МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ УНИТАРНОЕ ПРЕДПРИЯТИЕ

МОСКОВСКОЕ ОПЫТНО-КОНСТРУКТОРСКОЕ БЮРО «МАРС»

Григорьева О.Л., Селиванов Ю.И.

**Методическое пособие к лабораторным работам по курсу**

**«СИСТЕМНОЕ БОРТОВОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ»**

Москва, 2020

**Содержание**

[Введение 3](#_Toc42178182)

[1 Лабораторная работа №1. Работа с последовательно-параллельным каналом блока управления и контроля 4](#_Toc42178183)

[1.1 Цель работы 4](#_Toc42178184)

[1.2 Теоретическая часть 4](#_Toc42178185)

[1.3 Практическая часть 6](#_Toc42178186)

[1.4 Порядок выполнения работы 7](#_Toc42178187)

[1.5 Пример написания программы 7](#_Toc42178188)

[2 Лабораторная работа №2. Работа с мультиплексным каналом передачи данных в блоке вычислительных устройств 10](#_Toc42178189)

[2.1 Цель работы 10](#_Toc42178190)

[2.2 Теоретическая часть 10](#_Toc42178191)

[2.3 Практическая часть 12](#_Toc42178192)

[2.4 Порядок выполнения работы 13](#_Toc42178193)

[2.5 Пример формирования структуры для драйвера 13](#_Toc42178194)

# **Введение**

Все устройства, входящие в состав системы управления (СУ), имеющие бортовую операционную систему организуют взаимодействие этого самого устройства с различными подсистемами, входящими в состав СУ.

Так, например, системное БПО блока вычислительных устройств (БВУ) организовывает взаимодействие БВУ по мультиплексному каналу обмена данными с другими устройствами СУ, а системное БПО блока управления и контроля (БУК) организовывает взаимодействие между подсистемами по последовательно-параллельному каналу.

Данное пособие предназначено для получения студентами практических навыков по разработке системного БПО в части организации взаимодействия между различными подсистемами СУ.

# **1 Лабораторная работа №1. Работа с последовательно-параллельным каналом блока управления и контроля**

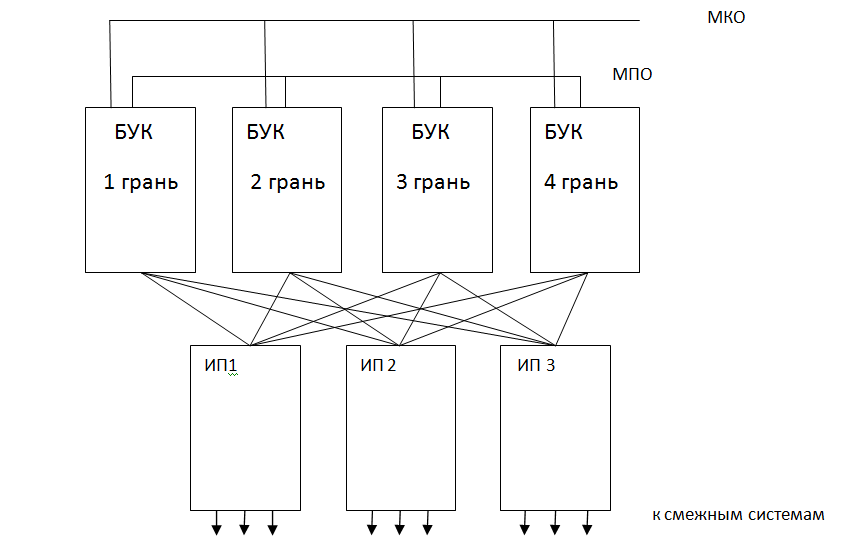
**1.1 Цель работы**

Цель данной лабораторной работы – получить практические навыки по разработке системного бортового программного обеспечения на примере организации взаимодействия между подсистемами по последовательно-параллельному каналу в блоке управления и контроля (БУК).

**1.2 Теоретическая часть**

Логика информационного взаимодействия между подсистемами БУК: частота, наполняемость массивов обмена с ОУ, - определяется протоколами информационного обмена между подсистемами. В протоколах определены форматы обменов с ОУ: прием, выдача, синхронизация; количество обменов за цикл; количество слов в каждом обмене; пословная роспись массивов обмена.

Структурная схема БУК представлена на рисунке 1.



