DRAGONPLUS ELECTRONICS

Audio/Video Peripheral Device

SuperSprite FM+ ™ (6309/6809)

FOR DRAGON & TANDY COLO[U]R COMPUTERS

User Guide

Revision 1.0 – 02 September 2021

Contents

TABLE OF CONTENTS

Version History

Vsn	Date	Author	Comment(s)
0.1	17 Aug 2021	John Whitworth	Initial public release
0.2	18 Aug 2021	John Whitworth	Minor updates. Add versioning and tidy up footers etc.
0.3	24 Aug 2021	John Whitworth	Addition of index.
0.4	28 Aug 2021	John Whitworth	Additional text about capabilities of stock Dragon (and possibly Coco) power supplies – thanks Richard H! Additional text about stand-off pegs added to Quick-ish Start User Guide. Appendix C added to summarize power requirements.
0.5	28 Aug 2021	John Whitworth	There are dark forces at work. Somehow the paper size had been set as 'Letter' and the language as 'U.S. English'. This has been rectified.
0.6	30 Aug 2021	John Whitworth	Added sample YM2149 & YM2413 BASIC programs. Added sample V9958 assembler program.

1 INTRODUCTION

1.1 WELCOME!

Thank you for buying this SuperSprite FM+ ™ graphics & sound card!

This card delivers exciting new features to your Dragon or Tandy CoCo, bringing its audio and visual capabilities up to 8-10 years[1] further into the future compared to the original launch of the Dragon or CoCo 1 & 2 machines.

The card is designed to be as flexible as possible with respect to installation and usage.

Before going into too much other detail, it's worth discussing power requirements for this board.

1.1.1 SuperSprite FM+ ™ Power

There are numerous chips on board the SuperSprite FM+ ™ board – some hungry for power, others not so. Don't worry – you won't need to wire up a 1KW ATX power supply – but it's likely you will require a little external power top-up.

If you're running a Dragon 32, 64, 200 or Tano, then it is likely your existing factory internal power supply will either not be able to cope or will be put under considerable strain by this expansion board. I've seen some D64 PSUs cope, and some not. So best to assume they will not. I am no CoCo expert, but I would envisage that the stock Coco PSUs would also struggle to supply the additional power required.

If you have a modern uprated PSU, then you will likely have no issues. One such example is Phill Harvey-Smith's PSU board.

If you have an old internal power supply, then please strongly consider making use of the SuperSprite FM+ ™'s external power option. Please see sections 1.6.2.1, 1.6.4.1 and 2.1.4.

1.2 PACKAGE CONTENTS

Within the box you should have received:

- The SuperSprite FM+ ™ card (built or kit form)
- Four nylon standoff pegs for mounting.

1.3 PRE-REQUISITES

You will require (not supplied):

- Full 21-pin to 21-pin SCART cable.

[1] The Coco 1 was released in 1980. The MC6847 was released in 1978. The V9958 graphics chip was released in 1988.

1.4 Optional Accessories

- Regulated PSU that can supply 5V @1A, centre-positive 2.1/2.5 mm barrel-jack.
- 2-pin audio connector cable (to connect machine's existing sound).
- 10-pin RGB/composite connector cable (to connect machine's existing video).
- A cartridge port extender if you plan to use a disk solution (e.g. MPI or Oojamaflip).

1.5 Special Features

1.5.1 V9958

The V9958 is an LSI video display processor (VDP). It uses an N-channel silicon gate MOS and is software compatible with TMS9918A and V9938.

The V9958 has the following features that are supported in this implementation:

- Outputs linear RGB.
- Built-in color palette for display in up to 512 colors.
- Capable of simultaneous display of 19,268 colors by using YJK system display.
- Capable of displaying up to 512x424 Pixels and 16 colors.
- Bit mapped graphics.
- Capable of displaying maximum of 256 colors simultaneously.
- 16K byte ~ 128K byte usable for display memory.
- 256 addresses, 4ms auto refresh function of DRAM.
- Expansion video memory can be connected.
- Eight sprites can be displayed for each horizontal line.
- Colors for sprites can be specified for each horizontal line.
- Area move, line, search and other commands.
- Command function usable in every display mode.
- Logical operation function.
- Addresses can be specified by coordinates.
- Vertical and horizontal scroll function.

1.5.1.1 SuperSprite FM+ ™ V9958 Specifics

The minimum video RAM (VRAM) required for this board is 128KB. This is supplied by four TMS4464 or equivalent ICs (4-bits x 64K). The default RAM is installed into the sockets marked IC4-7. Please observe correct orientation. Pin 1 should be lower right.

To upgrade the board to the maximum 192KB, simply insert another two TMS4464 or equivalent ICs into ICs 8 and 9.

1.5.2 CXA1645 RGB Output

This card utilizes the Sony CXA1645 RGB Encoder chip to provide a clean and stable RGB output suited for 75Ω display devices.

1.5.3 YM2149

The YM2149 is an SSG (Software-Controlled Sound Generator) and is an NMOS-LSI device designed to be capable of music generation. It only requires the host computer to initialize

its register array, thus reducing load on the CPU. Music generation is carried out by the three square-wave generators, noise generators and envelope generators according to the set parameters. This allows for the generation of music, special effects, warnings, and various other types of sounds.

The chip was essentially an enhancement of the General Instrument AY-3-8910 chip that was more suitable to use with the MSX computers as it contains a built-in clock frequency divider. This means that it can use the clock signal which is directly output from the V9958 chip.

The YM2149 has the following features that are supported in this implementation:

- Simple programming using the 8-bit 6809E or 6309E.
- Simple connection to external system through two 8-bit I/O ports.
- Wide voicing range of 8 octaves.
- Smooth attenuation by 5-bit envelope generator
- Built-in 5-bit D/A converter.
- Built in clock-frequency division.
- Low power consumption (typical 125 mW).

1.5.4 YM2413

The YM2413 is an LL-Type FM Operator, incorporating a DA Converter and Quartz Oscillator in addition to a Yamaha original FM Sound Generator, allowing for a much easier and more economical sound generating system assembly than comparable LSIs of the era. Tone data is stored in ROM for software simplicity, making it possible to execute data alterations involved in tone changes with just one 'instrument selection' operation. Furthermore, a built-in Tone Data Register with capacity to store one tone allows for the generation of sound effects and original tones.

The YM2413 was selected for this board owing to its small footprint and the fact that it does not require the provision of an additional digital-to-analog converter.

