TINSAT Features

- Pressure, temperature and humidity sensors
- Total of six A/D or four A/D and I^2C
- ATmega328P microcontroller at 12.288 MHz
- Serial interface
- LED output
- 9 V input
- 1200 bit/s AFSK or 9600 bit/s FSK or GFSK AX.25 packets
- 916.36 MHz ISM Band transmit frequency (other frequencies available on request)
- Up to 20 mW transmit power
- Very small size 38.6x51.3 mm single board or separate boards for sensors, controller and transmitter
- Bootloader for Flash and EEPROM programming using standard serial cable

Introduction

The TINSAT system consists of three separate modules. The Sensor Module (SM) includes a Freescale Semiconductor MPX4115A pressure sensor, Microchip TC1047A temperature sensor and Honeywell HIH–4000 humidity sensor. The Controller Module (CM) includes an Atmel ATmega328P 12.288 MHz microcontroller with serial interface and a light emitting diode (LED). The Transmitter Module (TM) includes an Analog Devices ADF7012 transmitter. The modules are available separately, as one single printed circuit board (PCB), as a SM/CM PCB or as a CM/TM PCB. Figure 1 is a photo of the single PCB configuration.

The heart of TINSAT is the CM. It typically accepts power from a 9V battery and provides analogue to digital conversion of the analogue outputs from the SM. The CM configures the TM and transmits data from the CM to the TM. Data is transmitted in AX.25 packets, either using 1200 bit/s AFSK or 9600 bit/s FSK or GFSK.

Sensor Module (SM)

The sensor specifications are listed in Table 1. Please refer to the appropriate data sheets for more detailed information. For the humidity sensor, a calibration sheet is included to allow for greater accuracy. The humidity sensor can also be compensated for temperature changes. See the HIH–4000 data sheet for further information.

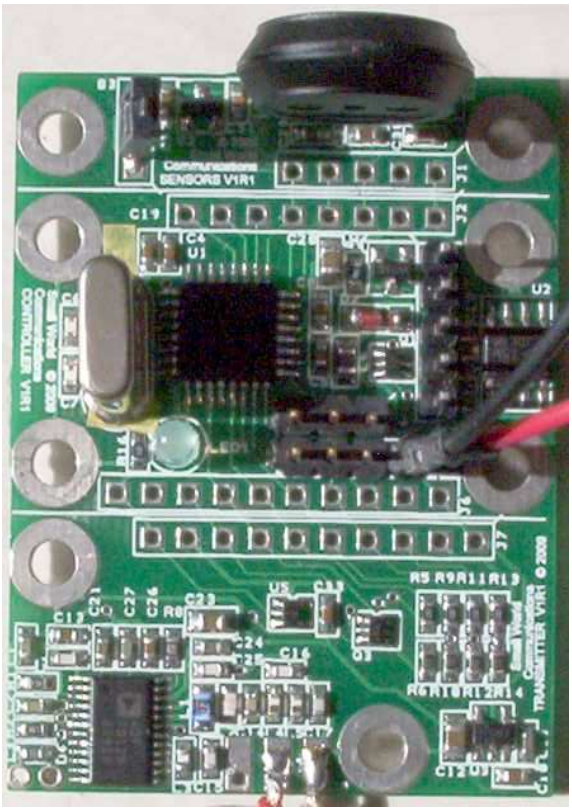

Figure 1: Single board configuation (larger than actual size).

Table 1: Sensor specifications

Sensor	ADC	Min Range	Min (V)	Max Range	Max (V)
Pressure	0	15 kPa	0.204	115 kPa	4.794
Temperature	1	–40 C	0.100	125 C	1.750
Humidity at 25 C	2	0 %	0.800	100 %	3.900

Table 2: Sensor Connector J1

Pin	Signal	I/O
1	VCC	I
2	GND	I/O
3	PRES	O
4	TEMP	O
5	HUMI	O

There is a single 5–pin inline connector on the SM with the connections given in Table 2. Pin 1 is to the right of the PCB.

