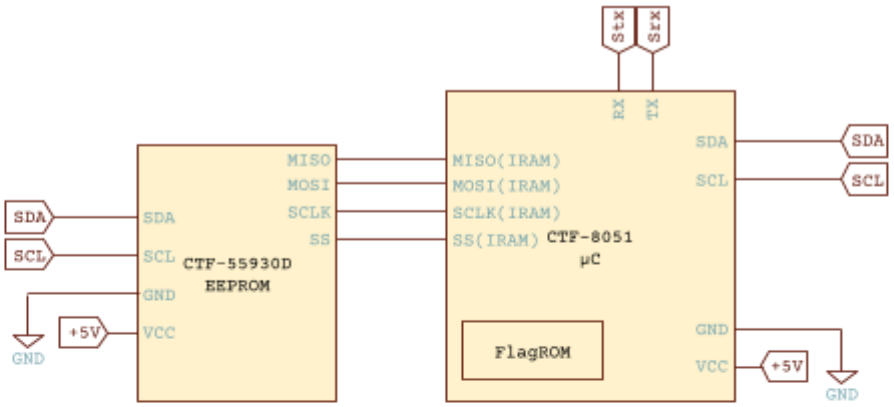

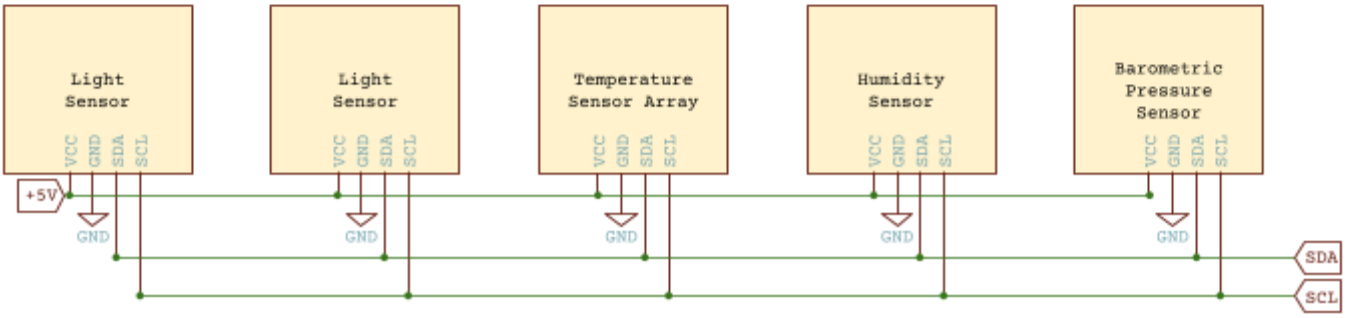

DYI Weather Station

Our company's Weather Station board gives you all you need to build your own weather station, including a convenient serial user interface based on Weathermatix2000™ protocol for reading values from sensors.

Included sensors:

Sensor	I2C 7-bit port	Data format
Humidity Sensor	119	1 byte (value from 0% to 100%)
Light Sensor (2x)	110 and 111	1 byte (value from 0 to 200)
Barometric Pressure Sensor	108	2-byte BE word (hPa)
Temperature Sensor Array	101	4x 1 byte value (°C)

Serial/UART module

The CTF-8051 microcontroller features a simplified serial communication device.

This device makes available two sets of ports for the 8051 to use to communicate over a pseudo-serial interface. Each set consists of a DATA SFR (Special Function Register) and a READY SFR.

Note: "output" denotes outgoing data from the 8051, while "input" denotes data sent to the 8051.

In case of the output SFR set (DATA @ 0xF2, READY @ 0xF3), the READY is set to 1 in case serial module is able to receive another byte of data to transmit. In practice READY will be set to 1 immediately after a byte is written to DATA since this pseudo-serial device transmits data instantaneously.

In case of the input SFR set (DATA @ 0xFA, READY @ 0xFB), the READY is set to 1 in case a byte is available to be received by the 8051. A read from the DATA SFR removes the read byte from the serial buffer and resets the READY SFR to 0 (if there is no more data at the moment) or 1 (if more data is immediately available).

- Writing to input DATA SFR is a no-op.
- Writing to input READY SFR is a no-op.
- Writing to output READY SFR is a no-op.
- Reading from the output DATA SFR port returns 0.
- Reading from the input DATA SFR if READY SFR is 0 returns 0.

Special Function Register declarations for SDK compiler:

```
__sfr __at(0xf2) SERIAL_OUT_DATA;
__sfr __at(0xf3) SERIAL_OUT_READY;
__sfr __at(0xfa) SERIAL_IN_DATA;
__sfr __at(0xfb) SERIAL_IN_READY;
```

I2C Controller module

The CTF-8051 microcontroller features a robust I2C DMA controller module.

Reading from I2C device:
1. Set XRAM buffer address in I2C_BUFFER_XRAM_LOW and I2C_BUFFER_XRAM_HIGH.
2. Set how much you want to read in I2C_BUFFER_SIZE.
3. Set device 7-bit address in I2C_ADDRESS.
4. Write 1 into I2C_READ_WRITE SFR - this will trigger the I2C transaction.
5. Check I2C_STATUS for status:
   0 - Ready / transaction completed.
   1 - Busy.
   2 - Error (device not found).
   3 - Error (device misbehaved).

Writing to I2C device:
1. Set XRAM buffer address in I2C_BUFFER_XRAM_LOW and I2C_BUFFER_XRAM_HIGH.
2. Set how much you want to read in I2C_BUFFER_SIZE.
3. Set device 7-bit address in I2C_ADDRESS.
4. Write 0 into I2C_READ_WRITE SFR - this will trigger the I2C transaction.
5. Check I2C_STATUS for status:
   [See status code description above]

Notes:

- Buffer size of 0 will just "ping" the device but not transfer any data.
- Reading from I2C_READ_WRITE returns 0.
- Writing to I2C_STATUS is a no-op.
- Top 7 bits of I2C_READ_WRITE writes are ignored.
- Top bit of I2C_ADDRESS is hardwired to 0.
- Setting I2C_BUFFER_XRAM_... and I2C_BUFFER_SIZE in such a way that would overflow XRAM will result in an overlap (i.e. writing/reading iterating address is truncated at 16 bits).
- If a device misbehaved there still might be some data copied transferred either direction.

Special Function Register declarations for SDK compiler:
```
__sfr __at(0xe1) I2C_STATUS;
__sfr __at(0xe2) I2C_BUFFER_XRAM_LOW;
__sfr __at(0xe3) I2C_BUFFER_XRAM_HIGH;
__sfr __at(0xe4) I2C_BUFFER_SIZE;
__sfr __at(0xe6) I2C_ADDRESS;  // 7-bit address
__sfr __at(0xe7) I2C_READ_WRITE;  // 0 is Write, 1 is Read
```

FlagROM module

The CTF-8051 microcontroller features an SFR-accessible ROM containing the flag. It's a very simple factory-programmed ROM device with an SFR-mapped interface.

The ROM has capacity to store up to 2048 bits (256x8).

To read the data from the ROM simply set the FLAGROM_ADDR register to the byte index and read the byte value from the FLAGROM_DATA register.

- Reading from the FLAGROM_ADDR register returns the currently set address.
- Writing to FLAGROM_DATA register is a no-op.

Special Function Register declarations for SDK compiler:
```
__sfr __at(0xee) FLAGROM_ADDR;
__sfr __at(0xef) FLAGROM_DATA;
```

CTF-55930 EEPROM

This Dual Interface EEPROM allows simultaneous access to data through both the I2C and the SPI interface without page locking. EEPROMs capacity depends on the exact model:

```
CTF-55930A   4096 bits (organized as 8x64x8)
CTF-55930B   8192 bits (organized as 16x64x8)
CTF-55930C  16384 bits (organized as 32x64x8)
CTF-55930D  32768 bits (organized as 64x64x8)
Note: Memory organization is denoted as PAGES x WORDS x BITS_PER_WORD.
```

In a typical application CTF-55930B serves as firmware storage for CTF-8051 microcontroller via the SPI(PMEM) bus.

Programming the CTF-55930

Programming this EEPROM is a two-step process. In the first step all bits are re-set to 1. In the second step a clear-mask is applied to clear selected bits to 0.
Typical process is as follows:
1. Connect all pins apart from Vcc to ground.
2. Apply 12V on Vcc pin for at least 100ms.
3. [OPTIONAL] Set the 7-bit I2C address - see the Address Setting section for details.
4. Disconnect 12V.
5. Connect the I2C interface and power on the device.
6. Using the I2C interface clear selected bits page by page (64 bytes at a time).

Note that the SPI interface is immediately active, so it's advised to hold the RST pin low on CTF-8051 until programming is complete.

I2C interface

Reading data from a 64-byte page is done in two steps:
1. Select the page by writing the page index to EEPROM's I2C address.
2. Receive up to 64 bytes by reading from the EEPROM's I2C address.

Programming the EEPROM is done by writing the following packet to the EEPROM's I2C address:

```
    <PageIndex> <4ByteWriteKey> <ClearMask> ... <ClearMask>
```

The PageIndex selects a 64-byte page to operate on. The WriteKey is a 4 byte unlock key meant to prevent accidental overwrites. Its value is constant: A5 5A A5 5A. Each ClearMask byte is applied to the consecutive bytes of the page, starting from byte at index 0. All bits set to 1 in the ClearMask are cleared (set to 0) for the given byte in the given page on the EEPROM:

```
    byte[i] ← byte[i] AND (NOT clear_mask_byte)
```

Note: The only way to bring a bit back to 1 is to follow the 12V full memory reset described in the "Programming the CTF-55930" section.