*Рисунок 1. Структурная схема БУК.*

Каждая грань БУК может управлять каждой гранью ИП, но одновременно работать с одной гранью ИП может только одна грань БУК. Для предотвращения одновременного обращения нескольких граней БУК к одной грани ИП формируется, так называемая, маска доступных граней для каждой грани БУК.

Маска может принимать следующие значения:

0 – управление ИП недоступно для данной грани;

1 – грань БУК управляет первой гранью ИП;

2 – грань БУК управляет второй гранью ИП;

3 – грань БУК управляет первой и второй гранями ИП;

4 – грань БУК управляет третьей гранью ИП;

5 – грань БУК управляет первой и третьей гранями ИП;

6 – грань БУК управляет второй и третьей гранями ИП;

7 – грань БУК управляет всеми гранями ИП.

В штатной работе всеми гранями ИП управляет ведущая грань БУК.

При включении БУК и нормативных результатах тестирования, ведущей назначается первая грань. При дальнейшей работе может происходить или плановая смена ведущей грани по командам с наземного комплекса управления, или автоматическая смена ведущей грани в случае ее отказа.

Для определения статуса граней БУК используется параметр – слово конфигурации, представляющий собой 16-разрядное слово (таблица 1). Ведущей является грань, у которой статус равен «7».

Таблица 1 – Слово конфигурации БУК

|  |  |  |  |
| --- | --- | --- | --- |
| № бита | Обозначение | Наименование | Примечание |
| 0 | configd\_buk | Код состояния 1-й грани БУК | **7** – активное ОУ;  **4** – пассивное ОУ;  **2** – холодный резерв  приоритета 1;  **1** – холодный резерв  приоритета 2;  **0** – холодный резерв  приоритета НКУ; |
| 1 |
| 2 |
| 3 |
| 4 | Код состояния 2-й грани БУК |
| 5 |
| 6 |
| 7 |
| 8 | Код состояния 3-й грани БУК |
| 9 |
| 10 |
| 11 |
| 12 | Код состояния 4-й грани БУК |
| 13 |
| 14 |
| 15 |

**1.3 Практическая часть**

Разработчику системного бортового программного обеспечения поступают исходные данные, слово конфигурации БУК и номер грани БУК.

Исходные данные представляют собой фрагмент протокола информационного взаимодействия бортового программного обеспечения БУК со смежными подсистемами (таблица 2).

Таблица 2 – Исходные данные для программирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Код сообщения | Идентификатор запроса | Адрес в микросхеме | Идентификатор команды | Тип обмена |
| 30 | zap1/0,1,2 = 5 | F h | kom1 | выдача |
| 44 | zap2/3,4,5 = 6 | 1E h | kom2 | прием |
| 51 | zap3/12 = 1 | 23 h | kom3 | выдача |

Маска доступных граней устанавливается в зависимости от слова конфигурации. В зависимости от параметров *zap1, zap2, zap3* необходимо сформировать массив значений из столбца «Код сообщений». Причем, в полученном массиве кодов сообщений не должно быть нулевых значений между кодами. Затем, руководствуясь столбцом «Тип обмена», следует или записать информацию в регистр (столбец «Адрес в микросхеме»), или считать информацию из регистра. Причем записывать информацию может только ведущая грань.

*Примечание* – для упрощения принимается, что регистр – это массив, а адрес – это элемент массива с таким индексом.

**1.4 Порядок выполнения работы**

1. Разработать программный модуль, состоящий из двух подпрограмм:
2. Подпрограмма, которая исходя из таблицы запросов (таблица 2, столбец 3), формирует массив текущего обмена для драйвера (последовательность кодов сообщений).
3. Драйвер, обеспечивающий выдачу/прием согласно таблице запросов (таблица 2). Причем, выдачу производит только ведущая грань.
4. Составить технические требования (ТТ) на данный программный модуль (см. дополнение отдельным файлом), содержащий:
   1. Блок-схему алгоритма.
   2. Перечень входных и выходных параметров.
5. Составить контрольные примеры к программному модулю. Контрольные примеры должны охватывать все «ветки» программы.

В качестве **отчета** о проделанной работе необходимо предоставить: текст программного модуля на языке Си (исходный текст), ТТ и контрольные примеры к программному модулю.

**1.5 Пример написания программы**

Запись *zap1/0,1,2 = 5* означает, что необходимо выделить значения разрядов 0, 1 и 2, исходя из двоичного представления параметра *zap1*, и результат должен быть равен «5». Буква «*h*» после числа или «*0x*» перед числом означает, что число представлено в шестнадцатеричном виде.

