

FLASH UPDATE PROTOCOL**Determining Which ISP (In-System Programming) Version is Used by Your Device**

Older devices are all programmed with the GoalTender ISP.

As of January 2005 all TEMCO devices are using the TEMCO ISP software

The following is a simple test to determine which ISP is in use:

- 1.) Connect the ISP jumper (of Flash Update jumper) on your device and power it up.
- 2.) Connect the device to the computer using the RS232-485 adapter and serial cable.
- 3.) Open Modbus Poll and connect at 19200 bytes, reading device address 255.
- 4.) If you are able to successfully read values from the device, then the device has the Temco ISP. Otherwise it is programmed with the GoalTender ISP

Note: in the event the module is programmed with GoalTender ISP, please communicate with one of our engineer to help you upgrade to TEMCO ISP!

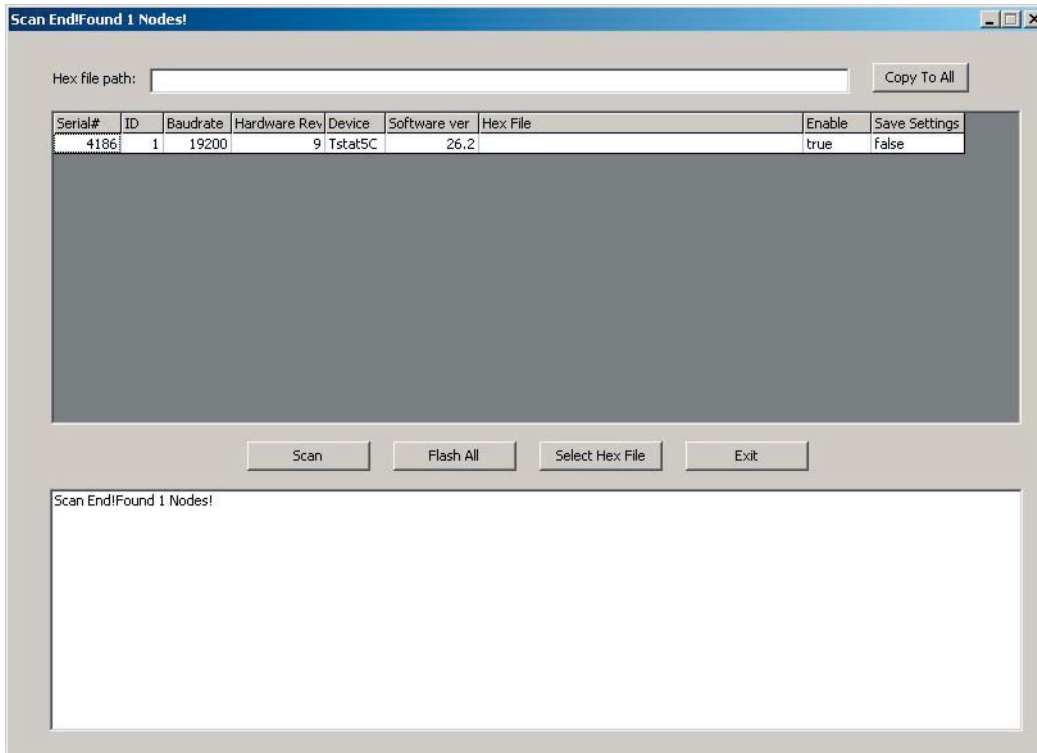
Instructions for Updating Devices with Temco ISP

For TEMCO devices that utilize the Temco ISP, the flash update must be done using the provided T3000 software. To perform a firmware update, follow these instructions:

- 1.) Download and install the T3000 software: <http://www.temcocontrols.com/ftp/software/9TstatSoftware.zip>
- 2.) Connect the device to the serial port of your computer using the RS232-485 converter included in the package.
- 3.) Power up the device.
- 4.) Open the NWT3000 software and select Update Firmware from the Tool menu:



5.) The software will now open the Update Firmware window and will scan for available devices.



6.) For each device that is found, you can specify the hex file to be used for the update. Do this by clicking in the Hex File column of the row you wish to specify. Alternatively you can click Select Hex File and then Copy to All if all devices are to receive the same file. You can also choose to save the current settings or to load the default settings by selecting True or False from the Save Settings column.

7.) At this point simply click Flash All and the software will update each device one by one.

