

Справ. №	Перв. примен.
	КФДЛ.431299.029ТО

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

МИКРОСХЕМА ИНТЕГРАЛЬНАЯ
1867ВЦ9Т
ЗАГРУЗЧИК
Описание программы
КФДЛ.431299.029Д4

Содержание

1 Назначение загрузчика	3
2 Обзор программы	3
3. Подготовка файла с программой для загрузки	4
3.1 Подготовка файла для загрузки через интерфейс SCI	4
3.2 Загрузка программы через интерфейс SCI	7
3.3 Загрузка пользовательской программы из внешней памяти	8
4 Листинг программы	10
Лист регистрации изменений	17

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

1 Назначение загрузчика

Загрузчик микросхемы 1867ВЦ9Т предназначен для загрузки пользовательской программы из внешней памяти процессора или через последовательный порт SCI с последующей передачей управления этой программе.

2 Обзор программы

Загрузчик ИС 1867ВЦ9Т доступен, если только вывод МР/МС# = 0. При этом вывод ВЮ определяет, начнет ли загрузчик загружать пользовательскую программу или передаст управление программе по адресу 0x0200. Если ВЮ = 1, то загрузчик начинает загрузку пользовательской программы с последующей передачей ей управления. Опции определения источника загрузки, а также картирование внутренней памяти В0, представлены в таблице 1. Если ВЮ = 0, то осуществляется переход на адрес 0x0200.

Таблица 1 – Опции загрузки

XINT3/IO	XINT2/IO	Описание
0	0	ST1.CNF = 0 Загрузка пользовательской программы через SCI
0	1	ST1.CNF = 1 Загрузка пользовательской программы через SCI
1	0	ST1.CNF = 0 Загрузка пользовательской программы из внешней памяти пространства I/O с адреса 0x4000
1	1	ST1.CNF = 1 Загрузка пользовательской программы из внешней памяти пространства I/O с адреса 0x4000

На рисунке 1 приведена блок-схема загрузчика.

