

ОРГАНИЗАЦИЯ ОБМЕНА ДАННЫМИ МИКРОКОНТРОЛЛЕРОВ PIC С ДАТЧИКАМИ ТИПА SHT1х РЕАЛИЗОВАННОГО В ПРОГРАММНОЙ СРЕДЕ FLOWCODE

Версия 1.0

Организовать обмен данными микроконтроллеров PIC с датчиками типа SHT1х в программной среде Flowcode (опрос) можно как программно, так и аппаратно используя шину I2C.

В Flowcode существует компонент "I2C_Master", который позволяет работать с шиной I2C программно или аппаратно.

Компонент довольно простой и интуитивно понятный, однако для правильной работы с ним, необходимо, хотя бы минимально, ознакомиться с шиной I2C в рамках описания ее в Datasheet на МК.

Общаться с датчиком довольно просто, посылаем ему управляющий байт (команду), в ответ получаем либо состояние регистра управления (статуса), либо значение температуры, либо значение влажности в зависимости от содержимого байта команды.

Датчик поддерживает несколько команд (первые три бита 000 - это адрес, и он всегда равен нулям):

- 000 00011 - измерить температуру;
- 000 00101 - измерить влажность;
- 000 00111 - прочитать регистр состояния;
- 000 00110 - записать регистр состояния;
- 000 11110 - программный сброс.

Для того чтобы датчик начал работать, нужно произвести следующие действия:

1. Выдать стартовую последовательность;
2. Передать байт команды.
3. Выдержать паузу (по Datasheet) или выполнить любые вычисления и продолжить.
4. Прочитать 2 или 3 байта (2 байта - данные и 1 байт - контрольная сумма).
5. Преобразовать полученные 2 байта в температуру или влажность. Алгоритм пересчета в Datasheet.

Микроконтроллер выступает в качестве ведущего устройства, датчик в качестве ведомого. Обмен между устройствами организуется в алгоритмах работы шины I2C.

Однако нужно обратить внимание, что датчик работает не совсем в алгоритмах шины I2C - нет стартовой и стоповой последовательности, согласно требований протокола I2C. Есть только стартовая последовательность, но своеобразной формы (рис.1) (см. Datasheet). Форму стартовой последовательности датчика нужно выдержать строго по Datasheet. Временные рамки стартовой последовательности могут быть больше указанных, главное не меньше.

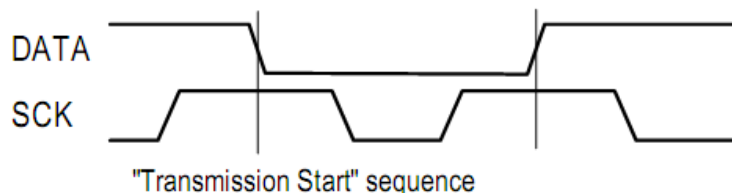


Рисунок 1 – Стартовая последовательность датчика SHT1х.

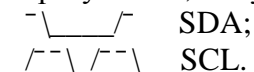
Поскольку стартовая последовательность есть определяющая, то остановимся на ней подробнее. В зависимости от реализации обмена (программно или аппаратно) стартовую последовательность можно формировать несколькими способами.

ПРОГРАММНЫЙ ОБМЕН

Реализация программного обмена (на основе макроса "I2C_Master") не накладывает ни каких ограничений на способ формирования импульсов стартовой последовательности (в дальнейшем "Старт").

1. "Старт" можно сформировать через "Блок Output", используя опцию Single Bit для записи бита в порт. Соответствие портов определяется назначенными портами для шины I2C. Формирование импульсов Старт нужно производить последовательно, чередуя выдачи уровней на выходах портов SCL и SDA (SCL_┐, SDA_┐, SCL_┘, SCL_┐, SDA_┐, SCL_┘). При высоких частотах кварца, рекомендуется вставлять между выдачами короткие паузы, порядка 10 мкс (однако все это подбирается практически).

В результате, получится:

 SDA;
SCL.

Пример формирования импульсов Старт приведен в программе.

2. "Старт" можно сформировать, описав его на СИ. Описание на СИ можно выдернуть из макросов компонента I2C. Это совсем не сложно, потому, что в макросах уже определены все условия и назначены порты SCL и SDA.

В макросе [Defines] компонента I2C введены определения, на их основе делаем свои (это необходимо для того, чтобы формировать импульс, не влезая в макросы компонента I2C). Таким образом, получаем самостоятельный макрос формирования Старт.