Protocol for Developers Wanting to Update Devices with Temco ISP

All devices programmed with Temco ISP are capable of being updated over the RS485 network. The master on the network sends a command to a particular device, which forces it to go into a 'flash update mode'. The device first resets itself and then jumps to the 'In System Programming' (ISP) code section. Note that all non-volatile parameters should be read and saved prior to this for safe keeping.

NOTE: Multiple-Write Command of the Modbus protocol is used.

Protocol

In order for the front end to communicate with the ISP flash, a series of registers have been defined, which are used as control registers for the Update functions. Reading and writing to these registers will allow the Front end to monitor the status of the update process. They are stored in the non-volatile memory space to keep track of the steps attempted and completed. Below is a description of these control status registers.

Register	Register address	Description
EEPROM_VERSION_NUMBER	4	Software Version
EEP_ADDRESS	6	ID number of the device
EEP_UPDATE_STATUS	16	Update Register state

Table 1. Flash Update Function Registers

It is important to note 'EEP_UPDATE_STATUS' which is located at register address 16. Writing to this register will cause the device to either reset itself, erase its flash or start programming depending on the action being taken. Below is a description of the values and explanation of the EEPROM_UPDATE_STATUS register.

Function	Value	Description of EEP_Update_Status
Update initialize	7Fh	Tell the Tstat to reset and jump into the ISP to be in update mode
Update ready		Tstat is in the ISP and ready to update
Erase flash	3Fh	Tell the Tstat to erase Flash Memory
Erase done		Erase Flash Memory done
Start Programming	1Fh	Start Programming - In upload state
Normal State	01h	Update is complete, tstat reboots with new flash image

Table 2. EEP_UPDATE_STATUS register value description

- For the device to jump into update mode, a write command of value 7Fh must be sent to the EEP_UPDATE_STATUS. The device will then reset itself and run in ISP mode. Note: the device will not send any response in this step. To verify the T3module is in ISP mode, the same write command must be sent again (write 7Fh to register #16), at which point the T3module will respond with a regular modbus response. This is necessary for clearing the Interrupt vectors and making sure all RAM memory is cleared.

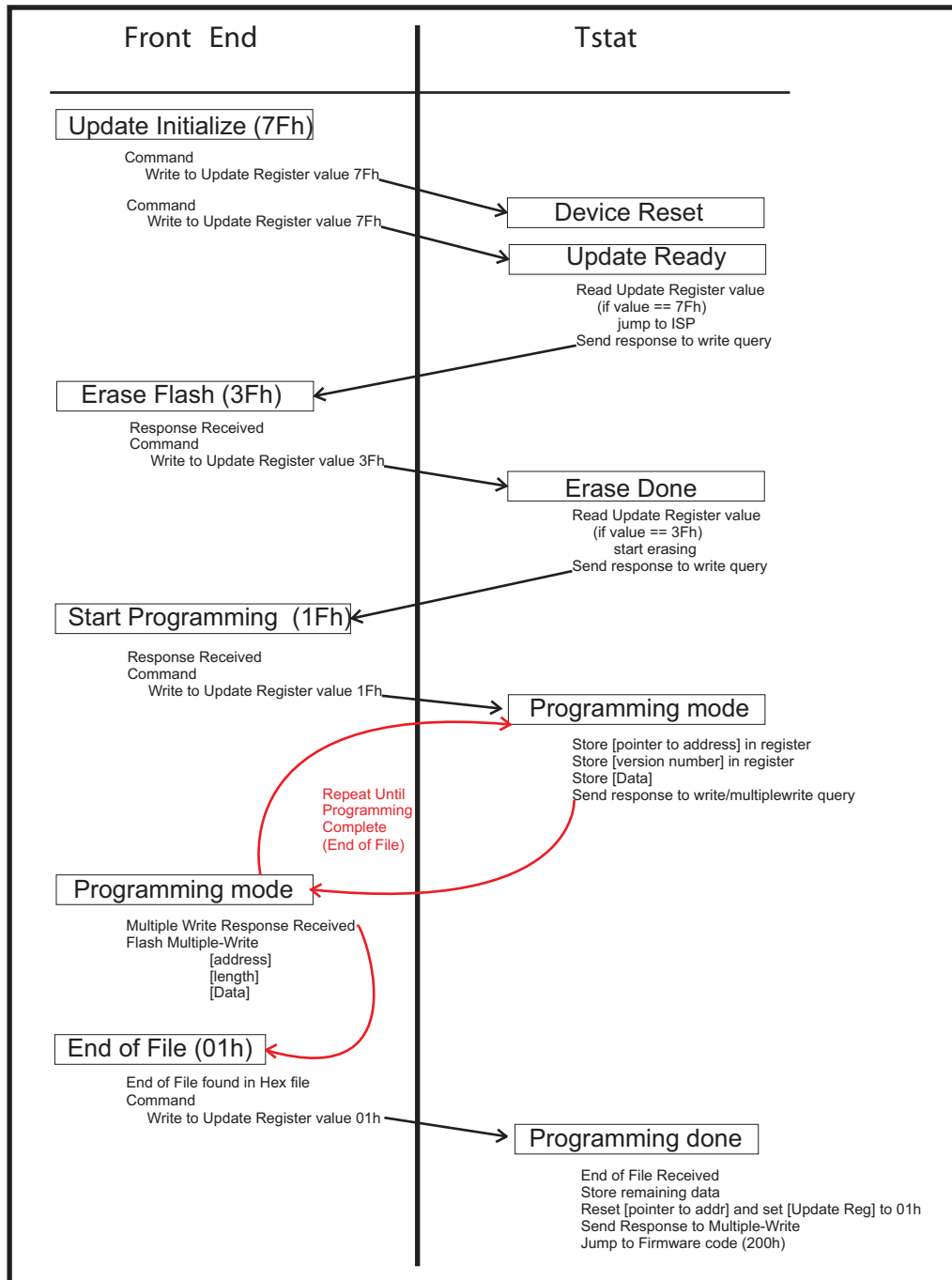
[Notice]: All devices are installed with a 'Flash Update' jumper. Linking the jumper upon resetting the device will force the firmware to start in ISP mode and then wait for further instructions. Note that at this stage the device may not have a proper ID and by default will be set to 254.