Инд. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

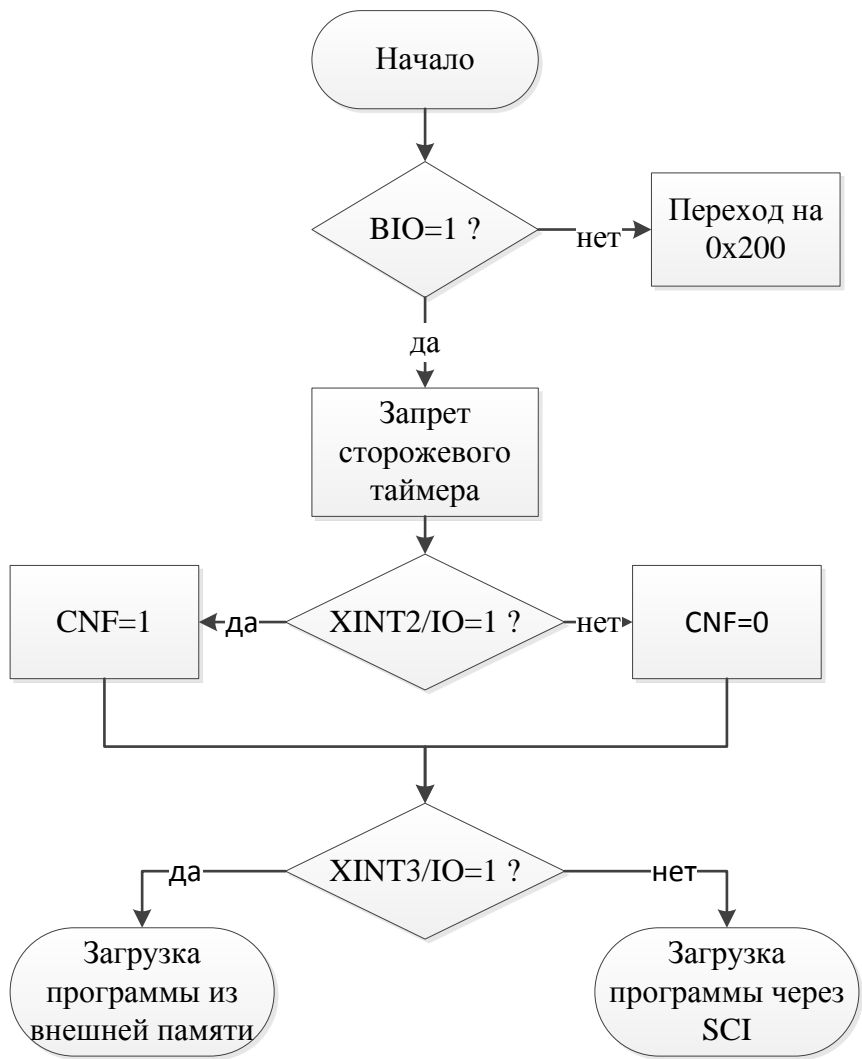


Рисунок 1 – Блок-схема загрузчика

Блок-схемы загрузки программы через интерфейс SCI и из внешней памяти приведены на рисунках 3 и 4 соответственно.

3. Подготовка файла с программой для загрузки

3.1 Подготовка файла для загрузки через интерфейс SCI

Перед загрузкой программы в ИС 1867ВЦ9Т через интерфейс SCI необходимо подготовить hex-код загрузочной таблицы пользовательской программы в специальном формате.

Загрузочная таблица должна содержать данные, приведенные в таблице 2.

Инд. № подл.	Подп. и дата	Взам. инв.№	Инв. № дубл.	Подп. и дата

Таблица 2 – Структура загрузочной таблицы

Номер слова	Описание обработки слова данных
1	Адрес точки входа в программу
2	Размер первой секции (количество 16-разрядных слов)
3	Стартовый адрес первой секции
4	Тип памяти (0 – программная память, 1 – память данных)
5	Данные первой секции
...	...
	Размер второй секции (количество 16-разрядных слов)
	Стартовый адрес первой секции
	Тип памяти (0 – программная память, 1 – память данных)
	Данные второй секции
...	...
	Размер последней секции (количество 16-разрядных слов)
	Стартовый адрес последней секции
	Тип памяти (0 – программная память, 1 – память данных)
	Данные последней секции
n	0000

Загрузочную таблицу можно получить с помощью утилиты F240_HEX.EXE, входящей в архив с программой SCILoader240.

Для запуска программы F240_HEX рекомендуется создать файл с расширением bat следующего содержания:

F240_HEX program_name.out

PAUSE

где program_name.out – имя исполняемой программы для ИС 1867ВЦ9Т.

При запуске bat файла на экране появится информация о преобразовании исполняемого файла в загрузочную таблицу – рисунок 2.

Инд. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

C:\WINDOWS\system32\cmd.exe

D:\tmp>F240_HEX op_abs.out
TMS320F240 COFF Version 1 Output-file BOOT-Loader Converter Version 1.00
(c) Copyright 1996 Texas Instruments
* Reading data from file : op_abs.out
* Reading entries for section: .vectors
* Reading entries for section: .text
* Reading entries for section: .dataB2
* Reading entries for section: .res
* Reading entries for section: .data
* Reading entries for section: .bss
* Reading entries for section: .vectors
*
* No of sections in file op_abs.out: 7
* No of converted sections: 3

OK, it's done

D:\tmp>PAUSE
Для продолжения нажмите любую клавишу . . .

```

Рисунок 2 – Создание загрузочной таблицы

При этом программа F240_HEX создает два файла – файл с загрузочной таблицей, он имеет расширение hex, а также файл с информацией о количестве обнаруженных секций, их размере, месте назначения, точке входа в программу и т. п. Этот файл имеет расширение inf.

Пример загрузочной таблицы:

