**Общее описание**

Каждому устройству на шине присваивается уникальный адрес <addrХХ>.

Каждое ведомое (slave) устройство должно иметь идентификатор (<addrХХ>), определяющий функциональность устройства (например, 1 – МШУ, 2-Синтезатор, 3-контроллер ключей итд), строковое имя. Причем одинаковых по функционалу устройств может быть несколько(например 2 МШУ)

Таблица 1

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **addr** | deviceType | **Примечание** |
| HF\_block | 1 | HF\_block | master |
| LNA1 | 2 | LNA | slave |
| KUkonv | 3 | KUkonv | slave |
| MRL | 4 | MRL | slave |
| Triplexer | 5 | Triplexer | slave |
| IKS | 6 | IKS | slave |
| LNA2 | 7 | LNA | slave |

Мастер хранит во флеш-памяти список устройств с которыми он работает (смотри таблицу 1).

Тип определяется функционалом и составом смонтированных элементов согласно таблице 2

Таблица 2

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№пп** |  | deviceType | **UART** | **RS232** | **RS485** | **USB** | **Термо-**  **датчик** | **нагрев** | **датчик тока** | **SPI 1** | **SPI 2** | **GPIO** |
| 1 | Master | HF\_block | да | да | да | да | нет | нет | нет | да | да | 6 |
| 2 | Slave | LNA | да | да | да | нет | 2 | да | да | нет | нет | - |
| 3 | Slave | KUkonv | да | да | да | нет | нет | нет | нет | да | да | - |
| 4 | Slave | MRL | да | да | да | нет | нет | нет | нет | да | да | - |
| 5 | Slave | Triplexer | да | да | да | нет | 2 | да | да | нет | нет | - |
| 6 | Slave | IKS | да | да | да | нет | нет | нет | нет | да | да | 27 |

Должна быть возможность считать этот список (таблица 1) и изменить (добавить или удалить устройство, а так же изменить тип, адрес или имя любого устройства).

**Запрос списка:**

По команде GET\_DEVICE\_LIST мастер выдает сохраненный список устройств в формате

DeviceList:<name1>,<addr1>,<deviceType1>;….<nameN>,<addrN>,<deviceTypeN>\r\n\r\n

**Запрос состояния**

Мастер автономно опрашивает все устройства из своего списка с темпом раз в секунду.

По запросу Мастер GET\_STATE выдает информацию о каждом подключенном устройстве в виде следующего списка:

State:<name1>,<addr1>,<deviceType>,<errorCode>,<parameters>\r\n

<name2>,<addr2>,<deviceType>,<errorCode>,<parameters>\r\n

<name3>,<addr3>,<deviceType>,<errorCode>,<parameters>\r\n

<name4>,<addr4>,<deviceType>,<errorCode>,<parameters><CRC>\r\n\r\n

Name – строковое имя устройства

Addr – адрес устройства на шине

deviceType – тип устройства

errorCode – код ошибки (ACK0 – ошибок нет) *На усмотрение Геннадия*

parameters – параметры, настройки и состояние устройства. Это поле зависит от функциональности устройства

*Параметры (parameters):*

Для плат, имеющих датчик температуры:

1 ) выдается текущая температура и состояние нагревателя (вкл/выкл)

TEMP=NONE/ERR/{value},HEAT=ON/OFF

TEMP – состояние/показания датчика температуры (none, если датчик отстутствует, err-неисправность)

Примечание : Датчиков несколько, выдавать от Slave целесообразно среднюю температуру

2) Пиковое значение на датчике тока

Контроллер GPIO (он же контроллер ключей) – состояние управляющих выходов GPIO

GPIO= {value}

Value – 32хбитное число, кодирующее состояние выводов GPIO. Расписать свободные выводы

**Команды:**

1. Включить/выключить обогрев

HEAT,ON/OFF,<addr>,<frameNo>\r\n

1. Включить/выключить имитатор

SIGNAL\_SIMULATOR,<addr>,<en/dis>,<frameNo>\r\n

1. Загрузить синтезатор:

Осуществляется автоматически при подаче питания на контроллер с задержкой 2 сек. А также принудительно по внешней команде

3.1) «***Задать диапазон***»

<addr><cmdId><frameNo><GNSSRange>,

по которой контроллер будет формировать команду синтезатору «SetAdf14»

(SetAdf14 <Addr> <14\_Registers\_values>) Значения этих 14-ти регистров представлены в таблице 3.