- All Modbus communication commands are always followed by a response. This Flash Update Protocol makes use of that criteria and thus only sends a response once the action has been completed. Therefore the 'update initialize' and 'erase flash' step require a longer timeout period than the 'programming' step. (250ms and 500ms respectively)
- Sending a write command of value 3Fh to EEP_UPDATE_STATUS will force the device to erase its entire flash memory. Once the response is received, the device is ready to download the data of the new firmware.
- Sending a write command of value 1Fh to EEP_UPDATE_STATUS will let the device know it is about to receive new firmware. The device is now ready to accept the new hex file and will maintain a running tally of the current programming location in the EEP_UPDATE_PTR.
- At this point, the data must be sent using the multiple-write command. Packets can be of size 1 data byte to a **maximum of 128 data bytes**.
- In the event of an interrupted flash update, the master can poll the EEP_UPDATE_PTR and begin programming from this location.

Below is a graphical representation of the protocol.

Example of a Programming Routine

The ISP has been designed using polling vectors rather than Interrupt vectors in order to free up as many interrupts for the program itself. Given that polling is now used, communications is more susceptible to timing and response delay problems. Therefore, when sending a write function or multiple-write function to the ISP device, a short timeout delay is required before receiving a response ($\approx 20\text{ms}$). If a response was not received during that period of time the FRONT END would need to resend the data once again. Below is a diagram representation of the Flash-Update Protocol.



Example of a Programming Routine (Front End Side)**UPDATE INITIALISE**

- 1 - Send Modbus **Write** Command to address **Update_Register** value **7Fh**
The device will reset itself. Make sure all volatile information be saved prior to this step
Device will not send a respond
- 2 - Send Modbus **Write** Command to address **Update_Register** value **7Fh** again
A response will be received if the Device has properly reset itself and booted under ISP mode

ERASE FLASH

- 3 - Send Modbus **Write** Command to address **Update_Register** value **3Fh**
A response will be received once the Device has properly Erase all Flash Memory
This will step require a longer response timeout period (approx 500ms)

START PROGRAMMING

- 4 - Send Modbus **Write** Command to address **Update_Register** value **1Fh**
A response will be received once the Device has properly set itself for programming mode

