

ReMem Programming Guide

May 6, 2006
Steve Adolph
V4.1

This document applies to ReMem systems that are running on release 4 firmware. Please use document V3.1 for release 3 systems, and V2.1 for pre release 3 systems.

Change record

- V3.0 - Dec 18/05 : first version for release 3 firmware
- V3.1 - Jan 24/06 : correction to MMU byte description.
- V4.0 - Apr 7/06 : added changes for T200 support, and the ReMem UART port
- V4.1 - May 6/06 : minor corrections, (more Jan '08)

1. Overview

ReMem II or ReMem2V2 (referred to in this document as ReMem) is a memory subsystem designed for the Model 100, Tandy 102 and Tandy 200 laptop. It consists of

- 2MB of static ram (referred to as ram)
- two instances of 2MB Flash memory (referred to as flash 1 and flash 2)
- a CPLD which performs the functions necessary to control, manage and adapt these memory resources to the 64KB memory space of the host computer
- a JTAG programming interface (the use of this interface will be covered in another document)

Besides providing enhanced memory functions, ReMem also provides an internal UART port for serial communication with future enhancements. ReMem exists in two hardware versions. The early version (ReMem II) was based on Xilinx XC2C256 CPLD, while the later (current) version (ReMem2V2) is based on Xilinx XCR3256 CPLD. This document describes how to program and use ReMem.

2. ReMem Registers

2.1. ReMem Firmware Revision Register

The firmware revision register is at IO port 71H or 113 decimal. At any time during normal operation, the Firmware Revision Register may be read by performing the following operation in M100 basic.

?INP(113)

This register is read only. The resulting number will indicate the firmware revision that is currently installed in ReMem.

Valid firmware release codes:

ReMemII

Release:	M100	T102
1	33	34
2	35	36
3	37	38
4	40	41

ReMem2V2

Release:	M100	T102	T200
1	X	X	X
2	X	X	X
3	X	X	X
4	64	65	66

2.2.Status and Control Register

To understand the state that ReMem is operating in, the CPLD provides a status and control register. This register is accessed for both reads and writes through IO port 70h (112 dec). This register will be referred to as the SCR.

The format of the Status and Control Register is as follows

Bit#	usage	readable	writable	default value
MSB	flash 1 status	yes	no	N/A
6	flash 2 status	yes	no	N/A
5	MMU map #2	yes	yes	0
4	MMU map #1	yes	yes	0
3	MMU map #0	yes	yes	0
2	MMU MAP IO ENABLE	yes	0	
1	RAMPAC ENABLE	yes	yes	0
LSB	ReMem MODE	yes	yes	0

A system reset (specifically, a state change from 0 to 1 on RAMRST signal in the M100) causes a return to default values for the status and control register. Soft resets, hard resets and power cycling all cause ReMem to revert to default status and control register contents.

Programming values into the status and control register requires the basic command

OUT 112,d

where d represents the byte of data you wish to place in the register. Current register contents can be polled via the basic command

?inp(112)

2.2.1.Flash status

The flash chips provide a signal that indicates it's readiness to accept new commands or memory accesses. The value of '1' indicates that the flash chip is ready, while a '0' indicates that the flash chip is busy. The flash chips cannot be accessed or programmed when busy. Please read the flash chip data sheet for more information.

2.2.2.ReMem Mode

ReMem provides memory access to the host computer in two ways :

- direct access, where the address lines from the CPU are directly mapped to the ram and flash chips without modification,
- and a memory mapped mode where ReMem maps, depending on a provisioned memory mapping table, the full 6MB of memory into the 64KB space for the 8085.

These two modes are referred to as normal and MMU mode. The LSB of the status and control register indicates which mode ReMem is currently operating in. A '1' indicates MMU mode and a '0' indicates normal mode.

The memory map must be correctly provisioned before the user attempts operation in MMU mode!

2.2.3.Rampac Enable

ReMem has built in support for Rampac, which was a 256KB add-on storage device for the M100. When bit 1 is set, ReMem allows access to the rampac memory using the Rampac I/O method, utilizing IO ports 81H (129 dec) and 83H (131 dec). When bit 1 is '0', all rampac functions are disabled.

2.2.4.MMU MAP I/O ENABLE

ReMem utilizes the built in rampac I/O method for two separate functions: the first and primary function is to provide access to the MMU map, and the second function is to emulate a rampac. When bit 2 is set, ReMem allows access to the rampac memory using the Rampac I/O method, utilizing IO ports 61H (97 dec) and 63H (99 dec). When bit 1 is '0', all MMU MAP I/O functions are disabled.

Note, the port numbers are different - the sector port is 97, while the data port is 99. Having a separate IO port for the MMU MAP allows rampac usage as well as MMU MAP access, and also prevents rampac programs from messing up the MMU map.

