTC thermistor with one with a lower resistance at the reference temperature. 50k, 25k, or 10kΩ thermistors provide the needed ranges. R3 may need to be replaced with an external resistor for the proper range. The firmware allows disabling R3 for this purpose.

Feedback Network

The feedback network is an input bridge consisting of R1, R2, feedback resistors R5/R6, and the coarse temperature set resistor R3. R1 and R2 shift the feedback voltage up to a level for easy comparison to the thermistor voltage. The feedback resistors R5/R6 monitor the voltage across the heater resistors, thus indicating the wattage provided to the thermal environment. When the two inputs of the Driver block are equal, the bridge is in balance and the controlled temperature is held at that point. The Temperature DAC feeds R4, which in turn provides a fine grain temperature adjustment.

Internal Heaters

Two nominal 0.565W heating elements are used to produce a minimum of 1W of total output power at -40 degrees with a 3.3V supply. The heaters are composed of NMOS FET devices in source follower configurations with a resistive load as the heating element. With an NMOS architecture, the HFB terminal swings between 0V to 0.7V as the HDRV terminal swings between 0V to 3.3V.

HDRV and HFB are controlled and sensed, respectively, by the Thermal Controller and no external connections to those pins are necessary. If more than 1W of power output is desired, external heater elements must be used. An external heater structure is similarly controlled by the Thermal Controller through HDRV and HFB pins. The TM200 does not support using internal and external heaters simultaneously because the IC is not capable of stabilizing two thermal loops.

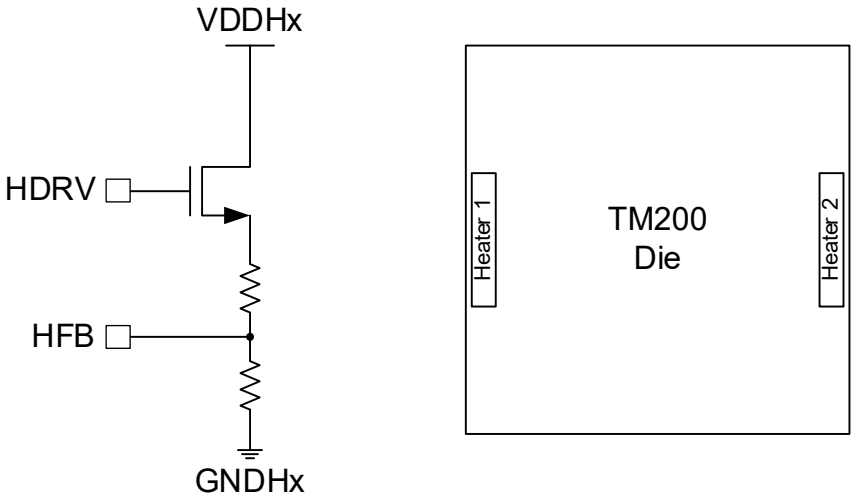

Figure 3-3 Internal Heater

External Heaters

Either FETs or bipolar transistors can be used for external heaters. FETs require very little gate drive from the HDRV pin, while bipolar power transistors need a base drive of 1/(transistor β). The maximum available drive is settable in the firmware to either 500 µa maximum or 7mA maximum. The 7mA maximum output supports up to 4 high beta bipolar power transistors that can drive thermal enclosures approximately 50 x 50 mm.

Loop Dynamics

The loop is stable with excellent phase margin based on the internal compensation network. The value of resistor R5 is selected for optimum thermal settling time without excessive overshoot. The internal compensation assumes current feedback in addition to thermal feedback. No external components are needed, except for a possible snubber network on the HDRV in some external drive scenarios.

Controlling the internal heaters in conjunction with external heaters is not supported as the loop stability dynamics can be significantly different. This is especially the case between NMOS and NPN transistors.

Current Limiter

The Current Limiter prevents the heaters from increasing beyond a pre-set value. A 7-bit DAC sets the maximum current the heater will deliver. When the heater power is applied and the cavity temperature is low, the heater current and the resulting HFB voltage increases until a comparator trips. Once tripped, the driver cannot deliver any more power to the heater and thus limits the temperature rate of rise in the cavity. The trip point of the comparator is set with the Current Limit DAC. The appropriate voltage trip value is dependent on the heater network and the full-scale voltage of HFB.

The TM200 firmware supports a delay time from power on to the time at which the heater power is enabled. It also supports soft start that ramps the current limit from 0 to the desired limit value over a defined time.

4 References

TM100 Datasheet

TM200 Datasheet

TMx00 Control Software & EVB Kit Guide