Выделение разрядов:

*zap1* & 7 – выделение разрядов 0, 1, 2;

*zap2* & 56 – выделение разрядов 3, 4, 5;

*zap3* & 4096 – выделение 12-го разряда.

Однако, нагляднее использовать шестнадцатеричное представление чисел:

*zap1* & 0x7 – выделение разрядов 0, 1, 2;

*zap2* & 0x38 – выделение разрядов 3, 4, 5;

*zap3* & 0x1000 – выделение 12-го разряда.

Проверка условия запросов на языке Си будет выглядеть следующим образом:

*if((zap1 & 0x7) == 5)*

*if(((zap2 & 0x38) >>3) == 6)*

*if(zap3 & 0x1000) или if((zap1 & 0x1000) != 0)*

Формирования массива для драйвера

*i=0;*

*if((zap1 & 0x7) == 5)*

*mass[i++]=30;*

*if(((zap2 & 0x38) >>3) == 6)*

*mass[i++]=44;*

*if(zap3 & 0x1000)*

*mass[i]=51;*

Пример варианта построения маски доступных граней исполнительных плат БУК:

*if((configd & 0xf) == 7)*

*n\_ved=1;*

*else if((configd & 0xf0) == 0x70)*

*n\_ved=2;*

*else if((configd & 0xf00) == 0x700)*

*n\_ved=3;*

*else if((configd & 0xf000) == 0x7000)*

*n\_ved=4;*

*if(n\_gr\_buk == n\_ved)*

*maska=7;*

где *configd* - слово конфигурации;

*n\_ved* - номер ведущей грани;

*n\_gr\_buk* - номер грани БУК;

*maska* - маска доступных граней.

Пример программы выдачи/приема:

*for(i=0;i<3;i++)*

*{*

*if((mass[i]==30) && (maska == 7))*

*mas\_ip[0xf]=kom1;*

*else if(mass[i]==44)*

*kom2=mas\_ip[0xe1];*

*else if((mass[i]==51) && (maska == 7))*

*}*

# **2 Лабораторная работа №2. Работа с мультиплексным каналом передачи данных в блоке вычислительных устройств**

**2.1 Цель работы**

Цель данной работы – получить практические навыки по разработке системного бортового программного обеспечения на примере организации взаимодействия блока вычислительных устройств (БВУ) по мультиплексному каналу обмена данными со смежными подсистемами.

**2.2 Теоретическая часть**

Логика информационного взаимодействия и форматы обменов с подсистемами по мультиплексному каналу обмена (МКО) определяется «Сводным протоколом информационного взаимодействия».

В штатной работе одна из четырех граней БВУ является контроллером (ведущей гранью) на МКО, т.е. именно она инициирует все обмены. А другие включенные грани являются мониторами, т.е. они могут только подслушивать всю информацию, которая передается по МКО.

При включении БВУ и нормативных результатах тестирования ведущей назначается первая грань. При дальнейшей работе может происходить или плановая смена ведущей грани по команде с наземного комплекса управления, или автоматическая смена ведущей грани в случае ее отказа.

Для определения статуса граней БВУ используется параметр, который называется словом конфигурации. Данный параметр представляет собой 16-разрядное слово (таблица 1). Контроллером является грань, у которой статус равен «9».

Для инициализации обмена по каналу МКО необходимо сформировать для драйвера МКО структуру, представленную в таблице 2.

Таблица 1 – Слово конфигурации БВУ

|  |  |  |  |
| --- | --- | --- | --- |
| № бита | Обозначение | Наименование | Примечание |
| 0 | configd | Код состояния 1-й грани БВУ | **9** – контроллер на  внутренней шине  и активное ОУ на  внешней шине;  **6** – монитор на  внутренней шине  и пассивное ОУ  на внешней шине;  **2** – холодный резерв  приоритета 1;  **1** – холодный резерв  приоритета 2;  **0** – холодный резерв  приоритета НКУ; |
| 1 |
| 2 |
| 3 |
| 4 | Код состояния 2-й грани БВУ |
| 5 |
| 6 |
| 7 |
| 8 | Код состояния 3-й грани БВУ |
| 9 |
| 10 |
| 11 |
| 12 | Код состояния 4-й грани БВУ |
| 13 |
| 14 |
| 15 |