```

8005 // Адрес точки входа в программу
0002 // Размер первой секции – два слова
8000 // Адрес начала первой секции
0000 // Секция располагается в программной памяти
1111 // Первое слово первой секции
2222 // Второе слово первой секции
0002 // Размер второй секции – два слова
0060 // Адрес начала второй секции
0001 // Секция располагается в памяти данных
3333 // Первое слово второй секции
4444 // Второе слово второй секции
0003 // Размер третьей секции – три слова
8020 // Адрес начала третьей секции
0000 // Секция располагается в программной памяти
1234 // Первое слово третьей секции
1234 // Второе слово третьей секции
1234 // Третье слово третьей секции
0000 // Окончание загрузочной таблицы

```

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

3.2 Загрузка программы через интерфейс SCI

Программа SCILoader240 использует файл с загрузочной таблицей для загрузки пользовательской программы с ПК в ИС 1867ВЦ9Т. Более подробная информация об SCILoader240 приведена в соответствующем руководстве оператора.

Получить файл с загрузочной таблицей можно также вручную, используя любой текстовый редактор, соблюдая структуру, указанную в таблице 2.

Для того чтобы загрузить программу в ИС 1867ВЦ9Т через интерфейс SCI, также можно использовать программное обеспечение, отличное от SCILoader240. При этом каждое слово из загрузочной таблицы следует отправлять побайтно – сначала старший байт, затем младший байт.

В загрузчике ИС 1867ВЦ9Т заданы величины $SCI_HBAUD = 0$, $SCI_LBAUD = 0x40$, что соответствует скорости передачи 19200 бит/с при частоте $SYSCLK - 10$ МГц. Формула расчета скорости передачи SCI приведена в техническом описании микросхемы 1867ВЦ9Т (КФДЛ.431299.029ТО раздел 13.4.5). В данном случае скорость передачи следует задать как $SYSCLK/520$. Остальные параметры COM порта должны быть выставлены следующим образом: 8 бит данных, отсутствие проверки на четность, один стоповый бит. Блок-схема загрузки программы через интерфейс SCI представлена на рисунке 3.

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

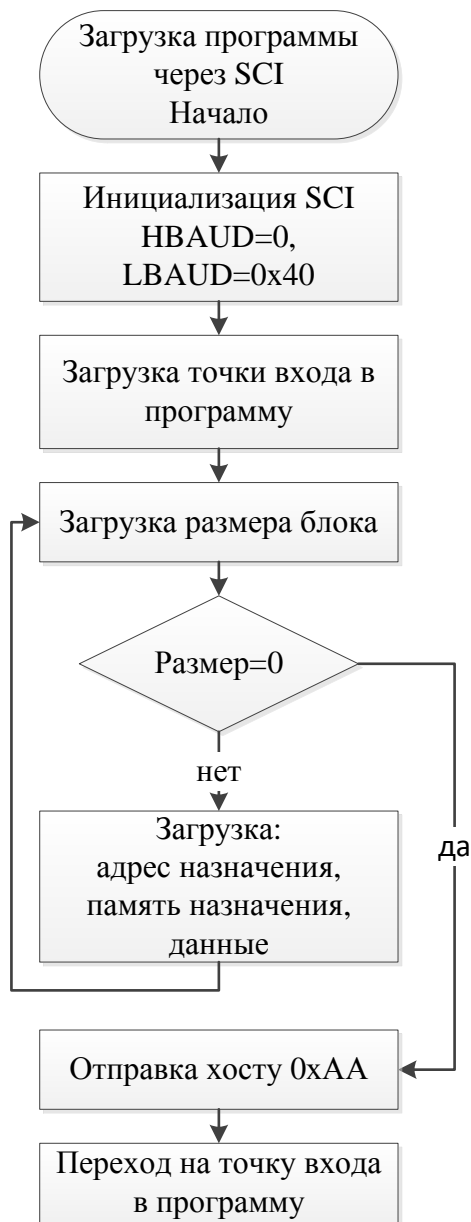


Рисунок 3 – Блок-схема загрузки программы через SCI

3.3 Загрузка пользовательской программы из внешней памяти

Если вывод ВІО = 1, а также вывод ХІNTЗ/ІО = 1, то осуществляется загрузка пользовательской программы из внешней памяти пространства І/О с адреса 0x4000.

По адресу 0x4000 должна располагаться разрядность внешней памяти – восемь, если память восьмиразрядная, и 0x10 – для шестнадцатиразрядной памяти. В восьмиразрядной памяти первым (по младшему адресу) должен быть расположен младший байт шестнадцатиразрядного слова, затем старший байт.