Table 3: ATmega328P Connections

Pin Name	Pin No.	Signal	I/O
PB0/PCINT0/**CLKO**/ICP1	12	CLKO	O
PB1/PCINT1/**OC1A**	13	TXDATA	O
PB2/PCINT2/OC1B/**SS**_	14	SS_	O
PB3/PCINT3/OC2A/**MOSI**	15	MOSI	O
PB4/PCINT4/**MISO**	16	MISO	I
PB5/PCINT5/**SCK**	17	SCK	O
PB6/PCINT6/**XTAL1**/TOSC1	7	XTAL1	I
PB7/PCINT7/**XTAL2**/TOSC2	8	XTAL2	O
PC0/PCINT8/**ADC0**	23	PRES	I
PC1/PCINT9/**ADC1**	24	TEMP	I
PC2/PCINT10/**ADC2**	25	HUMI	I
PC3/PCINT11/ADC3	26	PC3	I
PC4/PCINT12/ADC4/SDA	27	PC4	I
PC5/PCINT13/ADC5/SCL	28	PC5	I
PC6/PCINT14/**RESET**_	29	RST_	I*
PD0/PCINT16/**RXD**	30	RXD	I*
PD1/PCINT17/**TXD**	31	TXD	O
PD2/PCINT18/INT0	32	NC	I
PD3/PCINT19/**INT1**/OC2B	1	TXCLK_	I*
PD4/PCINT20/T0/XCK	2	NC	I
PD5/PCINT21/T1/OC0B	9	LED	O
PD6/PCINT22/AIN0/OC0A	10	NC	I
PD7/PCINT23/AIN1	11	MUXOUT_	I*
ADC6	19	NC	I
ADC7	22	NC	I

* Requires internal pullup resistor

Controller Module (CM)

The controller module uses an ATmega328P microcontroller with a 12.288 MHz crystal. A 78L05 linear voltage regulator is used to accept input voltages from 7 to 20 V to produce a nominal 5 V supply. This typically allows a 9 V battery to supply power to the CM. The ATmega328P provides up to 32kB of Flash memory, 2kB of RAM and 1kB of EEPROM. The microcontroller also has many other features of which the following are nominally used:

* 8–bit Timer/Counter0 (for internal clock)
* 16–bit Timer/Counter1 (for 1200 bit/s AFSK and 9600 bit/s FSK)
* INT1 Interrupt (for 9600 bit/s GFSK)
* Master/Slave SPI (slave for programming ATmega328P and master for programming transmitter)
* Serial USART (for computer interface)
* 6–channel 10–bit ADC (two of the available eight channels are not used)
* Crystal Oscillator XTAL1/XTAL2
* Clock output CLKO
* External reset RESET_
* Bootloader for Flash and EEPROM programming. This uses 2KB of the available 32KB of Flash memory.

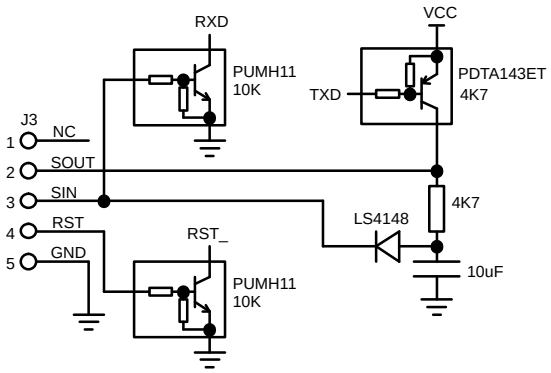

Figure 2: Serial Interface

The CM is used to interface to the SM and TM. A serial interface and LED output are also provided. The connections to the ATmega328P are listed in Table 3. For the pin name, the signal in bold indicates which of the pin options are nominally being used. Note that RESET_, RXD, TXCLK_ and MUXOUT_ inputs require the corresponding pullup resistors to be enabled within the ATmega328P. An underscore after a signal name indicates either an active low or inverted signal.