2.3.UART Port Status and Control Register

ReMem includes a UART port. The UART is controlled via IO to port 114d.

The format of the UART Port Status and Control Register is as follows:

Bit#	usage	readable	writable	default value
MSB	rx_buf_full	yes	no	0
6	tx_buf_empty	yes	no	0
5	not used	N/A	N/A	0
4	not used	N/A	N/A	0
3	not used	N/A	N/A	0
2	serial_port_en	yes	yes	0
1	clk_sel1	yes	yes	0
LSB	clk_sel0	yes	yes	0

Soft resets, hard resets and power cycling all cause ReMem to revert to default UART Port Status and Control register.

Programming values into the status and control register requires the basic command

```
OUT 114,d
```

where d represents the byte of data you wish to place in the register. Current register contents can be polled via the basic command

```
?inp(114)
```

2.3.1.rx_buf_full

This bit indicates that the rx buffer is either empty or full. A 1 indicates that there is a data byte in the buffer. The UART receiver will not accept another data byte until the RX buffer is read. Reading the buffer clears the buffer.

2.3.2.Tx_buf_empty

A zero in this bit indicates that the TX buffer is empty, and can accept a byte to transmit. If this buffer is not empty, this bit is set to 1, and the UART is attempting to transmit the byte.

2.3.3.Serial_port_en

Setting this bit enables the uart port. Resetting this bit disables the uart port.

2.3.4.Clk_sel1, Clk_Sel0

These 2 bit select a UART rate. Rates are as follows.

Clk_sel1	clk_sel0	Rate (Baud)
0	0	19200
0	1	38400
1	0	76800
1	1	153600

2.4.UART Port Data Register

The UART Port Data Register is accessed via port 115. Reads to the UART Port Data Register reads data from the Rx buffer. Writes to the UART Port Data Register writes to the Tx buffer.

3.ReMem Hardware Configuration

Default operating mode:

- all jumpers removed
- default flash for booting is flash 1
- ReMem responds to memory activity in all memory regions.

ReMemII may be configured externally by the setting of jumpers. These jumpers control how ReMem memory access is accomplished.

ReMem2V2 may be configured externally by the setting of jumpers, and by soldering a blob of solder across two closely spaced test points. These settings control how ReMem2V2 memory access is accomplished.

M100/T102

Jumper Name:	Nickname:	Function:
CFG1	boot	select boot flash (1 or 2)
CFG2	no_ram	allows ReMem to respond to 8000-FFFF
CFG3	no_rom	allows ReMem to respond to 0000-7FFF
CFG4	not used	not used

T200

Jumper Name:	Nickname:	Function:
CFG1	boot	select boot flash (1 or 2)
CFG2	no_ram	allows ReMem to respond to A000-FFFF
CFG3	no_rom	allows ReMem to respond to 0000-9FFF
CFG4	not used	not used

4.ReMem Operation

4.1.Normal Mode

When ReMem is operating in normal mode, the host CPU's address and data busses are directly connected to the memory resources on ReMem in the following manner:

4.1.1.M100/T102

Address range:

Standard ROM Access (STROM = 1) 0000H-7FFFH:

In this address range, ReMem assigns the 32kB of flash memory according to the setting of the boot jumper. When the boot jumper is not installed, ReMem assigns memory range 000000H to 007FFFH from flash 1 to CPU range 0000H-7FFFH. When boot jumper is installed, then ReMem selects the same memory range from flash 2. If no_rom jumper is installed, then ReMem does not respond to memory requests in this range.

Option ROM Access (STROM = 0) 0000H-7FFFH:

In this address range, ReMem assigns the 32kB of flash memory according to the setting of the boot jumper. When the boot jumper is not installed, ReMem assigns memory range 008000H to 00FFFFH from flash 1 to CPU range 0000H-7FFFH. When boot jumper is set, then ReMem selects the same memory range from flash 2. If no_rom jumper is installed, then ReMem does not respond to memory requests in this range.

RAM Access 8000H-FFFFH:

In this address range, ReMem assigns memory range 000000H to 007FFFH from the ram chip to CPU range 8000H-FFFFH. If no_ram jumper is installed, then ReMem does not respond to memory requests in this range.

User is of course free to put whatever code one wants, but clearly this is designed for legacy support for M100 OS as is, with an option rom programmed into the flash. An example can be the M100 OS in flash 1 000000H-007FFFH, and TSDOS in flash 1 008000H-00FFFFH.

Issuing the command

```
OUT 112,0
```

places ReMem in normal mode. (in fact, any out command to port 112 that has the LSB set to 0 will set ReMem into normal mode.)