The YM2413 has the following features that are supported in this implementation:

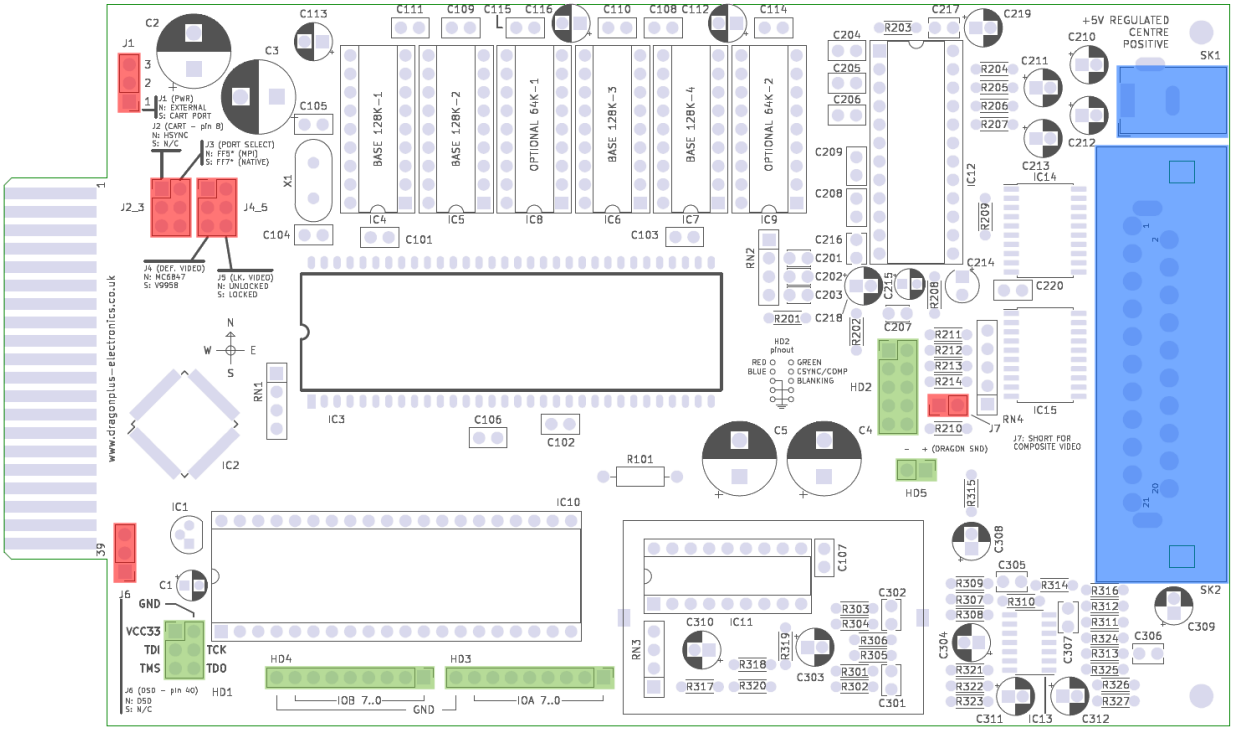
- FM Sound Generator for real sound creation.
- Two selectable modes: 9 simultaneous sounds or 6 melody sounds plus 5 rhythm sounds (different tones can be used together in either case).
- Built-in Instruments data (15 melody tones, 5 rhythm tones).
- Built in DA Converter.
- Built-in Quartz Oscillator
- Built-in Vibrato Oscillator/AM Oscillator.

1.6 BOARD LAYOUT

1.6.1 Important Board Locations

The colour-coded diagram below shows jumpers, headers and sockets as follows:

- Jumpers – RED
- Headers – GREEN
- Sockets – BLUE



N.B. markings on physical board supersede any denoted here, due to design improvements etc.

1.6.2 Board Jumpers

Compass orientation is printed on board. Jumper settings are denoted as North, South, Open or Closed.

1.6.2.1 J1: Power Supply Select

- *N: External power [DEFAULT]*
- *S: Host computer power*

The SuperSprite FM+ ™ card can be powered in one of two ways. Either the internal power of the host computer can be used if it has at least 800 mA of spare power capacity. Alternatively, a standard wall adapter that can supply 5V @1A, centre-positive via a 2.1mm or 2.5mm barrel jack may be used.

1.6.2.2 J2: HSYNC (Experimental Use – untested/unsupported)

- *N: CART (pin 8) connected to V9958 HSYNC*
- *S: CART (pin 8) unconnected [DEFAULT]*

1.6.2.3 J3: Port Select

- *N: Port range $FF56 - $FF5E (MPI compatible)*
- *S: Port range $FF76 - $FF7E (Dragon MMC compatible) [DEFAULT]*

The board has been designed to support two port address ranges – $FF56 - $FF5E or $FF76 - $FF7E – to ensure the best compatibility across the Dragon and Tandy Color Computer range, and to support the various multi-cartridge and disk solutions.

When the $FF56 - $FF5E range is in use, the ports are fully decoded and gated with P2/CTS. This ensures compatibility with Tandy MPI systems. When using this range, the DSD/SLENB signal back into the host computer is held at high impedance, preventing the P2/CTS line from being suppressed.

When the $FF76 - $FF7E range is in use the ports are fully decoded and DSD/SLENB is pulled low whenever either of the audio chips is addressed.

1.6.2.4 J4: Default Video Select

- *N: MC6847 (via RGB or composite inputs – HD2)*
- *S: V9958 [DEFAULT]*

Using two 74CBT3245 switching ICs, the board can switch between outputting the V9958 RGB signal or an internal video signal from the Dragon/Coco machine. The latter needs to be either composite or RGB.

The default display can be selected using a jumper on the board. The display can also be switched dynamically from code.

1.6.2.5 J5: Lock Video

- *N: Video switching is unlocked [DEFAULT]*
- *S: Video switching is locked*

Whilst software written to use the SuperSprite FM+ ™ board should always ensure that the correct video output is selected, many setups will be configured as dual screen. This jumper prevents software from changing the user selected default video (J4).

1.6.2.6 J6: DSD/SLENB select

- *N: DSD/SELNB is controlled by SuperSprite FM+ ™ board [DEFAULT]*
- *S: DSD/SLENB is unconnected.*

In MPI / FF5x port mode, this jumper should have no effect, as DSD/SLENB is held at high impedance. In FF7x port mode, this jumper can be used to prevent the SuperSprite FM+ ™ from pulling DSD/SLENB low. When using the Dragon MMC, this jumper should be set N.

1.6.2.7 J7: RGB/Composite Signal Select

- *Open: RGB Signal [DEFAULT]*
- *Closed: Composite Signal*

This jumper simply shorts out R214, which carries either the RGB Sync signal, or the composite signal. For composite no resistance is required, so this essentially bridges R214 to be zero ohms. Also see section 1.6.3.2 (HD2) for more details on how to use the board for RGB/composite host inputs.

1.6.3 Board Headers

1.6.3.1 HD1: CPLD Programming Header

- *Unused in normal deployment.*

1.6.3.2 HD2: Host Computer Display Input

- *Pinout printed onto PCB.*
- *For composite signal, pull BLANKING pin to 0v and fit J7.*
  - *The signal may require a 75R resistor load. If so, populate RN4 (as below).*
- *For RGB, resistors will need to be fitted to carry the signal through to the CBT3245 switches.*
- *Depending on the RGB device being used, these resistors will vary.*
  - *E.g., For Prime's Dragon RGB board, R211 - R213 = 220R; R214 = 270R.*
  - *I cannot offer support on any specific RGB devices.*
- *RN4 provides space for 4x 75R resistors to load the RGBS signal properly.*
  - *The square pad (south) is ground. Above this, from north-to-south is R, G, B and S.*
  - *Either 1 4x 75R array, or 4 75R 0.25W / 0.125W resistors can be used.*

1.6.3.3 HD3 & 4: YM2149 IO Ports A & B

- *Simple I/O ports supplied by YM2149 as standard. Can be used as digital joystick ports.*
- *All bits are pulled high internally by the YM2149.*

1.6.3.4 HD5: Host Computer Audio Input

- *One-channel audio input. Should be a screened cable connection.*
- *The ground pin is furthest (west) from the SCART connector.*

1.6.4 Board Sockets

1.6.4.1 SK1: External Power Supply Jack

- *Centre-positive 2.1mm or 2.5mm barrel jack supplying 5V @ 1A.*

1.6.4.2 SK2: SCART Socket

- *Connection to TV or SCART-HDMI Converter. Relevant pinout printed on underside of PCB.*

2 WHAT NOW?

2.1 QUICK-ISH START USER GUIDE

I know that quick start guides are usually printed on a very pretty, glossy fold-out piece of paper with completely intuitive diagrams. But that's when things are designed by big (or small) companies. This manual has been put together by just one person, and there really is a lot to cover! So, I am going to try and give 'get up and running' instructions as succinctly as possible here.