PROGRAMMING MODE

- 5 - Extracting data from Intel Hex file. A typical line would look like the following:
:10 0080 00 AF5F67F0 602703E0 322CFA92 007780C3 FD
- 6 - Verify checksum
 $10 + 00 + 08 + 00 + AF + \dots + C3 + FD = 900$
If two last digits of the sum is zero, Hex file is correct
- 7 - Send data using Modbus **Multiple-Write** Command
Address **0080h**
Data length of **10h**
Data **AF5F67F0 602703E0 322CFA92 007780C3**
- 8 - Repeat step 5 through 7 until end of Hex file is reached
IMPORTANT NOTE to ensure proper reset of the device, the value at address register 0000h of the Goal chip must remain as FF.
Most (but not all) of Temco's Hex file will contain this line:
:03 0000 00 020200 F9
Data written to the Goal Flash register **MUST** be modified from **020200** to **FF0200**

END OF FILE

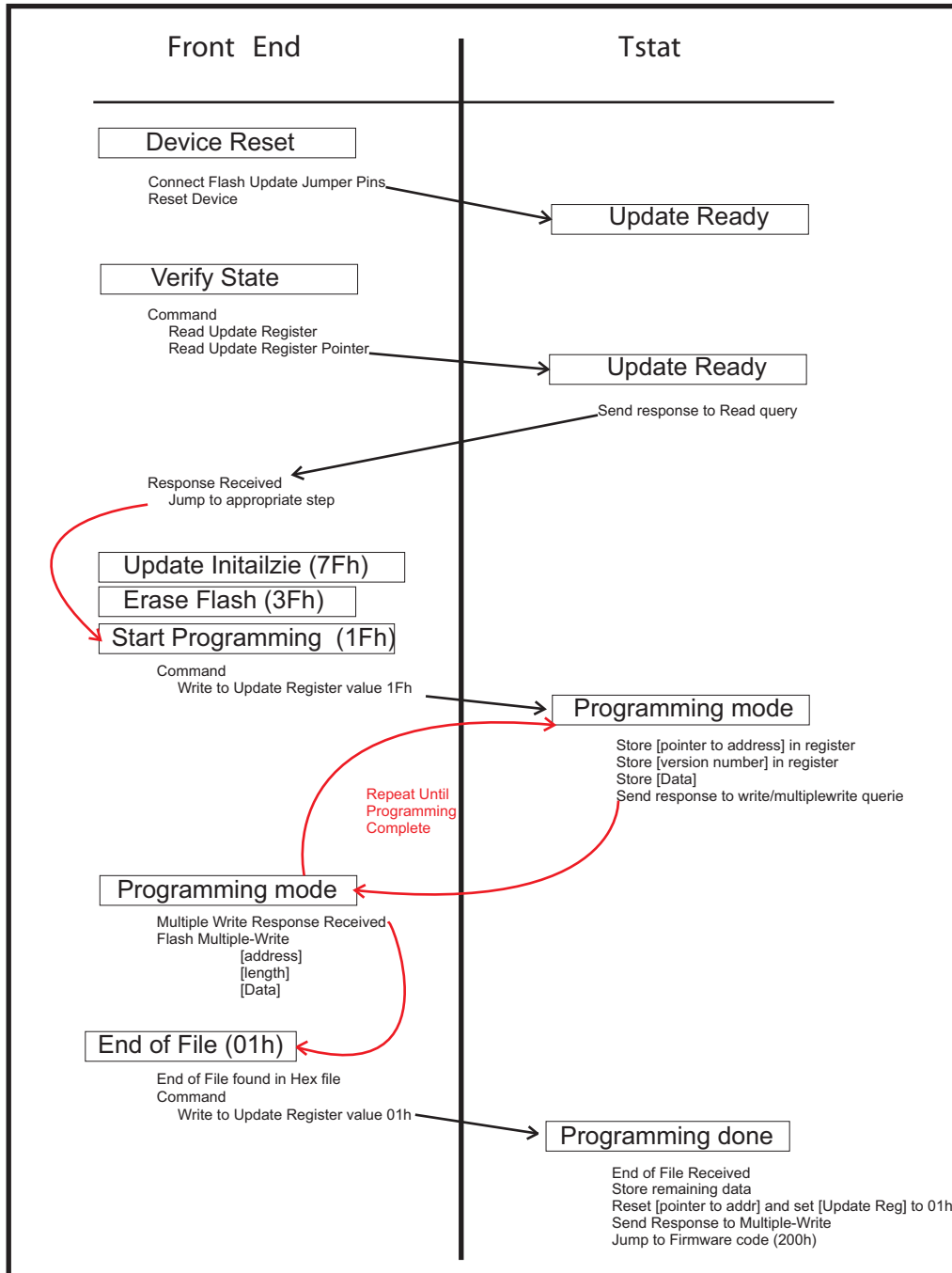
- 9 - End of file found in Hex file
:00 0000 **01** FF
Bit 7 and 8 are 01
- 10 - Send Modbus **Write** Command to address **Update_Register** value **01h**
This will cause the device to reset itself and boot in normal operation mode