4.1.2.T200

Address range:

Main OS ROM access 0000H-9FFFH, bank1=0:

In this address range, ReMem assigns the 40kB of flash memory according to the setting of the boot jumper. When the boot jumper is not installed, ReMem assigns memory range 000000H to 009FFFH from flash 1 to CPU range 0000H-9FFFH. When boot jumper is installed, then ReMem selects the same memory range from flash 2. If no_rom jumper is installed, then ReMem does not respond to memory requests in this range.

Multiplan ROM Access 0000H-9FFFH, bank2=0:

In this address range, ReMem assigns the 32kB of flash memory according to the setting of the boot jumper. When the boot jumper is not

installed, ReMem assigns memory range 010000H to 017FFFH from flash 1 to CPU range 0000H-7FFFH. When boot jumper is set, then ReMem selects the same memory range from flash 2. If no_rom jumper is installed, then ReMem does not respond to memory requests in this range. (range 8000-9FFF is not assigned)

Option ROM Access 0000H-9FFFH, bank3=0:

In this address range, ReMem assigns the 32kB of flash memory according to the setting of the boot jumper. When the boot jumper is not installed, ReMem assigns memory range 018000H to 01FFFFH from flash 1 to CPU range 0000H-7FFFH. When boot jumper is set, then ReMem selects the same memory range from flash 2. If no_rom jumper is installed, then ReMem does not respond to memory requests in this range. (range 8000-9FFF is not assigned)

RAM bank 1 Access A000H-FFFFH, bank4=0:

In this address range, ReMem assigns memory range 002000H to 007FFFH from the ram chip to CPU range A000H-FFFFH. If no_ram jumper is installed, then ReMem does not respond to memory requests in this range.

RAM bank 2 Access A000H-FFFFH, bank5=0:

In this address range, ReMem assigns memory range 00A000H to 00FFFFH from the ram chip to CPU range A000H-FFFFH. If no_ram jumper is installed, then ReMem does not respond to memory requests in this range.

RAM bank 3 Access A000H-FFFFH, bank6=0:

In this address range, ReMem assigns memory range 012000H to 017FFFH from the ram chip to CPU range A000H-FFFFH. If no_ram jumper is installed, then ReMem does not respond to memory requests in this range.

User is of course free to put whatever code one wants, but clearly this is designed for legacy support for laptop OS as is, with an option rom programmed into the flash. An example can be the M100 OS in flash 1 000000H-007FFFH, and TSDOS in flash 1 008000H-00FFFFH.

4.2.MMU mode

MMU mode allows full and flexible memory mapping on a 1 kb granularity. In MMU mode, ReMem relies on only one of the 8 maps to define the memory mapping function. The user may control which map is selected through the SCR bits 3,4 and 5. Instant (within 1 clock cycle) changes between maps are possible through out 112,x commands where bits 3,4,5 are changed to represent the desired map. It is not required to revert to normal mode to change the MMU mappings.

The structure of the overall MMU map is as follows:

- there are 8 separate MMU maps
- these maps are written sequentially in the 2MB static ram chip

- each map is 512 bytes long, so the set of 8 MMU maps occupies a 4KB range of ram locations
- map 000 is located from 1BF000H to 1BF1FFH
- map 001 is located from 1BF200H to 1BF3FFH
- map 010 is located from 1BF400H to 1BF5FFH
- map 011 is located from 1BF600H to 1BF7FFH
- map 100 is located from 1BF800H to 1BF9FFH
- map 101 is located from 1BFA00H to 1BFBFFH
- map 110 is located from 1BFC00H to 1BFDFFH
- map 111 is located from 1BFE00H to 1BFFFFH

The amount of MMU map required to define the mapping for a single 1K memory block, is 2 bytes. Hence, a 32KB space requires 64 bytes. The structure of the MMU mapping byte pair is as follows:

bit	use	description
MSB	not used	future application?
14	sector lock	setting this bit prevents writes to the block
13	ram CS	ram chip select, 0 means ram selected
12	flash 2 CS	flash #2 chip select, 0 means flash 2 selected
11	flash 1 CS	flash #1 chip select, 0 means flash 1 selected
10	map data	
9	map data	
8	map data	
7	map data	
6	map data	
5	map data	bits 0-10 hold the address pointer
4	map data	for the 2MB address space, there are 21 address bits
3	map data	CPU address bits 0 to 9 get appended to map data bits
2	map data	0 to 10, to create the actual 21 bit address used on
1	map data	ReMem.
LSB	map data	

NOTE: there is no hardware mechanism to prevent the user from selecting two or more memory devices at the same time. Please do not do this. To clarify if bits 11 12 13 were assigned 001 for example, that would be bad.

Valid values for bits 11-13 are 110, 101, 011 only!!!!!!!!!!!!
 Invalid values for bits 11-13 are 000, 001, 010, 100.