2.1.1 Getting Acquainted

I hope you will grow to love this board. I have certainly become extremely attached to it during its development. But just as with most relationships, things can be a bit awkward at first. What so say…what to do…where to put things…how to press the right buttons! ☺

Section 1.6 gave most of the details about what's what on the board. If you understood exactly what that section told you, and you are confident in getting up and running, then be my guest. But I am sure a lot of people will want some more explanation. So here goes.

2.1.2 Fit Standoff Pegs

You will need to decide how you intend to install this board. It is supplied as a bare-board – so in some configurations you will need to work out how to stabilize it. For instance, in an MPI, without a case, the board will be somewhat unstable, so you may need to fashion some kind of support.

If inserting the board into the cartridge port then you should use the standoff pegs to make supporting 'legs' at the SCART socket end of the board (screw one peg into the other with the board between. You should always ensure that the board is as level as possible when inserted into the cartridge port.

If you are using an Oojamaflip to install inside a Dragon, then don't forget to insert the SuperSprite FM+ ™ board 'solder side up'.

2.1.3 A Quick Background

This is not the potted history of the board's development. You can find that elsewhere online. This is just an explanation of why the board has been architected in the way it has.

Originally the board was a minor revision of CocoDemus' WordPak 2+ board. I'd built one, really liked it, and wondered if I might be able to refine the video output somewhat, as well as add just one audio PSG chip. The WordPak 2+ works by using ports $FF78 - $FF7B to control the V9958 chip – in place of the usual ports $FF98 - $FF9B that the MSX 2+ computers control the V9958 with.

To cut a long story short, I eventually ended up using ports $FF78 - $FF7F to control the V9958, YM2149 and the newcomer, the YM2413 FM sound chip. It worked OK for me, when I was testing using the Oojamaflip, but not for Pere when he was using his Tandy MPI. It turns out that $FF7F is used to tell the Tandy MPI what cartridge slot your program wants to access. Those port ranges were modified somewhat as you will see later in the document.

I ended up emailing Ed Snider (Zippster), who advised me that I ought to be making use of the 'proper' cartridge selection process – which is to use ports between $FF40 and $FF5F, controlling the device in use by setting the MPI port with $FF7F.

This presented me with a problem. The proper cartridge ports would be fine for anyone who had a Tandy MPI and that used a Coco SDC or normal floppy disk interface. But anyone using a DragonMMC[2] would find the device incompatible. I could have taken a short cut here and opted for just one set of ports – but I always wanted this board to be as widely usable as possible.

2.1.4 The SuperSprite FM+ ™ - Your Flexible Friend

OK – SuperSprite FM+ ™ is flexible – but please don't bend it like an Access credit card![3]

With some assistance from Phill Harvey-Smith, I engineered the board's port logic to be configurable using jumpers to support BOTH port ranges. That's not the only flexible aspect of the board. Let's look at those options in a bit more detail. Please note this is from a user's perspective. Developer's notes will follow later in the manual.

1. **The Power Supply:**
    a. You can power this board from the 5-volt supply on the cartridge port if you are certain that your computer has spare capacity of around 800mA.
    b. You can power this board externally using the barrel jack socket. You will need a 2.1mm or 2.5mm. **You must make sure that you set your power supply to provide the barrel jack with centre positive at 5 volts DC.[4]**

2. **Port Ranges:** the ports are, at machine level, how the board is controlled. There are two port ranges:
    a. $FF56 - $FF5E – MPI compliant
    b. $FF76 - $FF7E – native mode

    If you intend to use a DragonMMC device alongside this board then you will want the board to be set to native mode. If you intend to use just a CocoSDC or traditional floppy interface, then you can set either mode. If you have a more complex hardware setup, using diverse cartridge port ranges, then you will need to think carefully about your configuration.

3. **Default Video Selection:**

[2] This is not because the DragonMMC is wrongly architected. It's because the DragonMMC does so much more than act as a disk drive – like cassette and cartridge emulation as well as program snapshots. For this reason it cannot be gated using P2 like other cartridge devices.

[3] Some readers who remember the Access credit card in the UK may be interested to look at one of my other websites which covers the history of that credit card – https://www.accesscreditcard.info

[4] There is no external voltage protection circuitry on this board. To do so would necessitate more components and essentially a higher input voltage at the barrel jack, as even the simplest diode protection would involve a voltage drop of 0.6 – 0.7 volts. Please be careful to get the polarity correct.

a. MC6847. If you have either an RGB or composite MC6847 signal, then you can connect this to the SuperSprite FM+ ™ board, as per the instructions in section 1.6.3.2. In this instance, you would be setting the machine up as a single screen system – i.e. you would expect the SuperSprite FM+ ™ to switch between the MC6847 and V9958 outputs as necessary.
b. V9958. If you have neither an RGB or composite MC6847 signal, then you are likely to wish to use this board in a dual screen setup – or at least, will be controlling the TV input yourself.
c. Note that a warm boot (press of RESET) will always result in the default screen mode being selected again.

4. **Video locking:**
    a. Locked. If you do not wish software to be able to change your default video selection, then you will want to lock the video. This is most suitable for dual-screen setups, or setups where you have one TV, and wish to control the source input yourself.
    b. Unlocked. If you have connected either an RGB or a composite signal into the SuperSprite FM+ ™ board, and are comfortable with software switching between the two signals, then this is the mode to select. You'd probably set the default video to MC6847, so it boots up with the usual Dragon or Coco screen, and then software can switch video output at the correct time.

5. **Advanced Options**

    As this is a quick start guide (of sorts), I strongly recommend that you leave these jumpers well alone. They are described in more detail later in this document.

    a. DSD/SLENB Enable/Disable:
    b. HSYNC – CART Connection:

2.1.5 Running Software

If you want to run software supplied by other people, then as long as you have set your jumpers correctly, it's really just a case of making sure you are running software that's using the correct ports for your settings.

Even if you are using an MPI, then the board will work perfectly well using the FF7x (Native) port settings. But if you have other hardware connected that conflicts with those FF7x ports, then you'll need to use FF5x (MPI) mode.

2.2 TROUBLESHOOTING

2.2.1 DragonMMC Snapshot Workaround

The DragonMMC can perform snapshots of entire memory states of your Dragon or CoCo machine. Unfortunately, it uses the same interrupt mechanism to handle this as the SuperSprite FM+ ™ board uses to send VSYNC back to the CoCo/Dragon – the Non Maskable Interrupt (NMI).