Ниже приведены определения по [Defines] (см. макрос компонента I2C).

```
#define I2C_PORT      portc // порт и его пины определяем согласно назначениям
#define I2C_TRIS      trisc
#define SCL           3
#define SDA           4

#define I2C_DELAY      delay_us(5);

#define SET_SCL        set_bit(I2C_TRIS,SCL); //Configure SCL as Input
#define CLEAR_SCL      { clear_bit(I2C_TRIS,SCL); clear_bit(I2C_PORT,SCL); } //to 0
#define SET_SDA        set_bit(I2C_TRIS,SDA); //Configure SDA as Input
#define CLEAR_SDA      { clear_bit(I2C_TRIS,SDA); clear_bit(I2C_PORT,SDA); } //to 0
```

После определения, Старт будет описан следующим образом.

```
SET_SCL; // ┐
I2C_DELAY; //Delay
CLEAR_SDA; // ┐
I2C_DELAY;
CLEAR_SCL; // ┘
I2C_DELAY;
SET_SCL; // ┐
I2C_DELAY;
SET_SDA; // ┐
I2C_DELAY;
CLEAR_SCL; // ┘
// ┐ ┐ ┘ ┐ ┐ ┘ ┐ SDA;
// ┐ ┘ ┐ ┘ ┐ ┘ ┐ SCL.
```

Все это помещаем в блоки C Code в отдельный макрос Flowcode (рис.2), и вызываем в нужном месте.

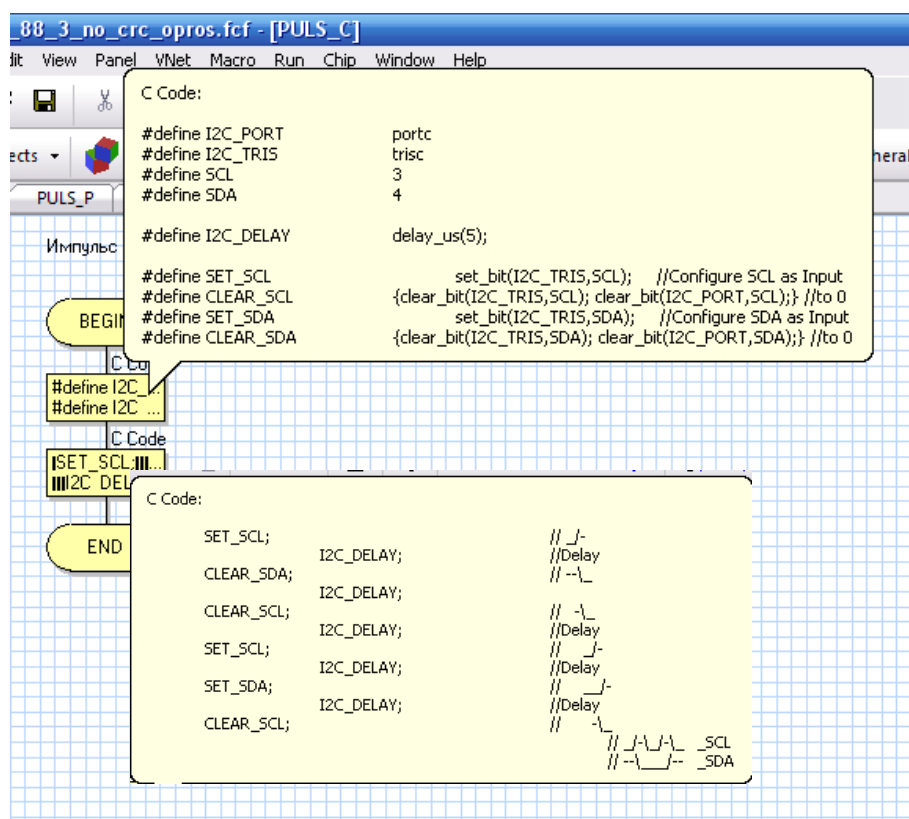


Рисунок 2 – Макрос формирования импульсов "Старт"

3. "Старт" можно сформировать на основе макросов MI2C_Restart и MI2C_Stop компонента I2C. Однако макрос MI2C_Stop требует небольших программных изменений. Макрос MI2C_Restart используется без изменений.