While not invalid, values of 111 in bits 11-13 are useless. Creates a black hole in the memory space.

Accessing the MMU map can be done in 2 different ways:

- 1) through the rampac IO method
- 2) by provisioning the MMU Map to allow memory 1BF000h-1BFFFFh to be present within the 96k address space of the CPU.

4.2.1.MMU map - to - rampac sector assignment:

There are 8 maps, with 4 separate regions in each map. In MMU mode, the rampac function supports 32 sectors of 128 bytes each. They are assigned as follows:

- Map 0 occupies sectors 0-3
- Map 1 occupies sectors 4-7
- Map 2 occupies sectors 8-11
- Map 3 occupies sectors 12-15
- Map 4 occupies sectors 16-19
- Map 5 occupies sectors 20-23
- Map 6 occupies sectors 24-27
- Map 7 occupies sectors 28-31.

Sectors 3, 7, 11, 15, 19, 23, 27 and 31 are unused - each map only requires 3 sectors. These spaces can be used for other things if required (1024 bytes total).

4.2.2.MMU access method #1

Method one is applicable in both MMU mode and normal mode.

- issue the following basic command: OUT 112,4: this sets bit 2 in the status and control register, enabling access to the MMU map
- issue basic command OUT 97, mm where mm represents the map "record" one wishes to update.
- issue the basic command OUT 99, nn where nn represents the mapping bytes * up to 64 (M100/T102) or 128 (T200) sequential OUT 99, nn commands are required to fully provision the one "sector" of the MMU map
- a good understanding of how RAMPAC works is useful knowledge here.

So, if one wanted to change the MMU map for map#3, ram region, 27th 1k block, the following sequence would work (M100/T102 only):

```
--  
OUT 112,4  -- turn on rampac MMU access  
OUT 97,9   -- select record #9 -- map 3 ram region  
FORI=1to52:?INP(99):NEXTI    -- advance 26 byte pairs  
OUT 99,X1  
OUT 99,X2  -- assign X1X2 to byte locations 53 and 54.  
---
```

4.2.3.MMU access method #2:

The user must be operating in MMU mode for this method to work. Using method #1, the memory space that holds the MMU map may be placed into the accessible memory space of the host computer. Once this has been accomplished, using memory peek/pokes to the correct locations will cause an immediate update to the memory mapping functions.

BE CAREFUL WHEN CHANGING THE MMU MAP WHEN IN MMU MODE!!!!!! INCORRECT MEMORY MANAGEMENT CAN BE FATAL TO APPLICATIONS!!!!!!

To use ReMem in MMU mode, the memory map must be initialized. Issuing the basic command out 112,1 sets ReMem into MMU mode.

4.2.4.MMU mode in M100/T102

MMU mode of operation allows the 96KB space (64KB of strom = 1, and 32kb of strom = 0) to be configured on a 1k block basis. Any 1K block of memory from the entire 6MB of memory space, can be assigned to any 1K block in the 96kB CPU memory space.

For each MMU map, the structure is as follows:

- each map is divided into 4 regions
- 000-03F - 64 bytes mapping 0000H-7FFFh STROM=0
- 040-07F - 64 bytes not used

- 080-0BF - 64 bytes mapping 8000h-FFFFh
- 0C0-0FF - 64 bytes not used

- 100-13F - 64 bytes mapping 0000H-7FFFh STROM=1
- 140-17F - 64 bytes not used

- 180-1FF - 128 bytes not used

These bytes must be written to the map as follows:

000-03F	080-0BF	010-13F
byte 0 byte 1	byte 0 byte 1	byte 0 byte 1
LSB....MSB	LSB....MSB	LSB....MSB

4.2.5.MMU mode in T200

MMU mode of operation allows the 192KB space to be configured on a 1k block basis. Any 1K block of memory from the entire 6MB of memory space, can be assigned to any 1K block in the 192kB CPU memory space.

For each MMU map, the structure is as follows:

- each map is divided into 4 regions
- 000-04F - 80 bytes mapping 0000H-9FFFH bank1=0
- 050-07F - 40 bytes mapping A000H-FFFFH bank4=0

- 080-0CF - 80 bytes mapping 0000H-9FFFH bank2=0
- 0D0-0FF - 40 bytes mapping A000H-FFFFH bank5=0

- 100-14F - 80 bytes mapping 0000H-9FFFH bank3=0
- 150-17F - 40 bytes mapping A000H-FFFFH bank6=0

- 180-1FF - 128 bytes not used

These bytes must be written to the map as follows:

0000-007F
byte 0 byte 1
LSB....MSB

080-0FF
byte 0 byte 1
LSB....MSB

100-17F
byte 0 byte 1
LSB....MSB