Ниже приведены примеры загрузочной таблицы для шестнадцатиразрядной и восьмиразрядной внешней памяти соответственно.

Инд. № подл.	Подп. и дата	Взам. инв.№	Инд. № дубл.	Подп. и дата

Пример загрузочной таблицы для 16-разрядной памяти

0010 // Разрядность памяти
 8005 // Адрес точки входа в программу
 0002 // Размер первой секции – два слова
 8000 // Адрес начала первой секции
 0000 // Секция располагается в программной памяти
 1111 // Первое слово первой секции
 2222 // Второе слово первой секции
 0002 // Размер второй секции – два слова
 0060 // Адрес начала второй секции
 0001 // Секция располагается в памяти данных
 3333 // Первое слово второй секции
 4444 // Второе слово второй секции
 0000 // Окончание загрузочной таблицы

Пример этой же загрузочной таблицы для 8-разрядной памяти

0008 // Разрядность памяти
 0005 // Адрес точки входа в программу (младший байт)
 0080 // Адрес точки входа в программу (старший байт)
 0002 // Размер первой секции – два слова (младший байт)
 0000 // Размер первой секции – два слова (старший байт)
 0000 // Адрес начала первой секции (младший байт)
 0080 // Адрес начала первой секции (старший байт)
 0000 // Секция располагается в программной памяти
 0000 // Секция располагается в программной памяти
 0011 // Первое слово первой секции (младший байт)
 0011 // Первое слово первой секции (старший байт)
 0022 // Второе слово первой секции (младший байт)
 0022 // Второе слово первой секции (старший байт)
 0002 // Размер второй секции – два слова (младший байт)
 0000 // Размер второй секции – два слова (старший байт)
 0060 // Адрес начала второй секции (младший байт)
 0000 // Адрес начала второй секции (старший байт)
 0001 // Секция располагается в памяти данных (младший байт)
 0000 // Секция располагается в памяти данных (старший байт)
 0033 // Первое слово второй секции (младший байт)
 0033 // Первое слово второй секции (старший байт)
 0044 // Второе слово второй секции (младший байт)
 0044 // Второе слово второй секции (старший байт)
 0000 // Окончание загрузочной таблицы (младший байт)
 0000 // Окончание загрузочной таблицы (старший байт)

При чтении данных из 8-разрядной внешней памяти содержимое старших разрядов [15...8] маскируется.

Инд. № подл.	Подп. и дата	Взам. инв. №	Инд. № дубл.	Подп. и дата

Блок-схема загрузки программы из внешней памяти пространства I/O представлена на рисунке 4.

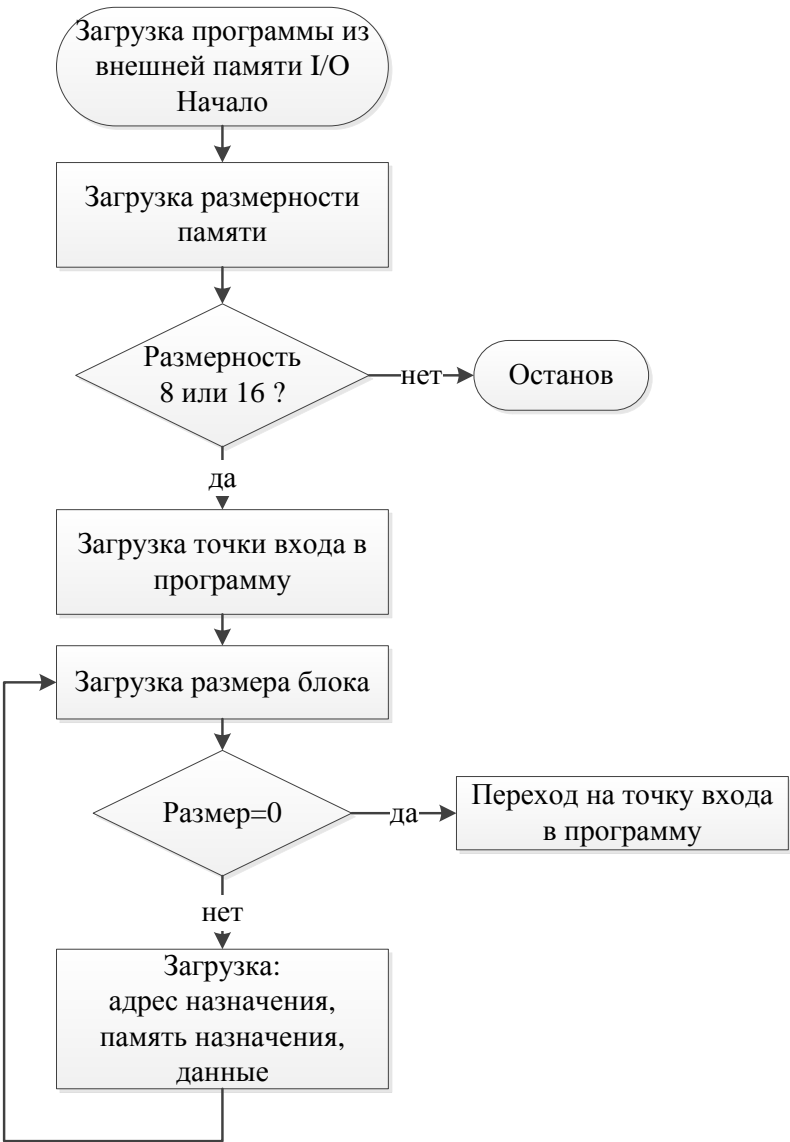


Рисунок 4 – Блок-схема загрузки из внешней памяти

4 Листинг программы