Figure 2 shows how the 5–pin serial interface is connected to the ATmega328P. The interface allows RS–232 level signals to be used. J3 can be connected to a 9–pin serial cable as indicated in Table 4. Pin 1 is to the top of the PCB, near the SM.

Table 4: 9–pin Serial Cable to Serial Interface

DB–9 Name	DB–9 Pin	Serial Name	Serial Pin
DCD	1	NC	1
RXD	2	SOUT	2
TXD	3	SIN	3
DTR	4	RST	4
GND	5	GND	5
DSR	6	–	–
RTS	7	–	–
CTS	8	–	–
RI	9	–	–

When using the serial cable, RST (DTR) must be kept low, otherwise the ATmega328P will reset. Alternatively, DTR can be disconnected from the connector. RST (DTR) is used with a bootloader program in the ATmega328P in order to load programs via a serial cable.

A high efficiency green LED is available on the CM. The circuit diagram is shown in Figure 3. The LED uses 2 mA of current for a luminous intensity of 2.3 mcd.

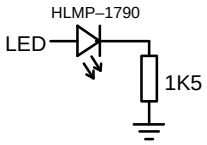

Figure 3: LED Circuit

In order to program the ATmega328P, a standard 6–pin (2x3 configuration) SPI connector is available using J5. Figure 4 shows the pin connections with pin 1 is to the right of the PCB. This allows the Atmel AVRISPII in–system programmer to be used.

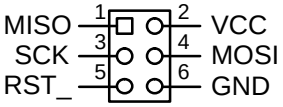

Figure 4: SPI Programming Connector J5

Table 5 shows the connections for the 2–pin battery connector J4, 8–pin connector to the ADC inputs J2 (only pins one to five are used by the SM), and 10–pin connector for the TM J6. Pin 1 is to the right of the PCB for J4, J2 and J6.

Table 5: Battery, SM and TM connectors in CM

Pin	J4 (battery)	J2 (SM)	J6 (TM)
1	VIN	VCC	VCC
2	GND	GND	GND
3	–	PRES	SCK
4	–	TEMP	MOSI
5	–	HUMI	SS_
6	–	PC3	TXDATA
7	–	PC4	MUXOUT_
8	–	PC5	TXCLK_
9	–	–	GND
10	–	–	CLKO

To program the TM, the ATmega328P operates in SPI Master mode, using signals SCK, MOSI and SS_. The transmitter is programmed to have a centre frequency of 916.36 MHz, frequency shift keying (FSK) or Gaussian FSK (GFSK) modulation, ±3 kHz frequency deviation and a transmit power of 3 mW (the legal limit in Australia for non frequency spread signals in the ISM band). A 12.288 MHz clock is provided to the TM using CLKO.

For 1200 bit/s AFSK (audio FSK), either 1200 Hz or 2200 Hz signals are sent via TXDATA. For 9600 bit/s FSK the data is sent directly to TXDATA. A 0 or 1 on TXDATA will modulated the transmitted signal with a frequency of 916.357 MHz or 916.363 MHz, respectively.

The transmitter can also be programmed for GFSK, OOK (on off keying), GOOK (Gaussian OOK) and ASK (amplitude shift keying). GFSK and GOOK use less bandwidth than FSK and OOK. For GFSK and GOOK, the TM generates a clock on TXCLK_. On the rising edge of TXCLK_, the next transmitted bit is sent via TXDATA. Note that only certain data rates can be used. Other transmit frequencies can also be programmed.

The MUXOUT_ signal indicates the status of the ADF7012 chip in the TM. On powerup, this signal is high, indicating that the ADF7012 is not yet ready to be programmed. When low, the ADF7012 is ready to be programmed.

Transmitter Module (TM)

The TM uses an Analog Devices ADF7012 ISM transmitter. The device parameters are programmed using an SPI interface with the ADF7012 acting as a slave. There are four 32–bit registers that need to be programmed.