Таблица 2 – Формат структуры для драйвера МКО

|  |  |  |
| --- | --- | --- |
| КС i | Указатель на массив для слов данных | Указатель на СЗО для данного КС |
| … | … | … |
| КС n | Указатель на массив для слов данных | Указатель на СЗО для данного КС |
| 0 | Признак конца массива | |

где – КС – командное слово, формат которого представлен на рисунке 1;

СЗО – слово завершения обмена, которое указывает на качество проведенной операции: были ли ошибки при передаче информации.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Адрес | | | | | Прием/ выдача | Подадрес | | | | | Количество слов | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Выдача – 0  Прием – 1 | | | | |  |  | | | | |  | | | | |
|  |  | | | | |  | | | | |

*Рисунок 1. Формат командного слова протокола обмена*

Для проверки нормативности передаваемой информации каждый массив обрамляется дополнением до контрольной суммы (ДКС), которое вычисляется по следующей формуле:

ДКС = 0x5555 – КнтС,

где КнтС – пословная контрольная сумма входного массива без учета разряда  
 переполнения.

После обмена по каналу МКО системное бортовое программное обеспечение проверяет ДКС каждого передаваемого массива.

**2.3 Практическая часть**

Разработчику системного бортового программного обеспечения поступают исходные данные в виде таблицы, представляющей собой фрагмент данного протокола (таблица 3)

Таблица 3 – Исходные данные для программирования

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Система | Обмен | Номер ОУ | Формат | Под-адрес | Число слов данных | Запрос ФПО | Идентифика-тор слов данных | СЗО |
| ОУ1 | прием | 1 | 2 | 1 | 5 | zap1\_mko/0 | from\_ou1[4] | szoou12 |
| выдача | 1 | 2 | 10 | zap1\_mko/12-15 =9 | to\_ou1[9] | szoou11 |
| ОУ2 | прием | 2 | 2 | 3 | 20 | zap2\_mko/8,9=2 | from\_ou2[19] | szoou22 |
| выдача | 1 | 4 | 30 | zap2\_mko/14=1 | to\_ou2[29] | szoou21 |

В зависимости от значения параметров zap1\_mko, zap2\_mko необходимо сформировать структуру драйвера МКО (таблица 2). Выдать информацию может только контроллер.

Выделение разрядов, поиск номера ведущей грани происходит аналогично лабораторной работе №1.

**2.4 Порядок выполнения работы**

1. Разработать программный модуль, состоящий из двух подпрограмм:
2. Подпрограмма приема по МКО. Сформировать структуру для драйвера в соответствии с фрагментом протокола для приема по МКО. Проверить ДКС: если ДКС «не норма», то установить признак в 6-й разряд СЗО.
3. Программу выдачи по МКО. Дополнить массив на выдачу ДКС. Сформировать структуру для драйвера при условии, что грань является контроллером.
4. Составить технические требования (ТТ) на данный программный модуль (см. дополнение отдельным файлом), содержащий:
   1. Блок-схему алгоритма.
   2. Перечень входных и выходных параметров.
5. Составить контрольные примеры к программному модулю. Контрольные примеры должны охватывать все «ветки» программы.

В качестве **отчета** о проделанной работе необходимо предоставить: текст программного модуля на языке Си (исходный текст), ТТ и контрольные примеры к программному модулю.

**2.5 Пример формирования структуры для драйвера**

Определение типа структуры:

typedef struct

{

unsigned short KC;

unsigned short \*ptr\_mass;

unsigned short \*ptr\_SZO;

} srt;

str str\_drv[3];

Заполнение структуры на прием:

i = 0;

if(zap1\_mko&1)

{

str\_drv[0].KC=0x0c25;

str\_drv[0].ptr\_mas=from\_ou1\_drv;

str\_drv[0].ptr\_SZO=&szoou12;

i=1;

}

if(((zap2\_mko&0x0300)>>8)==2)

{

str\_drv[0].KC=0x1434;

str\_drv[0].ptr\_mas=from\_ou2\_drv;

str\_drv[0].ptr\_SZO=&szoou22;

i++;

}

str\_drv[i].KC=0;

Аналогично происходит заполнение структуры на выдачу.

Подсчет ДКС принятого массива:

dks=ks=0;

for (i=0; i<19; i++)

ks += from\_ou\_drv[i];

dks = 0x5555 — ks;

Установить 6-й разряд:

szoou22 |= 0x40;