```
RSVECT      .include "C240APP.h"
             ; Вектор Reset, векторы прерываний
             .sect "vectors"
             B _c_int0

             .text
             .global _c_int0

_c_int0
```

Инв. № подл.	Подп. и дата	Взам. инв.№	Инв. № дубл.	Подп. и дата

```

; Проверка BIO, если BIO=0, то переход на 200h
bcnd 200h, BIO
;Disable watchdog (Vccp=5v), watchdog counter reset p6-12
LDP #00E0h
SPLK #0006Fh, WD_CNTL
SPLK #05555h, WD_KEY
SPLK #0AAAAh, WD_KEY

```

* Анализ XINTx/IO

```

BIT XINT2_CNTL, BIT6
bcnd nocnf, NTC
SETC CNF
nocnf BIT XINT3_CNTL, BIT6
bcnd SCILoad, NTC ; Если XINT3=0, загрузка через SCI

```

```

; запись модифицируемой IN
ldp #0h
splk #0AF7Fh, 60h ; запись кода in
splk #8000h, 61h ; модифицируемый PA
splk #7980h, 62h ; код b
splk next, 63h ; адрес перехода
splk #8B00h, 64h
splk #8B00h, 65h
splk #8B00h, 66h
lar AR0, #60h
mar *, AR0
lacc #11B0h ; адрес в программной памяти
rpt #6
tblw *+

```

```

splk #4000h, 7Eh ; Адрес начала ПЗУ в I/O!
; Загрузка данных из ПЗУ
call rio ; загружаем размерность ПЗУ
sub #8
bcnd mem8, EQ ; переход на загрузку из восьмого ПЗУ
sub #8
bcnd mem16, EQ ; переход на загрузку из 16-го ПЗУ
b end

```

* Загрузка из 16-го ПЗУ

```

mem16 call rio ; считываем точку входа в программу

```

Инд. № подл.	Подп. и дата	Взам. инв.№	Инв. № дубл.	Подп. и дата

```

load16  call rio
        bcnd end16, EQ ; в конец, если считали 0000
*****
* Копируем блок из 16-го ПЗУ в программную память
*****

        lar AR5, 7Fh ; размер блока
        mar *, AR5
        sbrk #1 ; коррекция размера блока для banz
        call rio ; адрес назначения
        sac1 7Dh ; сохраняем адрес назначения
        call rio ; память назначения
        bcnd l16pr, EQ ; если 0, то программная память
*** Загрузка в память данных ***
        lar AR4, 7Dh ; адрес назначения
        mar *, AR4
l16d  call rio
        sac1 *+, AR5 ; сохранение в память данных..
        banz l16d, AR4
        b load16 ; следующий блок..
*** Загрузка в программную память ***
l16pr  call rio
        lacl 7Dh ; адрес назначения
        tblw 7Fh ; копируем слово
        add #1 ; увеличиваем адрес назначения
        sac1 7Dh ; новый адрес назначения
        banz l16pr
*****
        b load16 ; следующий блок..
end16
        ; переход на адрес первого блока
        splk #4001h, 7Eh ; Точка входа в программу
        call rio
        bacc

*****
* Загрузка из восьмого ПЗУ
*****

mem8
        call getword ; считываем точку входа в программу
load8  call getword
        bcnd end8, EQ ; в конец, если 0000
*****
* копируем блок из восьмого ПЗУ в программную память
*****

        lar AR5, 7Fh ; размер блока

```

Инд. № подл.	Подп. и дата	Взам. инв. №	Инд. № дубл.	Подп. и дата

```

mar *, AR5
sbrk #1 ; корректируем размер блока для banz
call getword ; адрес назначения
sac1 7Dh ; сохраняем адрес назначения
call getword ; память назначения
bcnd l8p, EQ ; если 0, то программная память
*** Загрузка в память данных ***
lar AR4, 7Dh ; адрес назначения
mar *, AR4
l8d call getword
sac1 *, AR5 ; сохранение в память данных
banz l8d, AR4
b load8 ; следующий блок
*** Загрузка в программную память ***
l8p call getword
lacl 7Dh ; адрес назначения
tblw 7Fh ; копируем слово
add #1 ; увеличиваем адрес назначения
sac1 7Dh ; новый адрес назначения
banz l8p
*****

b load8 ; следующий блок..
; переход к точке входа
end8 splk #4001h, 7Eh
call getword ;
bacc
*****