Опираясь на способ формирования импульсов "Старт", "Стоп" и "Рестарт" (согласно описанию протокола I2C и описанию МК) можно определить, что последовательность импульсов "Рестарт" и "Стоп" стандартной шины I2C дает почти похожую последовательность "Старт" для датчика (рис.3). Однако окончание последовательности требует небольшой доработки, это несложно сделать, потому что обмен программный и можно изменить макрос "Стоп".

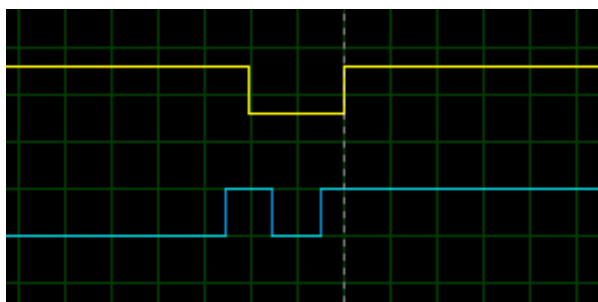


Рисунок 3 – Формирование импульсов макросами "Рестарт" и "Стоп"

Доработка макроса "Стоп" заключается в исключении **трех строк** и добавлении **двух**.

```
//      %i_CLEAR_SCL;                                //Set SCL Low
//      %i_CLEAR_SDA;                                //Set SDA Low
      %i_MX_I2C_DELAY;
      %i_SET_SCL;                                    //Set SCL High
      %i_MX_I2C_DELAY;
      %i_SET_SDA;                                    //Set SDA High
      %i_MX_I2C_DELAY;                                //my mim
      %i_CLEAR_SCL;                                //my mim
//      delay_ms(10);                                //Wait before reusing the I2C BUS
```

На рис. 4 приведена форма доработанной последовательности "Рестарт" и "Стоп".

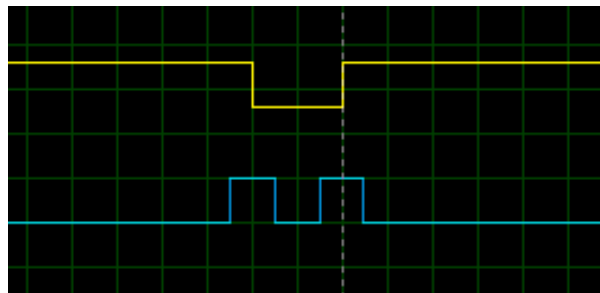


Рисунок 4 – Формирование импульсов макросами "Рестарт" и, доработанным, "Стоп"

Теперь создаем свой макрос в Flowcode, выглядит макрос "Старт" датчика следующим образом (рис 5).

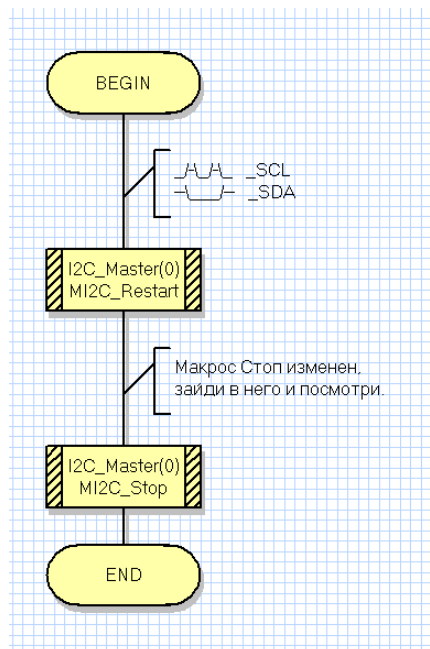


Рисунок 5 – Макрос формирования импульсов "Старт"

АППАРАТНЫЙ ОБМЕН

Аппаратный обмен накладывает определенные условия на работу порта, через который ведется обмен. В частности на два вывода SCL и SDA.

Согласно алгоритму работы шины I2C (описание МК) при выборе любого режима I2C, выводы SCL и SDA должны быть настроены на вход установкой соответствующих битов регистров TRIS в "1". Включение модуля I2C (MSSP) МК осуществляется установкой бита SSPEN в "1", после чего выводы SCL и SDA подключаются к модулю MSSP. В этом режиме доступ к выводам, кроме как через модуль MSSP закрыт.

Таким образом, на формирование импульса "Старт" через выводы SCL и SDA накладываются ограничения. Для того чтобы корректно формировать импульсы "Старт", выше приведенными способами, необходимо управлять работой модуля MSSP непосредственно через его регистры – включать и отключать модуль, и тем самым разрешать работу выводов на выдачу.

Поэтому самым удобным способом формирования импульсов "Старт" остается третий способ, потому что при аппаратном обмене формирование импульсов "Старт", "Стоп" и "Рестарт" тоже аппаратное (через модуль MSSP). Нет необходимости управлять работой модуля MSSP и использовать еще какие-нибудь способы формирования стартовой последовательности. Однако макрос "Стоп" и здесь требует доработки. Это самая простая доработка, необходимо закомментировать только одну строку.

```
cr_bit(pir1,SSPIF);  
st_bit(sspcon2,PEN);  
while(ts_bit(sspcon2,PEN));  
// delay_ms(10);
```

Таким образом мы сформировали импульсы Старт в разных режимах работы модуля.

ОПРОС ДАТЧИКА

Есть еще один интересный момент работы датчика – это время преобразования данных. В зависимости от установленной разрешающей способности и, в зависимости что преобразует датчик, показания температуры или влажности, это время меняется от 11 мс до 200 мс (по разным Datasheet времена паузы указаны разные).

Время, необходимое для преобразования данных, можно установить через паузу, длительность паузы можно подобрать практически.

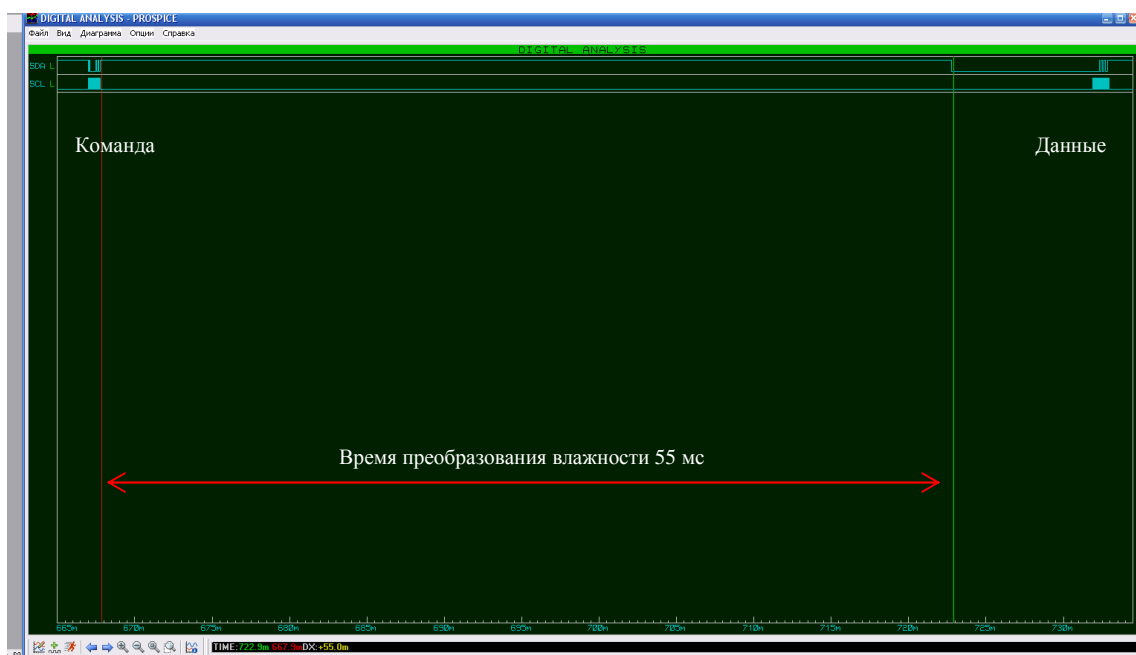
Однако есть возможность программным путем определить время окончания преобразования данных датчиком.

Пока датчик выполняет преобразование, он держит шину в высоком состоянии, когда датчик заканчивает преобразование, он отпускает шину данных, а поскольку вывод данных SDA все время настроен как вход, то можно контролировать состояние шины.

Если в процессе преобразования сканировать значение шины (читать шину), то можно определить время окончания преобразования.

Ниже приведена реализация опроса на СИ. Опрос шины организован в цикле по условию, период опроса 4 мс (можно задать любой разумный). Если по какой либо причине датчик не отпустит шину, то через 128 циклов (0,5 сек) опрос прервется через break и тем самым выведет программу из заикливания (это можно корректировать).

```
static char in;
static char counter;
in=1;
counter=0;
while (in == 1)
{
    in = portc.4;
    delay_ms(4);
    counter++;
    if (counter == 0x7F)           // Security: if the conversion is not completed
        break;                   // after > 0.5 sec -> break.
}
```



Примечание. Однако, самым наилучшим способом формирования паузы, является процесс организации полезных вычислений (или, например, динамического отображения на 7-сегментном индикаторе), а затем возврат к чтению датчика через указанное в Datasheet (или вычисленное практически) время.

ЧТЕНИЕ ДАННЫХ И ИХ ПРЕОБРАЗОВАНИЕ

Для организации чтения данных необходимо выполнить следующую последовательность (см. Datasheet).

1. Старт.
2. Команда.
3. Пауза.
4. Чтение байта (1).
5. Чтение байта (2).
6. Чтение байта (3).

Формирование импульсов "Старт" (пункт 1) и формирование Паузы-опроса (пункт 3) мы рассмотрели. Рассмотрим формирование команд и процедуру чтения байтов с использованием компонента "I2C_Master".

Для **передачи** данных (байта команды) по шине I2C используется макрос MI2C_Transmit_Byte. В макрос помещают код команды, напрямую или через переменную. После выполнения макроса, он возвращает результат своей работы – чтение импульса АСК. Если передача прошла успешно, то датчик подтвердит это (опустив шину) и возвращаемое значение будет равно нулю, в противном случае единице (рис. 6).

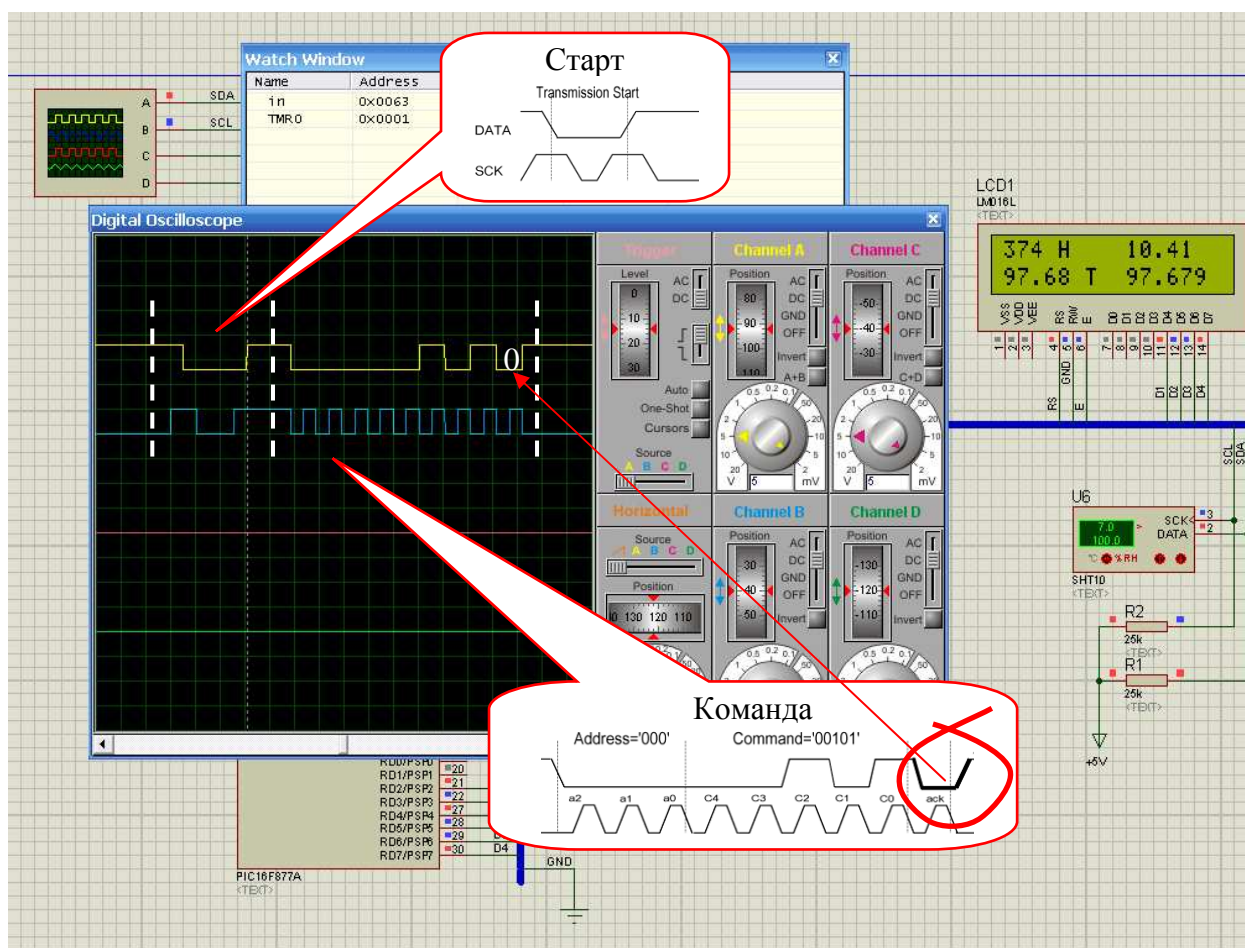


Рисунок 6 – Передача команды

Для **чтения** данных по шине I2C используется макрос MI2C_Receive_Byte. Макрос читает данные и выдает в шину I2C уведомление (бит подтверждения) Send Nack или Send Ack. Если бит подтверждения ACK не сформирован (высокий уровень SDA), то датчик прекращает выдачу байтов и его логика настраивается на обнаружения импульсов Старт.

Бит подтверждения формирует пользователь (программист) в зависимости от логики работы устройства. Подтверждение определяется в поле Last макроса и может иметь значения 0 или 1 (рис. 7).

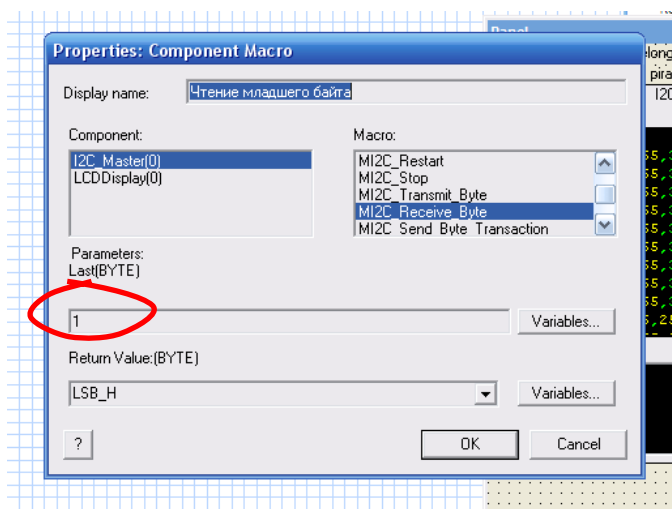


Рисунок 7 – Форма компонента I2C макроса MI2C_Receive_Byte

В данном случае, мы хотим организовать прием двух байтов (значение температуры или влажности) и игнорировать третий байт – контрольную сумму. Для этого последовательно размещаем два макроса MI2C_Receive_Byte. В каждом макросе мы читаем значения, полученные от датчика, и помещаем их в переменные. В первом макросе параметр Last определяем как "0", а во втором как "1". Таким образом, после приема второго байта датчик остановит передачу и настроится на прием Старта (рис. 8). Если нужно читать три байта, то Last определяем как "1" в третьем макросе.

Преобразование данных полученных от датчика начинается с формирования их полного значения. Согласно Datasheet значения от датчика могут поступать в различном формате. По умолчанию датчик настроен на разрешение 12 бит по влажности и 14 бит по температуре. После преобразования данных датчик помещает значения температуры (влажности) в два байта. Принятые байты нужно объединить в переменную типа Int. Например T и H переменные типа Int.

$$T = (MSB_T \ll 8) | LSB_T$$

$$H = (MSB_H \ll 8) | LSB_H$$

Теперь переменные T и H содержат полные значения данных, переданные датчиком. Дальнейшее преобразование проводят согласно Datasheet. Для более точного преобразования необходимо воспользоваться документом:

"Non-Linearity_Compensation_Humidity_Sensors_E.pdf".

Пример преобразования данных приведен в программе.

Для изменения параметров разрешения по температуре и влажности необходимо подать команду 00000110 - запись в регистр состояния, а затем передать байт 00000001 (см Datasheet).