As the ADF7012 requires a 3.3 V power supply, a low dropout voltage regulator (Texas Instruments TPS76433) is used to reduce the 5 V input supply voltage to 3.3 V.

In order to meet the required CMOS voltage levels between the 5 V level signals from the CM to the 3.3 V signals in the TM, adaptor circuits are required. Figure 5 illustrates the adaptor circuits used. The TM has a 10–pin SIL connector J7, which is the same as J6 in the CM (see Table 5). Pin 1 is to the right of the PCB.

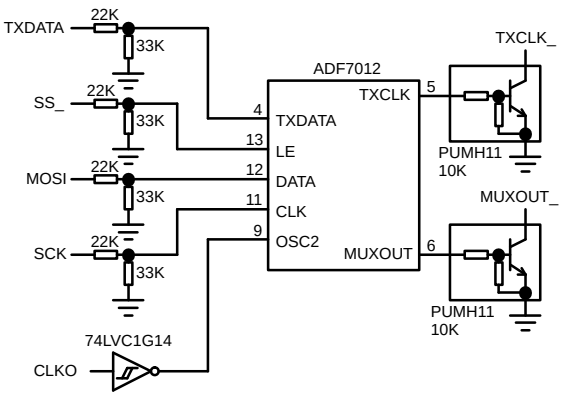

Figure 5: TM Adaptor Circuits

The loop filter in the transmitter was designed for a centre frequency of 916.36 MHz, 9600 bit/s GFSK, ±3 kHz frequency deviation and 20 kHz bandwidth. The circuit and values used in the loop filter are shown in Figure 6. Other components used in the VCO are also shown.

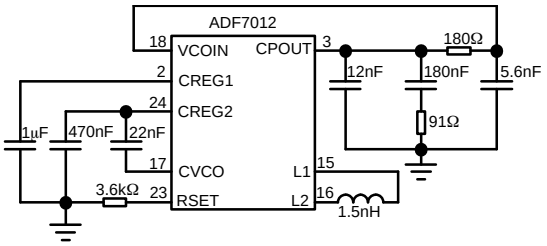

Figure 6: TM Loop Filter and VCO Components

In order to filter out the second and higher order harmonics at the transmit frequency, a fifth order low pass Chebyshev filter is used. The circuit used is shown in Figure 7. A 50 Ω matching filter prior to the transmit filter is also shown. An optional SMA connector J8 can be soldered to the TM.

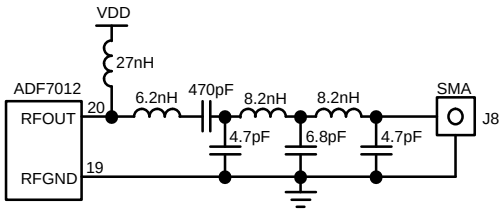

Figure 7: TM Matching and Transmit Filter

The TM can be programmed for other data rates and transmit frequencies, however the components shown in Figures 6 and 7 may need to be changed. Transmit frequencies range from 75 to 1000 MHz with data rates from 0 to 179.2 kbit/s. Transmit power can vary from –16 dBm (25 µW) to 13 dBm (20 mW).

Mounting

The mounting holes for the SM, CM and TM are shown in Figure 8. Alternatively, a single PCB can be mounted with a board size of 38.6 by 51.3 mm.

Electrical Specifications

Sensor Module
Supply current: 10.6 mA (max)
Supply voltage: 4.85 V (min), 5.35 V (max)

Controller Module
Supply current: 32.4 mA (max)
Supply voltage (VIN): 7 V (min), 20 V (max)
Output voltage (VCC): 4.75 V (min), 5.25 V (max)

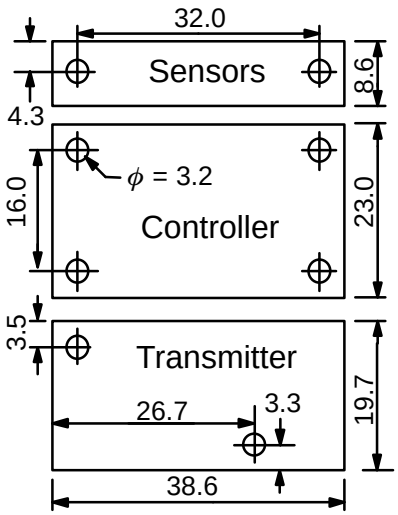

Figure 8: Mounting Holes

Frequency (CLKO): 12.288 MHz

Transmitter Module
Supply current: 45.9 mA (max), 23.2 mA for 3 mA VCO current and 3mW transmit power
Supply voltage (VCC): 3.6 V (min), 10 V (max)

Software

Included with TINSAT is a Pascal unit tinsat.mcl written for use with mikroPascal Pro for AVR [1]. The basic program structure is:

```
program tinsat_test;
uses tinsat;
begin
  {Main program goes here}
end.
```

The following procedures and functions are available:

`init_controller;`
This procedure initialises the I/O port pins and enables the A/D converter. It should be used only once at the beginning of a program.

`init_clock;`
This procedure initialises the internal clock to use Timer/Counter0. The clock period is specified by the variable `tic_period` in ms (default value is 1000 or 1 s). Every `tic_period` ms, the Boolean variable `tic` becomes `true`. If `tic_period` = 0, then `tic` is always `false`. The maximum `tic_period` is 65535 (65.535 s). For example, to make the `led` toggle every 1 s (the constant

end_of_time is equal to false so the loop never ends).

```
init_controller;
init_clock;
repeat
  if tic then
    begin{toggle LED}
      tic := false;
      led := led xor 1;
    end;{toggle LED}
until end_of_time;
```

If the clock is not being used, the following directive in tinsat.mpas should be undefined to reduce code complexity. This can be achieved by deleting the $ sign before DEFINE.

```
{$DEFINE do_clock}
```

`wait_for_tic;`
This procedure waits until `tic` becomes true and then sets `tic` to be `false`. The above toggle LED program can be written as

```
init_controller;
init_clock;
repeat
  wait_for_tic;
  led := 1;
  wait_for_tic;
  led := 0;
until end_of_time;
```

`sample(sensor:byte):integer;`
This function performs a 12–bit analogue to digital conversion on one of the six ADC inputs. The output is an `integer`. The input `sensor` is a `byte` (0 to 5 selects inputs ADC0 to ADC5, respectively) with the following available constants:

`pressure` = 0 = pressure sensor (ADC0)
`temperature` = 1 = temperature sensor (ADC1)
`humidity` = 2 = humidity sensor (ADC2)
`ground` = 14 = ground reference (0 V)
`bandgap` = 15 = bandgap voltage (1.1 V)

```
int2str(data:integer;
          width:short):string[6];
```

This function converts an `integer` input `data` into a `string` of length equal to the maximum of `width` and the number of characters in the integer. The maximum length is equal to six. The maximum value of `data` is 32767 and the minimum value is –32768. For example

```
data_out := int2str(42,4);
    {outputs “  42”}
data_out := int2str(42,0);
   {outputs “42”}
data_out := int2str(42,7);
   {outputs “    42”}
```

`char2str(character:char):string[1];`
This function converts `char` input `character` into a `string` of length one.

`init_tx(rate:byte);`
This procedure initialises the ADF7012 transmitter. For `rate` = 0 1200 bit/s AFSK is selected. Timer/Counter1 is set to operate in phase and frequency correct pulse width modulation. For `rate` = 1 9600 bit/s FSK is selected. Timer/Counter1 is set to operate in clear timer on compare match mode. For `rate` = 2 9600 bit/s GFSK is selected. The TXCLK_ input from the ADF7012 input is used to trigger an interrupt on the rising edge. The ADF7012 is programmed to have a centre frequency of 916.36 MHz, FSK modulation, a frequency deviation of ±3 kHz and a transmit power of 3 mW. The following constants have been defined for `tx_rate`

`afsk1200` = 0 = 1200 bit/s AFSK
`fsk9600` = 1 = 9600 bit/s FSK
`gfsk9600` = 2 = 9600 bit/s GFSK

To reduce code complexity, the following program directives can be used to enable one or more modulation options in tinsat.mpas.

```
{$DEFINE do_afsk1200}
{$DEFINE do_fsk9600}
{$DEFINE do_gfsk9600}
```

The default setting in tinsat.mcl only has `do_afsk1200` defined, that is 9600 bit/s FSK or GFSK are not implemented.

`send_tx;`
This procedure sends the length 230 `string` variable `data_out` to the transmitter in an AX.25 packet. The length six string variables `source`, `destination`, and `via` are used to indicate the source, destination and via (relay) in the AX.25 packet, respectively. These variables are initialised with the following values by `init_tx`:

```
source := 'TINSAT';
destination := 'CQ';
```

```
via := 'TELEM';
data_out := '';
```

For example, to send a pressure sample at 1200 bit/s AFSK (note that a 12–bit sample has at most four characters)

```
init_controller;
init_tx(afsk1200);
data_out := 'P = ' +
       int2str(sample(pressure),4);
send_tx;
```

`init_serial(baud:word);`
This procedure initialises the USART0 serial receiver and transmitter with 8 data bits, no parity, 1 stop bit and no flow control. The input `baud` must be equal to one of the following constant values. The data rate is given by `fosc/(16(baud+1))` where `fosc` = 12288000 (Hz) is the internal clock frequency.

`b1200` = 639 = 1200 bit/s
`b2400` = 319 = 2400 bit/s
`b4800` = 159 = 4800 bit/s
`b9600` = 79 = 9600 bit/s
`b14400` = 52 = 14491 bit/s (0.6% error)
`b19200` = 39 = 19200 bit/s
`b28800` = 25 = 29538 bit/s (2.6% error)
`b38400` = 19 = 38400 bit/s
`b57600` = 12 = 59077 bit/s (2.6% error)

`serial_out;`
This procedure sends the length 230 string variable `data_out` to the TXD pin of the ATmega328P. For example, to send a message at 9600 bit/s

```
init_controller;
init_serial(b9600);
data_out := 'Hello world';
serial_out;
```

`serial_in;`
This procedure receives the char variable `data_in` from the RXD pin of the ATmega328P. Note that this procedure will continuously wait until a character has been received. For example, to echo a received character

```
init_controller;
init_serial(b9600);
repeat
  serial_in;
  data_out := char2str(data_in);
  serial_out;
until end_of_time;
```

Bootloader

In order to program the controller using a serial cable, a bootloader is programmed into the ATmega328P. This uses the upper 2KB of the 32KB Flash memory available. In order to enter the bootloader program the following procedure should be used

1) Reset ATmega328P by forcing DTR high for 1 ms.
2) Wait 10 ms to allow the ATmega328P to configure itself and jump to the bootloader section.
3) Send the single character command “P” (without the quotes) in ASCII within 100 ms to SIN (TXD).

The serial port should be configured for 57,600 bit/s, no parity and one stop bit. If “P” is not received within 100 ms, the bootloader will reset any registers that were used and jump to the beginning of the application section (address 0000 in hexadecimal).

If “P” was successfully received by the bootloader it will send a carriage return (CR or 0D in hexadecimal) to SOUT (RXD). The following commands can then be sent to the bootloader:

“B” 2*dd “F” 128*dd
Performs a block Flash erase and load of 128 bytes (64 words) from the two byte address. If the command was successfully received, a CR will be returned. The address and data are sent low byte first. The byte address must have the seven least significant bits equal to zero.

“B” 2*dd “E” 16*dd
Performs a block EEPROM erase and load of 16 bytes from the two byte address. If the command was successfully received, a CR will be returned. The address and data are sent low byte first.

“g” 2*dd “F”
Performs a block Flash read of 128 bytes (64 words) from the two byte address. The address and data are sent low byte first. The byte address must have the seven least significant bits equal to zero.

“g” 2*dd “E”
Performs a block EEPROM read of 16 bytes from the two byte address. The address and data are sent low byte first.

"s"
Reads the three device signature bytes, low byte first. For the ATmega328P this is 1E, 95 and 0F in hexadecimal, low byte first.

"V"
Reads the two bootloader version number bytes, low byte first. For the current version, this is 00 and 00 in hexadecimal, low byte first.

"E"
Exits the bootloader and jumps to the application section. If the command was successfully received, a CR will be returned.

If an invalid command is sent to the bootloader, the character "?" is sent to SOUT. For commands longer than one character, each of the following characters must be sent within 100 ms of each other. Otherwise, the command sequence is considered to have ended and a "?" will be sent.

Tinsatcom

In order to program the CM from a computer, the tinsatcom 32–bit DOS software is provided. This is a console program written for the Windows operating system. It allows hexfiles generated by various AVR compilers to be selected and downloaded into the CM via a serial cable using the bootloader in the CM. Support for both Flash and EEPROM hexfiles are provided. The following commands are used for programming the Flash and EEPROM.

c = COM serial port. Most computers have two serial ports with 9–pin D–type connectors. The default port is COM1. Other port numbers can be selected if desired.

d = Hexfile directory. This is where the Flash and EEPROM hexfiles are located. The default directory is the same directory as tinsatcom.exe, expressed as "." without the quotes. If the full path is not given, for example "c:\tinsat", then the location is relative to the tinsatcom directory. Thus, locations such as "../tinsat_test" can be used. If there are spaces in the directory names, do not use ″ at the beginning or end of the directory name. This will cause an error. For example "/tinsat test" without the quotes is a valid directory name.

f = Flash hexfile name. This is the file name of the Flash hexfile. The file name should have the ".hex" extension, e.g., "tinsat_test.hex" without the quotes. There should be no space or ″ characters in the file name.

F = Program Flash. Entering this command will cause the selected Flash hexfile to be downloaded into the CM. The software causes the CM to enter the bootloader and uses the bootloader commands to first check the CM signature and bootloader version number. The hexfile is then written and checked one page (64 words or 128 bytes) at a time, before finally exiting the bootloader. While programming a "." is displayed for each page programmed. If there are any errors during programming, programming will stop and an error message is displayed. The default bootloader configuration of 57,600 bit/s, 8–bit wordsize, one stop bit and no parity is used.

e = EEPROM hexfile name. This is the file name of the EEPROM hexfile. This command is similar to the f command.

E = Program EEPROM. Entering this command will cause the selected EEPROM hexfile to be downloaded into the CM. The operation is similar to the F command, except that pages are 16 bytes long.

The program can also be used as a standard USART terminal. It has two models of operation, Receive Only and Interactive. Receive only mode allows the computer to continuously display characters sent from the CM. Interactive mode allows a user to send commands to the CM and display its response.

In Receive Only mode, the computer waits for 8–bit ASCII characters to be sent from the CM. If characters are sent from the CM, they are stored into a buffer of length 80. If no further characters are received in 100 ms or the buffer is full, the characters are output to the screen. To quit receive only mode, press ctrl–Q.

In Interactive mode, a prompt "> " is displayed waiting for characters to be input. Press "Enter" on the keyboard to send the characters to the CM. tinsatcom will then wait for up to 100 ms for a reply from the CM. If no reply is received, the prompt is redisplayed with an error message. If characters are sent from the CM, they are stored into a buffer of length 80. If no further characters are received in 100 ms or the buffer is full, the characters are output to the screen and the prompt redisplayed. To quit interactive mode, enter ctrl–Q after the

prompt. The character ctrl–Q appears as ◀ on the screen.

The following commands are used by the USART.

b = Baud rate. This selects the baud rate to be used. The default rate is 9600 bit/s. Depending on your particular computer, valid rates are 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600, 115200, 128000 and 256000 bit/s.

w = Word size. This is the number of data bits in each word transmitted. The default size is 8–bits. Valid wordsizes are 5, 6, 7 and 8 bits.

p = Parity. This checks if there are an odd number of errors in the data and parity bits (an even number of errors can not be detected). Valid selections are N for no parity bit, E for an even parity bit and O for an odd parity bit). The default parity is none.

s = Stop bits. This is the number of bits used to indicated the end of the word. Valid stop bits are 1 and 2. The default number is 1 stop bit.

i = Interactive. This puts the USART into interactive mode. Enter ctrl–Q to quit.

r = Receive only. This puts the USART into receive only mode. Press ctrl–Q to quit.

Other commands used are

h = Display help menu.
a = Display advanced help menu.
x = Exit tinsatcom.

When tinsatcom is first run it creates the initialisation file tinsatcom.ini (with the default parameters) in the same directory as tinsatcom.exe. Updated parameters are stored in this file, which are read the next time tinsatcom is run.

Fuse Bits

In order for the CM to function correctly, the Fuse Low Byte in the ATmega328P is programmed as 10101111 in binary or AF in hexadecimal. This corresponds to the following settings (fuse bits are active low):

Bit 7: CKDIV8 = 1 (Divide clock by 8 off)
Bit 6: CKOUT = 0 (Clock output on)
Bits 5..4: SUT[1:0] = 2 (Crystal oscillator, fast rising power)
Bits 3..0: CKSEL[3:0] = 15 (Low power crystal oscillator, 8.0–16.0 MHz frequency range)

As a bootloader is being used, the Fuse High Byte in the ATmega328P is programmed as 11011010 in binary or DA in hexadecimal. This corresponds to the following settings:

Bit 7: RSTDISBL = 1 (External reset disable off)
Bit 6: DWEN = 1 (dbugWIRE off)
Bit 5: SPIEN = 0 (SPI programming on)
Bit 4: WDTON = 1 (Watchdog timer off)
Bit 3: EESAVE = 1 (Preserve EEPROM on chip erase off)
Bits 2..1: BOOTSZ[1:0] = 1 (Boot size = 2048 bytes)
Bit 0: BOOTRST = 0 (Bootloader reset vector on)

In order to protect the bootloader against inadvertent writes using the SPM instruction, the Lock Bit Byte is programmed as 11101111 in binary or EF in hexadecimal. This corresponds to the following settings:

Bits 7..6: 3 (Not used)
Bits 5..4: BLB1[2:1] = 2 (SPM is not allowed to write to the bootloader section)
Bits 3..2: BLB0[2:1] = 3 (no restrictions for SPM and LPM in application section)
Bits 1..0: LB[2:1] = 3 (no memory lock features enabled)

The Extended Fuse Byte is left in its default setting of FF in hexadecimal.

Ordering Information

The order code is SW–TINSAT–*sctpc–n*

s = S for sensor module
c = C for controller module
t = T for transmitter module
p = P for single PCB
c = C for SMA connector
n = number of systems

If you need a custom solution for your application, please contact us for a quote.

References

[1] mikroElektronica, “mikroPascal Pro for AVR,” V1.50, http://www.mikroe.com/en/compilers/mikropascal/avr/

Small World Communications does not assume any liability arising out of the application or

use of any product described or shown herein; nor does it convey any license under its copyrights or any rights of others. *Small World Communications* reserves the right to make changes, at any time, in order to improve performance, function or design and to supply the best product possible. *Small World Communications* will not assume responsibility for the use of any circuitry described herein. *Small World Communications* does not represent that devices shown or products described herein are free from patent infringement or from any other third party right. *Small World Communications* assumes no obligation to correct any errors contained herein or to advise any user of this text of any correction if such be made. *Small World Communications* will not assume any liability for the accuracy or correctness of any engineering or software support or assistance provided to a user.

Small World Communications, 6 First Avenue,
Payneham South SA 5070, Australia.
info@sworld.com.au ph. +61 8 8332 0319
http://www.sworld.com.au fax +61 8 7117 1416