To Resume a Previously Interrupted Programming Routine

If during the programming sequences the upload was interrupted, there is still possibility for the front end to resume its programming routine. The EEPROM_UPDATE_STATUS register keeps track of which step is being performed during the update protocol and the EEPROM_UPDATE_PTR keeps track of which register is currently being written to.

- If the device was in the Erase Flash mode, the EEPROM_UPDATE_STATUS register will read 3Fh. The Front End is then required to repeat this step and follow up from there.
- If the device was in the Programming mode, the EEPROM_UPDATE_STATUS register will read 1Fh. The Front End then needs to read the EEPROM_UPDATE_PTR register in order to know where the last upload was being performed. Note that this register is written to before the data has been uploaded. **Thus, in order to resume this step the Front End needs to re-write to this register again** and then follow up from there.

Assuming the new software has only been partially uploaded, the Flash Update Jumper pins need to be connected upon resetting of the board. The following diagram represents the update resume procedure.



IMPORTANT:

In order for the device to jump into the ISP mode, it has to reset itself. Upon reset, if the value at address register 0000h is FF the device will jump to the ISP code section. This is a hardware criteria of the Goal Chip and an efficient way to jump to In System Programming mode while clearing all buffers. **The front end must ensure that only value FF is to be written to address register 0000h.** When reading the hex file, there will be a line such as this:

Data of the new Firmware

;03 0000 00 020200 F9

(Intel Hex format described below):



need to change to this to

Modified data to be uploaded

;03 0000 00 FF0200 FC

Intel Hex File

All firmware files produced by our compilers are saved under the Intel Hex file format. This format of record can be broken down in its different fields as described below.

Example of an Intel Hex file

Take for instance a typical message such as the following:

```
:ll aaaa tt D1D2D3D4 D5D6D7D8 D9D0D1D2 D3D4D5D6 ee  
:10 0080 00 AF5F67F0 602703E0 322CFA92 007780C3 61
```

- The first character (:) indicates the start of a record.
- The next two characters indicate the record length (10h).
- The next four characters give the load address (0080h).
- The next two characters indicate the record type. (00)
- Then we have our data
- The last two characters are a checksum (sum of all bytes + checksum = 00).

Record types:

- 00 - Data record
- 01 - End of file record
- 02 - Extended segment address record
- 03 - Start segment address record
- 04 - Extended linear address record
- 05 - Start linear address record

Flash Update Jumper

In the case where the device is locked, there is still a possibility to reboot the device and upload a new firmware. This requires to physically link the jumpers of the Flash Update Jumper pins during restart:

- Power down the device
- Link the jumpers of the Flash Update Jumpers
- Power up the device

Doing the above steps will force the device to be in ISP mode so that new firmware can be loaded. In order to return to normal operation once the upload has been done the Jumper needs to be removed and power need to be recycled.

Frequently Asked Questions

Revision History

Firmware Version 1.0:

- First Prototype version of Temco ISP
- Protocol explanation on page 26:

Update Initialize	20h
Erase Flash	30h
Start Programming	40h
Normal State	FFh

Firmware Version 2.0:

- Major change in the ISP protocol required due to hardware specification
- Protocol explanation on page 26:

Update Initialize	7Fh
Erase Flash	3Fh
Start Programming	1Fh
Normal State	01h

Firmware Version 2.1:

- Added change in LED display to know the device is functioning normally in ISP mode
- Added control of Driver Chip to Relays to ensure they remain off during Upload

Firmware Version 3.0:

- Added networking feature such as SCAN and PLUG-AND-PLAY
- Reading of Modbus Registers through Modbus Communication now returns information such as Serial Number, Software version, Product address, Product Model and Product Hardware Revision

Firmware Version 3.1:

- Stores the ISP Firmware Version Number in the chip Flash Registers

Firmware Version 3.2:

- New SCAN and PLUG-AND-PLAY featured added
- Added safety conditions such that device cannot restart with a partial code
- Made serial communication timeout flexible