Currently, therefore, DragonMMC snapshot functionality must be disabled before using any V9958 functionality, to avoid hardware crashes. This is achieved using the following POKE from BASIC.

```
POKE &HFF56, PEEK(&HFF56) AND 31
```

If the program that you are trying to run is the $FF7x version, and upon execution you are experiencing crashes, then prior to executing the machine code, try entering the above POKE.

2.2.2 Garbage on Screen at Boot

I have had this fault numerous times – and it still catches me out to this day. Whilst it could, of course, be caused by other things, I've found that most of the time it's because I've got the SuperSprite FM+'s power jumper set incorrectly. E.g. I have it set to use external power, but have not plugged in a power adapter. Or vice versa – i.e. I'm expecting the board to use the host computer's power, but the jumper is set to the external power position.

3 PROGRAMMER'S GUIDE

3.1 SCOPE OF THIS GUIDE

This manual cannot possibly cover every element regarding programming this board. You'll ultimately need to consult the datasheets for the V9958, YM2149 and YM2413 in order to use those chips fully. The following gives some essential info and useful guidance that is specific to SuperSprite FM+ ™.

3.2 THE DEVICE PORTS

This board has a lot of address ports. It would theoretically be possibly to reduce the numbers of ports, but it would likely add more complexity to programming, and possibly use up slightly more processor cycles at run time.

Essentially you can address each chip exactly as Yamaha specify in the datasheets. The table below specifies the port numbers and descriptions for each of the two mapping options.

				Mode	
				MPI	
Device	**Port Description**	**R**	**W**	**Compliant**	**Native**
YM2413	Register Address		✔	$FF56	$FF76
	Register Data		✔	$FF57	$FF77
V9958	Port #0 VRAM Data	✔	✔	$FF58	$FF78
	Port #1 Status Register	✔		$FF59	$FF79
	VRAM Address		✔		
	Register set-up		✔		
	Port #2 Palette Registers		✔	$FF5A	$FF7A
	Port #3 Register Indirect Addressing		✔	$FF5B	$FF7B
YM2149	Register		✔	$FF5C	$FF7C
	Data	✔	✔	$FF5D	$FF7D
Video Mux	Video out select - Bit 0 (0=V9958; 1=MC6847)		✔	$FF5E	$FF7E

3.3 PORTABLE CODE

If you use a DragonMMC, or if you want to write code that people with a DragonMMC can use, then you will want to support the native mode. But equally, if you want to ensure that your code will run successfully in MPI compliant environments, then you will want to use the MPI compliant ports.

In most cases, the native mode ports will work fine. Unless, of course, you happen to want to use devices which already reside in those port addresses.

The simple advice to programmers is to write code which supports both. This is actually quite easy when using a compiler such as Ciaran Anscomb's excellent asm6809. The following guidance has been supplied by Pere Serrat.

The trick is having if-then-else directives to tell the compiler which part should be used and which part should be discarded dependent upon the value of a variable that is set in the batch file. The batch file calls the compiler twice using a different value for that variable.

Below is a sample batch file and the beginning of the assembler source code containing the IF_ELSE.

COMPILE.BAT

```
asm6809 -3 -D -v --define MPI=1 --l=%1M.LST.ASM --o=%1M.BIN %1.ASM
asm6809 -3 -D -v --define MPI=0 --l=%1O.LST.ASM --o=%1O.BIN %1.ASM
```

VGM.ASM

```
; --------------------------------------------------------------------------------
--
; VGMPLAY v0.H Pere Serrat 2021-06-22
; Play VGM files from disk on SuperSprite FM+ ™ module for MPI or OOJAMAFLIP
; NOT using interrupts but calculated delays to synchronize received commands
; --------------------------------------------------------------------------------
--
HOOKSINI        equ     $015e   ; system Hooks beginning
HOOKSEND        equ     $01a9   ; system Hooks end
MPICFG          equ     $ff7f   ; MPI control register
STRNGOUT        equ     $90e5   ; to print string on screen
        IF MPI
VREG            equ     $ff56   ; SSFM+ YM2413 register Port for MPI
VDATA           equ     $ff57   ; SSFM+ YM2413 data Port for MPI
FMPORT          equ     $ff5c   ; SSFM+ YM2149 register select port for MPI
FMDATA          equ     $ff5d   ; SSFM+ YM2149 data port for MPI
        ELSE
VREG            equ     $ff76   ; SSFM+ YM2413 register Port for OOJAMAFLIP
VDATA           equ     $ff77   ; SSFM+ YM2413 data Port for OOJAMAFLIP
FMPORT          equ     $ff7c   ; SSFM+ YM2149 register select port for OOJAMAFLIP
FMDATA          equ     $ff7d   ; SSFM+ YM2149 data port for OOJAMAFLIP
        ENDIF ; MPI
SPEED1          equ     $ffd8           ; std speed
SPEED2          equ     $ffd9           ; double speed
; --------------------------------------------------------------------------------
--
                                org     $1500
                                put     $1500
; --------------------------------------------------------------------------------
--
```

3.3.1 File notation

You may notice above that Pere's example suffixes the generated assembler listing and binary files with either the letter 'O' or 'M'. That has become a default notation as it has distinguished between MPI-compliant ($FF5x) and Oojamaflip ($FF7x) versions of the generated code.

If you are writing code for others to run, then obviously you can use whatever notation makes the most sense to you, but you should make it clear to your users which code is compiled for which range of ports.

If you require BASIC loader modules, then you will need to retain a similar file naming convention for those as well. Or just keep the code for the different modes in separate folders.

3.3.2 Supporting End-User Screen Configurations

Some people will use the SuperSprite FM+ ™ in single-screen mode, whilst others will use it in dual-screen mode. In order that software works well for both sets of users, the following is recommended:

- Even if you are developing on a dual screen system, ensure that you always assert the correct screen mode into the board's Video Out Select. For instance, if your program starts in MC6847 screen mode, then to ensure that everyone's SuperSprite FM+ ™ is displaying correctly, you should POKE &HFF7E,1 or POKE &HFF5E,1.
- Similarly, when you wish to switch across to V9958 mode, issue the opposite command – i.e. POKE &HFF7E,0 or POKE &HFF5E,0.
- Those people using a dual-screen set-up will have set their video to 'locked' and so these screen mode change POKEs will have no effect for them.

Note that a warm boot (press of RESET) will always result in the default screen mode being selected again.

3.3.3 DragonMMC Snapshot Workaround

The DragonMMC can perform snapshots of entire memory states of your Dragon or CoCo machine. Unfortunately, it uses the same interrupt mechanism to handle this as the SuperSprite FM+ ™ board uses to send VSYNC back to the CoCo/Dragon – the Non Maskable Interrupt (NMI).

Currently, therefore, DragonMMC snapshot functionality must be disabled before using any V9958 functionality, to avoid hardware crashes. This is achieved using the following POKE from BASIC.

```
POKE &HFF56, PEEK(&HFF56) AND 31
```

3.4 Clock Frequencies

To play music/sounds correctly from the YM2149/YM2413, you will need to know the clock frequencies being used to drive these chips. This differs from some other computer implementations – notably Ed Snider's CoCo-PSG – which follows the Atari ST clock configuration.

This board has both the YM2149 and YM2413 being driven using the clock output from the V9958, which is how this is set up for the MSX 2+ range of computers.

Essentially the V9958 chip is driven using a 21.47727 MHz crystal. One pin on the V9958 outputs another clock signal which is exactly one-sixth of this clock. Therefore, the YM2149 and YM2413 chips are being driven by a 3.579545 MHz clock. It is no coincidence that this is the same frequency as the NTSC subcarrier – nor is it a secret that the Z80 CPU in some machines with these chips is clocked using that V9958 output clock.

The YM2149 has an additional functional pin over its older sibling, the AY-3-8910 audio chip. The additional pin was added by Yamaha when they licensed the design from General

Instruments – and its sole purpose is to divide the input clock by 2 (3.579545 MHz would be too high for practical use). Therefore, the usable clock frequency for this board is as follows:

- YM2149 – 1.789773 MHz
- YM2413 – 3.579545 MHz

3.5 PROGRAMMING THE V9958

There is some sample BASIC code on CocoDemus' site for the WordPak 2+ which can be used to check the operation of the V9958. Most graphical programming will be done in machine code until such time as someone is able to develop BASIC extensions for the board (that won't be me!)

3.5.1 Sample Assembler Code

The following code is from Pere Serrat, and this is designed to display chequered bars on the screen. You'll see that it is written in a format best compiled by ASM6809, as it makes use of the methods already described in section 3.3 for compilation across both sets of port.

3.6 PROGRAMMING THE YM2149

Ultimately this manual cannot include the complete specifications and guidelines as to how the YM2149 is programmed. That detail can be found in the original YM2149 datasheet. However, some guidelines are presented in this section as well as in Appendix E, where a sample BASIC program is provided.

3.6.1 High/Low Byte Values for Note Generation

Given that the YM2149 clock frequency is known, the note frequencies have been supplied in Appendix A.

These are the values which should be used with registers 0 & 1, 2 & 3 and 4 & 5. For instance, to play 2nd octave C on channel A:

- Write 185 to R_0 and 6 to R_1.

3.6.2 The YM2149 I/O Ports

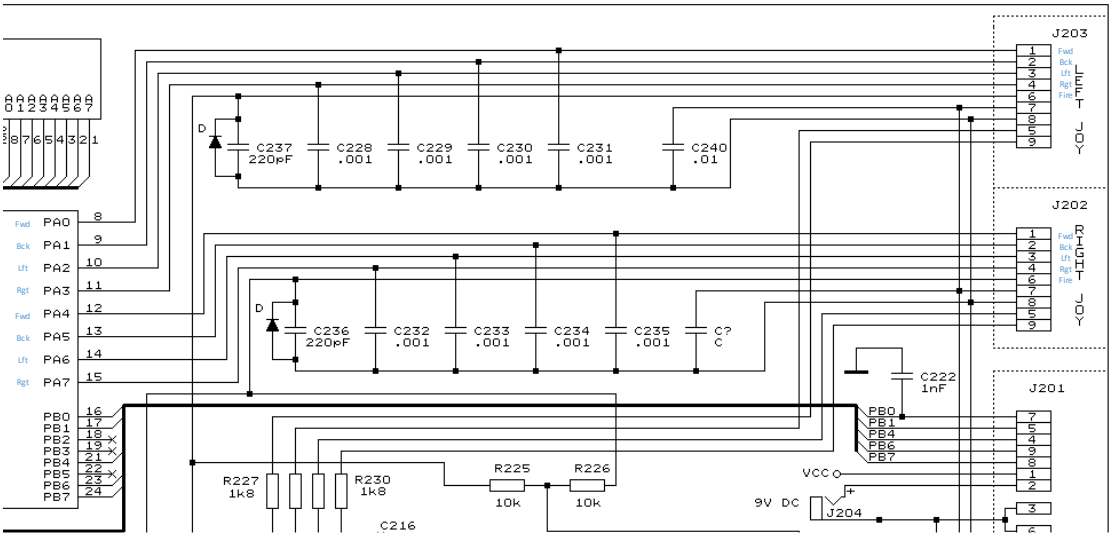
Please note that this is a tested, but unsupported, function. What do I mean by that? I mean that I have tested the I/O port using BASIC commands to detect the current state of both ports A and B. And I concluded that the settings of my 8-bit DIP switches were reflected in the port reads.

This means that architecturally I know this to be sound, but that I have not tested every YM2149's I/O ports A & B, and do not guarantee that this will work correctly. This certainly does not mean that I am selling duff chips – just that the I/O ports are a bonus that I did not mandate in my design. Put simply, I will sell you a YM2149 that I know is musically sound, but I have not tested the I/O ports at all. So if there is an issue there, you will need to source an alternate YM2149.

The YM2149 I/O ports are easy to read – even from BASIC.

I/O Port A	I/O Port B
POKE &HFF*C,7	POKE &HFF*C,7
POKE &HFF*D,191	POKE &HFF*D,127
POKE &HFF*C,14	POKE &HFF*C,14
?PEEK(&HFF*D)	?PEEK(&HFF*D)

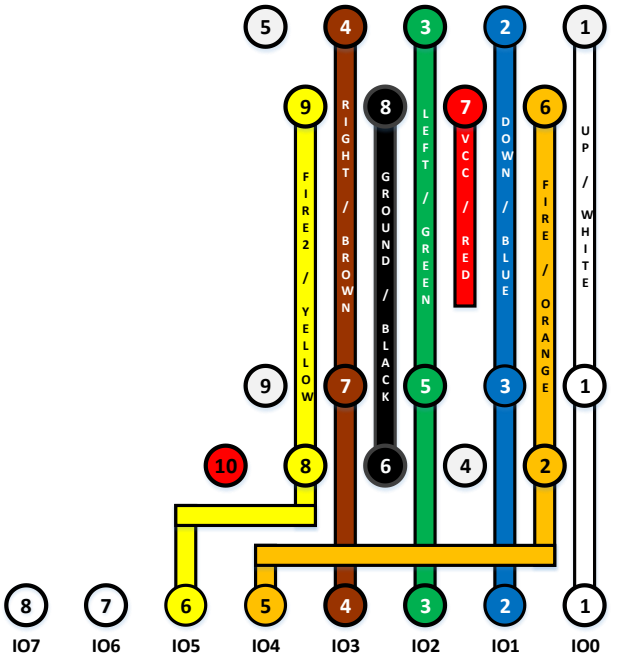
3.6.2.1 Using the YM2149 I/O Ports for Digital Joysticks



As per the above schematic, the Atari 2600 console used one 8-bit wide bus to read position of both joysticks, but not fire button with the following order (LSB first): Forward/Up, Backward/Down, Left, Right.

The Coco-PSG also uses the directions in the same order – however, it uses one 8-bit port per controller, including the fire button. This makes much more sense for the Dragon implementation as well – and therefore the usage of the I/O ports are recommended as per the Coco-PSG.

Pinout from solder side of straight 9-pin DSUB connector.



3.7 PROGRAMMING THE YM2413

For full details of how to program the YM2413, please consult the datasheet. The following BASIC program is a very simple test to enable you to hear the chip in action.

```
10 RE=&HFF76:'CHANGE TO &HFF56 for MPI mode
15 DA=RE+1
20 READ RE$,DA$
30 IF RE$="END" THEN RESTORE:FOR P=1 TO 100:NEXT P:GOTO 20
35 IF RE$="PAU" THEN FOR P=1 TO 100:NEXT P:GOTO 20
40 POKE RE,VAL("&H"+RE$)
50 POKE DA,VAL("&H"+DA$)
60 GOTO 20
1000 DATA 00,20,01,20,02,3F,03,00,04,FF,05,FF,06,0F,07,0F
1010 DATA 10,C1
1020 DATA 30,00
1030 DATA 20,11
1040 DATA PAU,PAU
1050 DATA 20,13
1060 DATA PAU,PAU
1070 DATA 20,15
1080 DATA PAU,PAU
1090 DATA 20,17
1100 DATA PAU,PAU
1110 DATA 20,19
1120 DATA PAU,PAU
1130 DATA 20,1A
1140 DATA PAU,PAU
1150 DATA 20,1C
1160 DATA PAU,PAU
1170 DATA 20,1F
2000 DATA END,END
```

APPENDIX A – YM2149 PSG NOTE VALUES @ 1.789773 MHZ

	DESIRED FREQUENCY (Hz)								CALCULATED FACTOR = Freq / (16 * Fnote)							
Octave	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
C	33	65	131	262	523	1047	2093	4186	3390	1721	854	427	214	107	53	27
	35	69	139	277	554	1109	2217	4435	3196	1621	805	404	202	101	50	25
D	37	73	147	294	587	1175	2349	4699	3023	1532	761	380	191	95	48	24
	39	78	156	311	622	1245	2489	4978	2868	1434	717	360	180	90	45	22
E	41	82	165	330	659	1319	2637	5274	2728	1364	678	339	170	85	42	21
F	44	87	175	349	698	1397	2794	5588	2542	1286	639	321	160	80	40	20
	46	92	185	370	740	1480	2960	5920	2432	1216	605	302	151	76	38	19
G	49	98	196	392	784	1568	3136	6272	2283	1141	571	285	143	71	36	18
	52	104	208	415	831	1661	3322	6645	2151	1076	538	270	135	67	34	17
A	55	110	220	440	880	1760	3520	7040	2034	1017	508	254	127	64	32	16
	58	117	233	466	932	1865	3729	7459	1929	956	480	240	120	60	30	15
B	62	123	247	494	988	1976	3951	7902	1804	909	453	226	113	57	28	14

	HIGH and LOW BYTES to be SENT TO THE YM2149															
Octave	1		2		3		4		5		6		7		8	
Values	High	Low	High	Low	High	Low	High	Low	High	Low	High	Low	High	Low	High	Low
C	13	62	6	185	3	86	1	171	0	214	0	107	0	53	0	27
	12	124	6	85	3	37	1	148	0	202	0	101	0	50	0	25
D	11	207	5	252	2	249	1	124	0	191	0	95	0	48	0	24
	11	52	5	154	2	205	1	104	0	180	0	90	0	45	0	22
E	10	168	5	84	2	166	1	83	0	170	0	85	0	42	0	21
F	9	238	5	6	2	127	1	65	0	160	0	80	0	40	0	20
	9	128	4	192	2	93	1	46	0	151	0	76	0	38	0	19
G	8	235	4	117	2	59	1	29	0	143	0	71	0	36	0	18
	8	103	4	52	2	26	1	14	0	135	0	67	0	34	0	17
A	7	242	3	249	1	252	0	254	0	127	0	64	0	32	0	16
	7	137	3	188	1	224	0	240	0	120	0	60	0	30	0	15
B	7	12	3	141	1	197	0	226	0	113	0	57	0	28	0	14

APPENDIX B – BOARD HIGHLIGHTS

This peripheral device has been designed and developed with much passion and care. Although somewhat limited for choice with the truly retro components (you procure whatever you can of the V9958, YM2149, YM2413, CXA1645 and RAM chips), all new components have been chosen carefully to offer the best performance.

Amongst other components, this board contains:

- 4 networked resistors
- 5 headers
- 7 jumpers
- Between 13 and 15 integrated circuits (depending on VRAM)
- 42 resistors
- 53 capacitors
- 640 solder joints

With the exception of those used for IC decoupling, all capacitors have been chosen for quality rather than cost.

- All audio/video circuit electrolytics are either Rubycon or Panasonic.
- Most audio/video circuit ceramic capacitors are Kemet C0G/NP0 – the highest quality ceramic dielectric.

The board you see before you is the fifth version of the board. Board prototyping, whilst cheaper than in the past, is still a relatively expensive exercise – but it is far better to get the boards perfect before public release.

APPENDIX C – BOARD/COMPONENT POWER REQUIREMENTS

	Single Component Current (mA) (from manufacturer datasheets)				
	Min	Avg	Max	Board Count (Max)	Total Power
V9958			230	1	230
YM2149		25	40	1	40
YM2413		5	10	1	10
CXA1645		(31+12) = 43		1	43
TMS4464			65	6	390
BA14741			37	1	37
XC9536XL			10	1	10
74CBT3245			2.5	2	5
				Total mA	**765**

The above are maximum values and are unlikely to be reached – for instance, it is unlikely that all six memory chips would be accessed at once. The more likely scenario is two at once.

However, it is a prudent value to consider when working out the power requirements for the board – and thus 800mA of spare power supply capacity is recommended.

APPENDIX D – SAMPLE ASSEMBLER CODE FOR V9958

```
; ------------------------------------------------------------------------------
; G7CHKR v1 for SuperSprite FM+ module - Pere Serrat 2021-02-08 ; annotations updated where highlighted – John Whitworth 2021-08-30
; will work in native mode if HD6309 present but doesn't use 6309 special opcodes!
; ------------------------------------------------------------------------------
                        org     $7000
; ------------------------------------------------------------------------------
HOOKSINI        equ     $015e                           ; system Hooks beginning
HOOKSEND        equ     $01a9                           ; system Hooks end
MPICFG          equ     $ff7f                           ; MPI control register
DISINT          equ     $50                             ; disable interrupts
SYSCUR          equ     $88                             ; word containing system cursor text pointer
ENAINT          equ     $af                             ; enable interrupts
SHOWTEXT        equ     $90e5                           ; $90e5 for Dragon; $b99c for Coco.
READKBD         equ     $a000                           ; indirect call to read keyboard
        IF MPI                                          ; for use in an MPI
PDATA           equ     $FF58                           ; VRAM data - Port #0
PCONF           equ     $FF59                           ; status, setup and VRAM address - Port #1
PPAL            equ     $FF5A                           ; palette port (via Reg#16)- Port #2
PINDA           equ     $FF5B                           ; indirect addressing (through Reg#17) - Port #3
        ELSE                                            ; for use in an OOJAMAFLIP
PDATA           equ     $FF78                           ; VRAM data - Port #0
PCONF           equ     $FF79                           ; status, setup and VRAM address - Port #1
PPAL            equ     $FF7A                           ; palette port (via Reg#16)- Port #2
PINDA           equ     $FF7B                           ; indirect addressing (through Reg#17) - Port #3
        ENDIF
; ------------------------------------------------------------------------------
PgmBeg          pshs    cc                              ; save interrupts state
                orcc    #DISINT                         ; disable interrupts ...
                jsr     $ba77                           ; do a text CLS for MC6847
                                                        ; switch to slot 0 where SSFM+ sits
                lda     MPICFG                          ; get current slot used by disc system
                sta     oldSlot                         ; save value
                clr     MPICFG                          ; point to the SSFM+ device. Slot 1 (value $00)
                                                        ; save system hooks
                ldx     #HOOKSINI                       ; point to 1st system hook
                ldu     #copHooks                       ; point to buffer area
                ldb     #$39                            ; get a RTS opcode
L01             lda     ,x                              ; get 1st byte of a hook
                sta     ,u+                             ; put into buffer
                stb     ,x                              ; substitute by a RTS
                leax    3,x                             ; point to next hook
```

```
               cmpx    #HOOKSEND                       ; already done?
               blo     L01                             ; no, loop
               lbsr    Set6309                         ; set 6309 native mode if present
; - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
PgmLp1         bsr     SetGraph                        ; set graphic mode (256x192 256 col)
               bsr     DoChkr                          ; create checkered patterns
; - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
PgmEx          orcc    #$50                            ; disable all interrupts
               ldx     #HOOKSINI                       ; point to 1st system hook
               ldu     #copHooks                       ; point to buffer area
L02            lda     ,u+                             ; get a byte from buffer
               sta     ,x                              ; put as 1st byte of hook
               leax    3,x                             ; point to next hook
               cmpx    #HOOKSEND                       ; already done?
               blo     L02                             ; no, loop
               lda     oldSlot                         ; get received slot number
               sta     MPICFG                          ; set MPI
               tst     is6309                          ; found a 6309 inside?
               beq     PgmExit                         ; no skip next
               fcb     $11,$3d,$00                     ; ldmd #0 (enter emulation mode)
PgmExit        puls    cc,pc                           ; restore interrupts state and return
; ------------------------------------------------------------------------------
SetGraph       ldu     #settings                       ; point to settings to be sent to WordPad2+
               ldx     #PDATA                          ; point to Data port
               ldb     #SETLEN                         ; number of bytes to be sent
SG01           lda     ,u+                             ; get a byte, move pointer
               sta     1,x                             ; send to SSFM+
               decb                                    ; decrement counter
               bne     SG01                            ; not yet done? loopback
               rts                                     ; return
; ------------------------------------------------------------------------------
DoChkr                                                 ; first must verify that VDP is free!
               ldd     #$028f                          ; S#2 to R#15

               sta     1,x                             ; send status register
               stb     1,x                             ; send register 15
DcLoop         lda     1,x                             ; get status data
               rora                                    ; send bit0 to carry
               bcs     DcLoop                          ; if set (still busy), loop
               ldd     #$0000                          ; to begin from top left of screen
               bsr     SetVdpWrit                      ; set point to begin writting
; - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                                                       ; Now paint some vertical checkered bars
               lda     #192/2                          ; groups of 2 rows to paint
               sta     groupCtr                        ; save to variable
NextGrup       ldu     #BarDat1                        ; point to data
NextLine       ldy     #8                              ; number of byte pairs to copy (different columns or bars)
```

```
NextPat         ldb     #16                             ; bar width in byte pairs (each 2 pixel)
NextByte        lda     ,u                              ; get a byte (1 pixel)
                sta     ,x                              ; send to VDP port #0
                lda     1,u                             ; get next byte (1 pixel)
                sta     ,x                              ; send to VDP port #0
                decb                                    ; decrement width control
                bne     NextByte                        ; not done, loop using same two bytes
                                                        ; we have painted 16-byte pairs = 32 bytes
                leau    2,u                             ; point to next byte pair
                leay    -1,y                            ; decrement pair counter
                bne     NextPat                         ; not done? loop
                                                        ; we have painted 8x32=256 bytes (256 pix) so none left!
                cmpu    #BarEnd                         ; got ot end of data?
                bne     NextLine                        ; no, loopback for inverted colours row
                                                        ; we have done a double line with chckered patterns
                dec     groupCtr                        ; decrement group counter

                bne     NextGrup                        ; not yet done? loop
                rts                                     ; return
; - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
BarDat1         fcb     $e0,$1c,$03,$e3,$fc,$1f,$01,$ff ; 4 byte pairs
                fcb     $22,$55,$77,$88,$99,$aa,$bb,$cc
                fcb     $1c,$e0,$e3,$03,$1f,$fc,$ff,$01 ; inverted to produce the checkered pattern
                fcb     $55,$22,$88,$77,$aa,$99,$cc,$bb
BarEnd          equ     *
; ---------------------------------------------------------------------------------------------
; sets the VRAM pointer to the medium and low byte received in regD adding write flag
; Saves it in a work variable without flag
; ---------------------------------------------------------------------------------------------
SetVdpWrit      ldx     #PDATA                          ; point to port #0
                std     <workPtr                        ; save received pointer for later use in caller
                ldd     #$008e                          ; get address high byte and register
                sta     1,x                             ; send address high byte to port #1
                nop                                     ; *** ADDED TO HELP VDP GET BYTES ***
                stb     1,x                             ; send register number to port #1
                ldd     <workPtr                        ; get received address
                stb     1,x                             ; send address low byte to port #1
                adda    #$40                            ; set write flag on
                sta     1,x                             ; send address medium byte + write flag to port #1
                rts                                     ; return
; ---------------------------------------------------------------------------
Set6309         clr     is6309                          ; set no 6309 present
                fcb      $10,$86,$55,$55                ; ldw   #$5555  (load the double register W = regE+regF)
                fcb      $1f,$61                        ; tfr   w,x            (transfer to regX)
                cmpx     #$ffff                         ; did it fail? (will fail on a 6809)
                beq      Not6309                        ; yes, skip next
                ldx     #text02-1                       ; point to 6309 message
```

```
              jsr     SHOWTEXT                          ; show it
              fcb   $11,$3d,$01                         ; ldmd  #$01   (enter native mode)
              inc     is6309                            ; flag 6309 present
Not6309       rts                                       ; return
; ---------------------------------------------------------------------------
copHooks      rzb     26                                ; buffer for 1st byte of hooks
oldSlot       fcb     $00                               ; received slot number in MPI
is6309        fcb     $00                               ; flag
workPtr       fdb     $0000                             ; work pointer
groupCtr      fcb     $00                               ; counter
; - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
settings      fcb     $0e,$80         ;  0  DG IE2 IE1  M5  M4  M3   0 -> set M3,M4,M5 (set G7 mode)
              fcb     $40,$81         ;  0  BL IE0  M1  M2   0  SI MAG -> set BL,/MAG,/IE0 (screen enable)
              fcb     $0a,$88         ; MS  LP  TP  CB  VR   0 SPD  BW -> set /TP,VR,/SPD (VRAM 64kx1, sprites disabled)
              fcb     $00,$89         ; LN   0  S1  S0  IL  E0 *NT  DC -> set 192 lines, NO interlace, no even/odd screens - NTSC
              fcb     $11,$87         ;TC3 TC2 TC1 TC0 BD3 BD2 BD1 BD0 -> set background/border colour (from palette), only low nibble works
; - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
              fcb     $1f,$82         ;  0   0 A16   1   1   1   1   1 -> Pattern Layout (select Page0) $00000 - %0000 0000 0000 0000
; - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
              fcb     $f8,$85         ;A14 A13 A12 A11 A10   1   X   X -> set A10 to 15 Sprite Attrib Table
              fcb     $01,$8b         ;  0   0   0   0   0   0 A16 A15 ->  here $0fa00 - %1111 1100 0000 0000
                                      ;                                 -> Sprite Colour Table is $200 above Attrib Table
                                      ;                                 -> here will be $01c00
              fcb     $1e,$86         ;  0   0 A16 A15 A14 A13 A12 A11 -> set A12-13-14-15 Sprite PattGen  $0f000 - %1111 0000 0000 0000
; - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
              fcb     $01,$92         ; V3  V2  V1  V0  H3  H2  H1  H0 -> center image horizontally - NTSC
; - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
setRew        fcb     $00,$8e,$00,$40 ; send address $000000 to reg14 and write enable
SETLEN        equ     *-settings      ; should be 16
; - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
text02        fcn     /FOUND A HD6309 CPU INSIDE/
; ---------------------------------------------------------------------------
       if ((*+9)%256 > 0)                               ; if not multiple of 256
              rzb      256-((*+9)%256)                  ; fill to make it DW4 compatible
       endif                                            ; end
; ---------------------------------------------------------------------------
PgmEnd        equ     *
```

APPENDIX E – SAMPLE BASIC PROGRAM FOR YM2149

The following code is a slightly simplified version of that which was demonstrated on the special CocoTalk edition, broadcast 14th August 2021.

```
0 REM PSG-TEST.BAS by John Whitworth (2021)
1 REM Based on an MSX-BASIC program by A. Gilbert (1985)
2 HDW=&HFF7C:GOTO 10: 'SET TO &HFF5C FOR MPI MODE
4 FORRG=0TO6:POKE HDW,RG:POKE HDW+1,DV(RG):NEXT:FORRG=8TO13:POKE HDW,RG:POKE HDW+1,DV(RG):NEXT:POKE HDW,7:POKE
HDW+1,DV(7):RETURN
10 GOSUB 9000:'SET T2 MODE
15 GOSUB 9800:'DIM VARIABLES ETC.
20 CLS
30 PRINT "PSG TEST PROGRAM"
40 GOSUB 9600:'INSTRUCTIONS
50 CLS
60 GOSUB 9400:'SCREEN LAYOUT
70 POKE&HFFD7,0:'Main Loop
80 IF RC=0 THEN 110
90 PRINT@((2+LR)*32)," ";:PRINT@((2+SL)*32),">";
100 LR=SL:RC=0
110 A$=INKEY$
115 IF A$="C"THENCN=1
116 IF A$="S"THENCN=0
120 IF A$=CHR$(10)ORA$=CHR$(91)THENSL=SL+1:RC=1
130 IF A$=CHR$(94)ORA$=CHR$(95)THENSL=SL-1:RC=1
135 IF A$="Q" THEN 1000
140 IF SL<0THENSL=0
150 IF SL>13THENSL=13
160 IFA$=CHR$(8)ORA$=CHR$(21)THENDV(SL)=DV(SL)-1:VC=1
170 IFA$=CHR$(9)ORA$=CHR$(93)THENDV(SL)=DV(SL)+1:VC=1
180 IF VC=0THEN300
190 IFDV(SL)<0THENDV(SL)=0
200 IFDV(SL)>SV(SL)THENDV(SL)=SV(SL)
210 PRINT@((2+SL)*32)+28,"";:PRINTUSING"###";DV(SL);
220 VC=0
225 GOSUB4
```

```
300 IFCN=1THENGOSUB4
500 GOTO 80
1000 REM IF MPI MODE SELECT SDC AND EXIT TO BASIC
1070 IF HDW=&HFF7C THEN GOTO 1090
1080 POKE&HFF7F,&H33:POKE&HFFD6,0
1090 END
9000 REM SET T2MODE
9010 POKE HDW+2,0:REM FF5E/FF7E
9020 IFCHR$(PEEK(&H7802))<>"T"THEN9040
9030 I=PEEK(&H7803)-48:IFI=1ORI=2THENEXEC&H7800 ELSEEND
9040 FORI=&H80 TO&H8D:POKE HDW-3,0:POKE HDW-3,I:NEXT: REM FF59/FF79
9045 IF HDW=&HFF7C THEN LOAD"T2INSTCO.BIN":CLOSE:EXEC&H7400:GOTO 9060
9050 LOAD"T2INSTCM.BIN":CLOSE:EXEC&H7400
9060 V=&HF4:M=HDW-3:POKEM,V:POKEM,&H87:POKEM,V:POKEM,&H8C
9070 REM IF MPI CHANGE TO SuperSprite. Hooks are saved by the driver
9080 IF HDW=&HFF7C THEN GOTO 9099
9090 POKE&HFF7F,0
9099 RETURN
9400 PRINT@0,"  Rn : Register Name  :   Value"
9410 FOR RG=0TO13:PRINT@((RG+2)*32)+2,"";:PRINTUSING"##";RG;:PRINT" : ";:PRINTUSING"%            %";RN$(RG);" :
";:NEXT
9499 RETURN
9600 PRINT:PRINT"Instructions? (Y/N)"
9610 A$=INKEY$:IF A$="" THEN 9610
9620 IF A$="N" OR A$="n" THEN RETURN
9630 IF A$<>"Y" AND A$<>"y" THEN GOTO 9610
9640 CLS:PRINT"Instructions":PRINT
9650 PRINT "Use the up/down arrow keys to move to the desired register number (below the    green line). Then
increase/decrease register value (above green line) with left/right arrow keys."
9690 PRINT:PRINT"As on-screen info is limited, you should consult the YM2149/AY-3-8910 datasheet for information
regarding the sound register functions. Remember that if all    volume channels (regs 8-10) are set to off, then
no sound will be heard.
9700 PRINT:PRINT"Press any key to leave instructions and start program."
9710 A$=INKEY$:IF A$="" THEN 9710
9799 RETURN
9800 DIM DV(14),RN(14),RN$(14),SV(14):'DATA VAL, REG NO, REG NAME, MAX REG VAL
9810 FORB=0TO13:READSV(B):NEXT
```

```
9820 FORB=0TO13:READRN$(B):NEXT
9830 SL=0:'Sel. reg. 0
9840 RC=1:LR=0
9850 DV(7)=56
9851 DV(11)=170:DV(12)=17
9890 RETURN
9900 DATA 255,15,255,15,255,15,31,63,16,16,16,255,255,15:REM Max channel values
9910 DATA "Freq A fine","Freq A rough","Freq B fine","Freq B rough","Freq C fine","Freq C rough"
9920 DATA "Noise Freq.","Mixer Settng","Volume A","Volume B", "Volume C"
9930 DATA "Env Freq Fine", "Env Freq Rough", "Env Shape"
```

Index