App note: ARM code in sideways ROMs

Introduction

This note describes how to formulate a sideways ROM which resides in the 6502 based host microcomputer and which contains ARM code intended for the coprocessor. This can then be set to copy the code into the ARM coprocessor at startup and run a language other than BASIC (the default).

Conventions used in this document

The following typographical conventions are used throughout this guide:

Hexadecimal numbers are prefixed with ampersand.

Decimal numbers have no prefix.

Binary numbers may be denoted with a leading percent and given in decending bit significant order (ie.for an eight bit number they will be written in the order %76543210).

Multibyte data is stored in memory in little endian form.

Copyright

History

V0.10 First draft

V0.11 Changed example code to have the first two ARM instructions with the same condition code

V0.12 Renamed

Behaviour

Default general coprocessor case

At startup the MOS sends round service call &FE if a coprocessor is detected, this is used by the coprocessor as an opportunity to output its replacement startup banner, and it is used by the host to explode the soft fonts (if appropriate for that machine) before the other workspace is allocated.

Following this the MOS sends round service call &FF to finish off any host related activities for the coprocessor to work such as claiming the EventV, BRKV, and copying the Tube code into what would normally be the language workspace in the host (zero page &00-&7F and &400-&700).

The remainder of the startup sequence proceeds as usual, filing systems are started, and the rest of the host's workspace is allocated.

Lastly, the default language is selected (for the Master this is as configured in CMOS memory, otherwise it is the ROM in the highest socket number). The language is copied across to the coprocessor taking into account the relocation address, if given, and the language is entered at the relocation address.

Default ARM coprocessor case

This is the same as the general case, except that a special check is made for 6502 BASIC being copied into the coprocessor as this cannot be run natively in ARM mode.

If the language copied across is seen to be any of BASIC I to BASIC IV then the "ARM Tube OS" will start ARM BASIC V instead which is held in the flash memory on the coprocessor already. It assumes at this point that the MOS has already printed the BASIC banner so backs up the cursor and prints the correct banner in its place: as far as the MOS is concerned BASIC started normally.

Languages other than BASIC which are not marked as ARM code as detailed below will be faulted and the coprocessor will default to a supervisor prompt, ready to accept commands

```
I cannot run this code
*
```

this allows software to be loaded from disc for example.

ARM coprocessor case with a sideways ROM containing ARM code

If the language copied across after service call &FF is marked as ARM code, the "ARM Tube OS" will inspect the 6502 JMP instruction at the relocation address given in the sideways ROM (or &8000 if no relocation is flagged) and branch to the language entry point as though the psuedo command

```
*GO ?&8001+(?&8002*256)
```

had been entered.

The address given after the 6502 JMP instruction must be a multiple of 4 as ARM code must be word aligned. Additionally, the relocation address given in the sideways ROM header is effectively restricted to the range &8000-&FFFC by this requirement as

- the application space starts at &8000 in the coprocessor
- there are only 2 bytes after the JMP instruction

however this is unlikely to cause a serious restriction as the size of language ROM held in a sideways ROM is limited to 16kbytes (or possibly slightly more by compressing the ARM code and having it decompress itself).

The bottom byte of the relocation address (if present) should be zero to ensure the whole sideways ROM header is copied into the coprocessor, as some fields are inspected and checked by the "ARM Tube OS".

Configuring the default language

Master series microcomputer

The default language selected at reset is stored in battery backed CMOS memory, using the command

```
*CONFIGURE LANG <rom id>
```

and the current setting can be displayed with

```
*STATUS LANG
```

normally this will be the BASIC language, whose rom id can be found using the command

```
*ROMS
```

It is taken into account on the next hard reset.

Model B and B+ microcomputers

The default language selected at reset is determined by the first language found, so the language in the highest ROM socket will be used.

To avoid having to remove the lid repeatedly when switching the ARM coprocessor on or off, some careful use of software on the language entry point (see example below) forces the MOS to use the next language down when started as the processor type is not checked as rigourously as on the Master series.

Example listing

The following listing includes an application which prints all of the letters of the alphabet then quits, it is written in the assembly syntax expected by BBC BASIC V. Alternatively it could be translated into an alternate assembly format to allow it to be mixed with code written in 'C' for example.

```
REM Application written in ARM code residing in the host
REM (C)2005 SPROW
:
DIM image% 512
osby = &FFF4 : oswc = &FFEE
:
FOR X = 4 TO 6 STEP 2
P% = &8000 : O% = image%
[OPT X
OPT     FNjmp(language)
OPT     FNjmp(service)
EQUB&CD               \Denote language and service entries, and ARM code
EQUBcopyright MOD256  \Offset to copyright string
EQUB&01               \Version number divided by 10
:
.title
EQUS"ALPHABET"        \ROM title and hence language name too
EQUB&00
.version
EQUS"0.11"            \Version string
EQUB&00
:
.copyright
EQUB&00
EQUS"(C)2005 SPROW"   \Copyright string
EQUB&00
ALIGN
:
.language
\The language entry point is entered in ARM mode, and possibly 6502 too
EQUW    &01C9         \6502="CMP#1";       ARM=harmless "MVNNES R0, R9, ASR#3"
EQUW    &11F0         \6502="BEQ langdown"
EQUW    &0060         \6502="RTS";         ARM=harmless "ANDNE R0, R0, R0, RRX"
EQUW    &1000         \6502=not executed
B       main_armcode  \6502=not executed;  ARM=branch to the real start
:
.langerr
\A handy 9 byte gap to fill
EQUS    "?egaugnaL"
.langdown
\Someone's tried to enter the ROM on a 6502 processor, try the next ROM down
EQUW    &0BA9
OPT     FNjsr(oswc)
OPT     FNjsr(oswc)
EQUW    &00A0
OPT     FNldaay(title)
EQUW    &08F0
EQUW    &20A9
OPT     FNjsr(oswc)
```

```
EQUB    &C8
EQUW    &F310
EQUW    &0DA9
OPT     FNjsr(oswc)
EQUW    &FFA0
EQUW    &00A2
EQUW    &AAA9
OPT     FNjsr(osby)
EQUW    &F286
EQUW    &F384
EQUW    &F4A4
EQUB    &88
EQUW    &F2B1
EQUW    &4029
EQUW    &07F0
EQUB    &98
EQUB    &AA
EQUW    &8EA9
OPT     FNjmp(osby)
EQUB    &88
EQUW    &F010
EQUW    &08A0
OPT     FNldaay(langerr)
OPT     FNjsr(oswc)
EQUB    &88
EQUW    &F710
EQUW    &FE30
:
.service
\The service call handler is always running on the 6502 host, does nothing
EQUB    &60
ALIGN
:
.main_armcode
\Print the alphabet then return
MOV     R0, #ASC"A"
.main_loop
SWI     "XOS_WriteC"
BVS     main_quit
ADD     R0, R0, #1
CMP     R0, #ASC"Z"
BLS     main_loop
SWI     "XOS_NewLine"
.main_quit
SWI     "OS_Exit"
]
NEXT
OSCLI("SAVE OUTPUT " + STR$~(image%) + " " + STR$~(O%) + " 8000 FFFBBC00")
END
:
DEF FNjsr(addr%):[OPT X:EQUB&20:EQUWaddr%:]:=X
DEF FNldaay(addr%):[OPT X:EQUB&B9:EQUWaddr%:]:=X
DEF FNjmp(addr%):[OPT X:EQUB&4C:EQUWaddr%:]:=X
```