Таблица 3.

|  |  |  |  |
| --- | --- | --- | --- |
| № регистра | Значение для L1 | Значение для L2 | Значение для L5 |
| 0 | 0x302510 | 0x302740 | 0x302790 |
| 1 | 0xF53F7C1 | 0xBD70A31 | 0xDAE1471 |
| 2 | 0x1D007D2 | 0x540192 | 0x440192 |
| 3 | 0x3 | 0x3 | 0x3 |
| 4 | 0x32008984 | 0x32008984 | 0x32008984 |
| 5 | 0x800025 | 0x800025 | 0x800025 |
| 6 | 0x35008476 | 0x35008476 | 0x35008476 |
| 7 | 0x60000E7 | 0x60000E7 | 0x60000E7 |
| 8 | 0x15596568 | 0x15596568 | 0x15596568 |
| 9 | 0x7047CC9 | 0x7047CC9 | 0x7047CC9 |
| 10 | 0xC0067A | 0xC0067A | 0xC0067A |
| 11 | 0x61200B | 0x61200B | 0x61200B |
| 12 | 0x15FC | 0x15FC | 0x15FC |
| 13 | 0xD | 0xD | 0xD |

3.2) Задать «***сдвиг частоты***»

<addr><cmdId><frameNo><Frequency>,

в которую компьютер передает частоту (дробное число), а контроллер пересчитывает частоту в значения регистров синтезатора (номера регистров: 0, 1, 2, 10, 13) в соответствии с командой (SetAdf5 <addr> <5\_Registers\_values>). Алгоритм пересчета приведен в приложении А

и продублирован в репозитории https://bitbucket.org/semen966/registers/src/master/Registers/Program.cs.

*Примечание: предусмотреть возможность загрузки содержимого регистров (L1Command, L2Command, L5Command) извне (ПЭВМ через USB master) и сохранять в памяти программ контроллера.*

1. Установить состояние выводов GPIO (хранить в контроллере последнее состояние. При включении применять его.)

SET\_GPIO,<addr>,<value>,<frameNo>\r\n

frameNo – номер пакета. Ответная квитанция на команду должна содержать такой же номер

На каждый запрос должно быть получено подтверждение.

Приложение А

Алгоритм пересчета частоты в значения регистров синтезатора

*/// <summary> Частота гетеродина для L1, Мгц </summary>*

**const** **double** FG\_L1 = 5939.58d;

*/// <summary> Частота гетеродина для L2, Мгц </summary>*

**const** **double** FG\_L2 = 6287.40d;

*/// <summary> Частота гетеродина для L5, Мгц </summary>*

**const** **double** FG\_L5 = 6338.55d;

*/// <summary> Частота фазового детектора, МГц</summary>*

**const** **double** FD = 10d;

*/// <summary> Шаг перестройки, Гц </summary>*

**const** **int** STEP = 500;

*/// <summary> Частотный диапазон </summary>*

**enum** GNSSRange { L1, L2, L5 };

*/// <summary>*

*/// Расчет значения регистров*

*/// <param name="fd">Частота Допплра, Гц (изменяется от -150000 до 150000 с шагом 500)</param>*

*/// <param name="range">Диапазон</param>*

**static** **void** RegistersCalc(**int** fd, GNSSRange range, **out** **int** r0, **out** **int** r1, **out** **int** r2, **out** **int** rD)

{

*//Выбор частоты гетеродина*

**double** fg = 0;

**switch** (range)

{

**case** GNSSRange.L1: fg = FG\_L1; **break**;

**case** GNSSRange.L2: fg = FG\_L2; **break**;

**case** GNSSRange.L5: fg = FG\_L5; **break**;

}

*//Отношение частоты гетеродина к частоте фазового детектора*

**double** n = (fg + (fd / 1000000d)) / FD;

*//целая часть - пишется в регистр 0.*

**int** iNT = (**int**)n;

*//номер регистра*

**int** registerNum = 0x000000;

*//значение регистра 0*

r0 = (iNT << 4) | 0x300000 | registerNum;

*//дробная часть n*

**double** remain1 = n - iNT;

*//дробная часть смещенная*

**double** frac = remain1 \* Math.Pow(2, 24);

*//дробная часть увеличенная, переведенная в целое*

**int** frac1 = (**int**)frac;

registerNum = 0x000001;

*//значение регистра 1*

r1 = frac1 << 4 | registerNum;

*//дробная часть, оставшаяся от frac после отделения frac1*

**double** remain2 = Math.Round(frac-frac1,5);

**int** mod2 = (**int**)Math.Pow(FD, 7) / STEP;

**int** frac2 = (**int**)(remain2 \* mod2);

*//Поиск наибольшего общего делителя*

**int** nod = GetNOD(frac2,mod2);

*//Сокращаем*

frac2 /= nod;

mod2 /= nod;

*//стираем ненужниые биты*

**int** lSB\_Frac = frac2 & 0b00000000000000000011111111111111;

**int** lSB\_Mod = mod2 & 0b00000000000000000011111111111111;

registerNum = 0x000002;

*//значение регистра 2*

r2 = ((lSB\_Frac << 18) | lSB\_Mod << 4) | registerNum;

*//стираем ненужниые биты*

**int** mSB\_Fraq = frac2 & 0b00001111111111111100000000000000;

**int** mSB\_Mod = mod2 & 0b00001111111111111100000000000000;

registerNum = 0x00000D;

*//значение регистра D*

rD = ((mSB\_Fraq << 4) | (mSB\_Mod >> 10)) | registerNum;

}

*/// <summary> Поиск наибольшего общего делителя </summary>*

**static** **int** GetNOD(**int** a, **int** b)

{

**while** (b != 0)

{

**var** temp = b;

b = a % b;

a = temp;

}

**return** a;

}

}