* Загрузка данных через SCI
*****
SCILoad
; Инициализация SCI
SPLK #0017h, SCI_CCNTL ; 1 stop bit,no parity,8 char bits,
; async mode, idle-line protocol
SPLK #0013h, SCI_CNTL1 ; Enable TX, RX, internal SCICLK
SPLK #0000h, SCI_HBAUD
SPLK #0040h, SCI_LBAUD ; Baud Rate=19200 b/s 10 MHz
SYSCLK)
SPLK #0022h, SCI_PORT_C2 ; Enable TXD & RXD pins
SPLK #0033h, SCI_CNTL1 ; Relinquish SCI from Reset.

call getSCIword ; считываем точку входа в программу
sac1 7Dh
loadSCI call getSCIword
bcnd endSCI, EQ ; в конец, если 0000
*****

```

Инд. № подл.	Подп. и дата	Взам. инв.№	Инв. № дубл.	Подп. и дата

* Загружаем блок из SCI в память

```
sac1 7Fh
lar AR5, 7Fh ; размер блока
mar *, AR5
sbrk #1 ; корректируем размер для banz
call getSCIword ; адрес назначения
sac1 7Eh ; сохранение адреса назначения
call getSCIword ; считываем память назначения
bcnd lSCIpr1, EQ ; если 0, то программная память
```

*** Загрузка в память данных ***

```
lar AR4, 7Eh ; адрес назначения
mar *, AR4
lSCIId call getSCIword
sac1 *+, AR5
banz lSCIId, AR4
b loadSCI
```

*** Загрузка в программную память ***

```
lSCIpr1 lac1 7Eh ; загрузка адреса назначения
lSCIpr sac1 7Eh ; сохранение адреса назначения
call getSCIword
sac1 7Fh
lac1 7Eh ; адреса назначения
tblw 7Fh
add #1 ; корректируем адрес назначения
banz lSCIpr
```

```
b loadSCI
endSCI
; Отправка хосту AA - успешная загрузка
lacc #0AAh
ldp #0E0h
sac1 SCI_TX_BUF
; переход на адрес первого блока
ldp #0
lac1 7Dh
bacc
nop
nop
nop
*****
end b $
nop
nop
```

Инд. № подл.	Подп. и дата	Взам. инв.№	Инв. № дубл.	Подп. и дата

nop

* Считывает слово из SCI в аккумулятор
* (считывается из ПЗУ 16 разрядов с мусором в старших)

getSCIword

```
LDP #00E0h
WH      BIT  SCI_RX_STAT, BIT6 ; Test RXRDY bit
        BCND      WH, NTC
        LACC      SCI_RX_BUF, 8 ; Первый байт - старший
WL      BIT  SCI_RX_STAT, BIT6 ; Test RXRDY bit
        BCND      WL, NTC
        OR        SCI_RX_BUF ; Второй байт - младший
        LDP #0000h
        ret
        nop
        nop
        nop
```

* Считывает слово из восьмого ПЗУ в аккумулятор
* (считывается из ПЗУ 16 разрядов с мусором в старших)
* 7F содержит прочитанное слово (АСС также его содержит)

getword

```
call rio
and #0FFh
sac1 7Ch ; получили младшие разряды
call rio
and #0FFh
sac1 7Fh
lacc 7Fh, 8 ; получили старшие разряды
or 7Ch
sac1 7Fh
ret
nop
nop
nop
```

* Считывает слово из I/O в АСС
* 7E содержит адрес чтения из I/O
* 7F содержит прочитанное слово из I/O

Инд. № подл.	Подп. и дата	Взам. инв.№	Инв. № дубл.	Подп. и дата

rio

```
; 7E - адрес чтения I/O
lacc #11B1h ; адрес IN в ACC
tblw 7Eh ; корректируем адрес для I/O
      b 11B0h
nop
nop
      nop
```

next

```
lacc 7Eh ; инкрементирование адреса
add #1
sacc 7Eh
lacc 7Fh ; загружаем прочитанное слово в ACC
ret
nop
nop
nop
.end
```

Инв. № подл.	Подп. и дата	Взам. инв.№	Инв. № дубл.	Подп. и дата

Лист регистрации изменений

[illegible]

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата