# CML-12SDP256

Development Board for Motorola / Freescale
MC9S12DP256

# CONTENTS

# GETTING STARTED

The Axiom CML-12SDP256 single board computer is a fully assembled, fully functional development system for the Freescale MC9S12DP256 microcontroller. Provided with wall plug power supply and serial cable.  Support software for this development board is provided for Windows 95/98/NT/2000/XP operating systems.

This development board applies option selection jumpers.  Terminology for application of the option jumpers is as follows:

> Jumper on, in, or installed = jumper is a plastic shunt that fits across 2 pins and the shunt is installed so that the 2 pins are connected with the shunt.

> Jumper off, out, or idle = jumper or shunt is installed so that only 1 pin holds the shunt, no 2 pins are connected, or jumper is removed.  It is recommended that the jumpers be idled by installing on 1 pin so they will not be lost.

Development board users should also be familiar with the hardware and software operation of the target HCS12 device, refer to the Freescale User Guide for the device and the CPU12 Reference Manual for details.  The development board purpose is to assist the user in quickly developing an application with a known working environment or to provide an evaluation platform for the target HCS12.  Users should be familiar with memory mapping, memory types, and embedded software design for the fastest successful application development.

Application development maybe performed by applying the embedded MON12 (**default**) or NOICE firmware monitors, or by applying a BDM cable with supporting host software. The MON12 monitor provides an effective debug method for assembly level software, but has limitations in C code developments.  For C/C++ code development it is recommended that source code or symbolic debug capability be provided in the debugging environment.  The NOICE monitor or BDM interface with supporting software tools should be applied for C/C++ code development so the host PC can provide the symbolic support needed.  User should verify the NOICE or BDM development environment supports the C compiler to be applied, not all development environments support all compilers.

The MON12 and NOICE monitors are provided in the development board HCS12 internal flash memory and apply some HCS12 resources for operation.  See the respective chapter for each monitor for details on operation and resources applied. User should note both monitors apply operation of the HCS12 expanded wide mode data and address bus on HCS12 I/O ports A, B, E, and K for access to the external Ram.  The external ram provides a development memory where code to be debugged can be loaded or modified quickly and software breakpoints applied.  After the application is tested, the code can be relocated to the internal flash memory space of the HCS12 and programmed into the flash memory for dedicated operation.

User applications developed by applying MON12 or NOICE monitors can be modified and relocated for operation as a stand-alone application.  By applying the MON12 Autostart feature, the user application will operate from Reset or Power on conditions to provide a dedicated operation of the application. See the Autostart section in this manual for more information.

Follow the steps in this section to get started quickly and verify everything is working correctly.

# Installing the Software

1. Insert the Axiom 68HC12 support CD in your PC.  If the setup program does not start, run the file called "SETUP.EXE" on the disk.

2. Follow the instructions on screen to install the support software onto your PC.
   You should at minimum install the AxIDE for Windows software.

3. The programming utility "AxIDE" requires you to specify your board.  You should select "CML12SDP256" version of your development board.

# Board Startup

Follow these steps to connect and power on the board for the default Monitor operation.  This assumes you're using the provided AxIDE utility (installed in the previous section) or a similar communications terminal program on your PC.  If you're using a different terminal program than the one provided, set it's parameters to 9600 baud, N,8,1.

1. Set the CML12Sxxx board Option jumpers to default positions:

   **MEM-EN = IN, ECS = IN, JP1 = IN, NOAUTO (SPARE) = IN**

   **MODC = Out, JP2 = do not care, see COM Ports.**

2. Connect one end of the supplied 9-pin serial cable to an available serial COM port on your PC.  Connect the other end of the cable to the P-COM port on the CML12Sxxx board.

3. Apply power to the board by plugging in the power adapter that came with the system.

4. If everything is working properly, you should see a message to "**PRESS KEY TO START MONITOR…**" in your terminal window.  Press the ENTER key and you should see:

```
Axiom MON12 - HC12 Monitor / Debugger V256.x
Type "Help" for commands…

> _
```

5. Your board is now ready to use!  If you do not see this message prompt, or if the text is garbage, see the **TROUBLESHOOTING** section at the end of this manual.

# Support Software

There are many programs and documents on the included HC12 support CD you can use with the CML12Sxxx board.   You should install what you want from the main menu then browse the disk and copy what you like to your hard drive.

At minimum, you should install the AxIDE program.  This provides the flash programming utility and communication with the board via the COM port and the supplied serial cable.  This program includes a simple terminal for interfacing with other programs running on the CML12Sxxx and information from your own programs that send output to the serial port.

Also on the disk are free assemblers AS12 and MCU-EZ, the open source GNU C/C++ compiler tools for HC11/12, example source code, and other useful software.  The introductory tutorial in this manual uses the free AS12 assembler integrated into the AxIDE program.  This is a simple assembler with limited capability.  For a more powerful assembly tool, install the Freescale MCUez program from the CD.  This will allow you to use PAGED program memory in your application.

## Software Development

Software development on the CML12Sxxx can be performed using either the MON12 monitor installed in internal FLASH of the MCU, a third party debugger (Debug12, NoICE, CodeWarrior, etc.) or a Background Debug Module (BDM) connected to the BDM PORT connector.  Any of these tools can be used to assist in creating and debugging your program stored in RAM (see **Memory Map**).

After satisfactory operation running under a debugger, your program can be written to Internal Flash Memory using the included programming utilities.  The Mon12 firmware in the MCU flash provides the interrupt vectors in Ram memory and an Autostart feature to launch your application.  Your program may then run automatically whenever the board is powered on or RESET is applied.

# TUTORIAL

This section was written to help you get started developing software with the CML12SXXX board.  Be sure to read the rest of this manual as well as the documentation on the disk if you need further information.

The following sections take you through the complete development cycle of a simple "hello world" program, which sends the string "Hello World" to the serial port.

## Creating Source Code

You can write source code for the CML12SXXX board using any language that compiles to Freescale 68HC12 instructions.  Included on the software disk is a free Assembler, AS12.

You can write your source code using any ASCII text editor. You can use the free EDIT, WordPad, or Notepad programs that come with your computer. Note that the source file must be simple ASCII text without any document formatting added. Once your source code is written and saved to a file, you can assemble or compile it to a Freescale S-Record (hex) format.  This type of output file usually has a .MOT, .HEX or .S19 file extension and is in a format that can be read by the programming utilities and programmed into the CML12SXXX board.

It is important to understand the development board's use of Memory and Addressing when writing source code so you can locate your code at valid addresses.  For example, when in debug mode, you should put your program CODE in External RAM.  In assembly language, you locate the code with ORG statements in your source code.  Any lines following an ORG statement will begin at that ORG location, which is the first number following the word ORG, for example: `ORG $4000`. You must start your DATA (or variables) in a RAM location unused by your program, for example: `ORG $1000`.

In "debug mode" you'll be using a debugger utility (Mon12, NoICE, etc) which will handle initialization, interrupts vectors (reset, timers, etc), and the STACK.  When finished debugging, you must add code to your application to handle the initialization of the CPU, STACK and possibly the Interrupt vectors.  Some initialization is required to set the bus frequency, bus mode, internal EEprom and Flash memory programming clock rates, and others, see the CML12.ASM file for a sample. Set the stack at the top of your available internal RAM below the Ram interrupt vector table, for example $3F80, in assembly this would be `LDS #$3F80`.  Also install the RESET vector address in the Auto Start area, see the chapter in this manual.

If you are applying a software development tool that also provides a BDM cable interface to the board, the monitor installed in the flash is not required.  The BDM software tools may have the capability to erase and program the flash memory.  If this is the case, you may develop code in the external ram memory or internal flash without applying the monitor resources.  The MON12 S record is provided on the support CD to program into the flash if desired.   The BDM will allow locating programs in memory and applying resources reserved for the monitors.

A look at the example programs on the disk can make all of this clearer.  If you're using a compiler instead of an assembler, consult the compiler documentation for methods used to locate (MAP) your code, data and stack.


# Assembling source code

An example program called "HELLO.ASM" is provided under the `\EXAMPLES\CML12` directory of the CD and if you installed AxIDE, under that programs `\EXAMPLE` directory.  You must use the example for the MCU type installed on the CML12Sxxx board. For example use the CML-12SDP256 example on the DP256 version board.

You can assemble source code by using the AxIDE "BUILD" button or command line tools under a DOS prompt by typing:

```
AS12 HELLO.ASM –L HELLO
```

Most compilers and assemblers allow many command line options so using a MAKE utility or batch file is recommended if you use this method.  Run AS12 without any arguments to see all the options, or see the AS12.TXT file on the disk.

The utility software, AxIDE, provided with this board contains a simple interface to this assembler.  Use it by selecting "Build" from its menu.  This will prompt you for the file to be assembled.  **NOTE**:  You must select your board from the pull down menu first, or it may not build correctly.

**DO NOT** use long path or file names (> 8 characters).  The free assembler is an older DOS based tool that does not recognize them.

If there are no fatal errors in your source code, 2 output files will be created:

> **HELLO.S19**        a Freescale S-Record file that can be loaded or programmed into memory
>
> **HELLO.LST**        a common listing file which provides physical address information with resulting opcode and operand information. Warnings and error messages are provided with a summary at the end of the file.

The listing file is especially helpful to look at when debugging your program.  If your program has errors, they will be displayed in the listing or fatal errors will prevent output from being generated.  The end of the listing file generally provides a count of errors or warnings in the file.

If you prefer a windows integrated programming environment, try the Freescale MCU-EZ tools. Refer to the MCU-EZ documentation on the disk for more information.

Also, a port for the free GNU C compiler and tools for the HC12 is available on the CD under \Shareware and also online at  www.gnu-m68hc11.org.  Note that this version does not support HC12 Paging operation, check the web site for updates.

# Running your application

After creating a Freescale S-Record file you can "upload" it to the development board for a test run.  The provided example "HELLO.ASM" was created to run from external RAM so you can use the MON12 Monitor to test it without programming it into Flash.

If you haven't done so already, verify that the CML12Sxxx board is connected and operating properly by following the steps under "GETTING STARTED" until you see the Mon12 prompt, then follow these steps to run your program:

1.  Press and release the RESET button on the CML12Sxxx board.  You should see the PRESS ANY KEY message.  Hit the return key ↵ to get the monitor prompt.

2.  Type **LOAD**  ↵
    This will prepare Mon12 to receive a program.

3.  Select Upload and when prompted for a file name select your assembled program file in s-record format that was created in the previous section called:    **HELLO.S19**
    Your program will be sent to the board through the serial port.

4.  When finished loading you will see a done message and the > prompt again.  Type  **GO 4000** ↵
    This tells MON12 to execute the program at address $4000 hex, which is the start of our test program.

5.  If everything is working properly you should see the message "Hello World" echoed back to your terminal screen.  Press RESET to return to the monitor.

6.  If you do not get this message, see the **TROUBLESHOOTING** section in this manual

You can modify the hello program to display other strings or do anything you want. The procedures for assembling your code, uploading it to the board and executing it remain the same. MON12 has many features such as breakpoints, memory dump and modify and simple program trace (no redirect of the PC is followed). Type HELP at the MON12 prompt for a listing of commands or consult the Mon12 documentation on the disk for more information.

For a more powerful debugger with many advanced features such as source level debugging, you can use the NoICE debugger software. A full-featured demo version is provided on the CD, which you can use to get started. **NOTE**: To use this program instead of MON12 you must set the Autostart, see the NOICE chapter for details.

# Programming HCS12 Flash EEprom

After debugging, you can program your application into Flash Memory so it executes automatically when you apply power to the board as follows:

1. Make a backup copy of HELLO.ASM then use a text editor to modify it.

2. Remove the comment ';' character before one of the following lines to initialize the stack pointer which is necessary when running outside of a debugger:

    ```
    LDS     #$3F80   ; initialize stack location…
    ```

3. Re-Assemble HELLO.ASM as described in the "Assembling Source Code" section.

4. Select **Program** from the AxIDE menu and follow the message prompts. When prompted for a file name, enter the new HELLO.S19 file.

5. Press the RESET button on the board before clicking OK. When prompted to Erase, choose Yes.

6. When finished programming, Reset the board to get the Monitor prompt again. Use the monitor **AUTO** command to set the Autostart Reset vector:

    ```
    >AUTO 4000 ↵

      AutoStart ON, effective address = 4000

    >
    ```

7.  Verify NOAUTO option jumper is not installed. (Spare jumper on revision C boards)

8. RESET or re-apply Power to the board. Your new program should start automatically and the "Hello World" prompt should be displayed in the terminal window.

To return to the MON12 monitor program, install the NOAUTO option jumper then press RESET. Execute the monitor command NOAUTO to disable the Autostart and allow removal of the NOAUTO option jumper for normal operation.

# MON12 OPERATION

Mon12 is an embedded monitor / debug utility that allows loading a compiled software program (S record) into Ram memory for testing and debug.  The monitor may control the execution of the software by applying the SWI software interrupt service.  Other features allow memory and register examination or modification.

Communication with the monitor is provided on the HCS12 SCI0 serial port or COM port on the development board.  Default settings are 9600 baud with 8/n/1 bit settings.  Flow control is not provided so the host PC communication software should be set to None or Hardware flow control.  AxIDE utility software is recommended for use on a windows based host PC.

The monitor relies on resources from the HCS12 target to provide the monitor environment. The resources include 16K bytes of flash memory and 512 bytes of internal ram memory.  The user must respect the monitor's memory map when applying the monitor to help debug code. Restricted memory areas:

**Monitor Program space:  0xC000 - 0xFFFF Flash or Flash Page $3F.**

**Monitor Data space:        0x3E00 - 0x3FFF, Internal Ram.**

**Monitor Console:             COM Port and SCI0.**

**Monitor Autostart:          0xFEC - 0xFEF, Internal EEprom.**

Monitor application provides for redirection of interrupt vectors through the ram based interrupt table, initialization of SCI0 serial port, initialization of HCS12 flash and EEprom programming clock rates, initialization of 8 MHz E clock from 4Mhz reference crystal, and detection of auto start enabled operation.  The HCS12 memory map is fixed under monitor operation.

The monitor provides for interrupt vectors in the monitor data space from 0x3F8A - 0x3FFD. The vectors are in the same order as the default hardware table for the HCS12 located at address 0xFF8A - 0XFFFD (see table).  The Reset vector is reserved, user should apply Auto Start for application starting from Reset.

**MON12 operation notes:**

1. CML-12SDP256 monitor application configures target HCS12 for 8MHz E clock, lower flash block (page $3E) disabled from memory map, and external access clock stretch set to 3 cycles.   User can increase clock speed in application by modifying PLL control and setting new baud rate for serial port.  Defaults will return whenever monitor is Reset.

2. Mon12 will not trace into interrupts.  To trace an interrupt service set a breakpoint in the service routine and then trace.

3. Mon12 trace is limited to expecting the next linear address.  Program counter modification, branches, calls, or subroutines will not trace correctly.

4. Monitor start-up procedure:

   A)  Determine if Auto Start in enabled and proceed to vector if not a value of $FFFF.

B)  Set Stack, Initialize memory map and SCI0 port and send prompt.

C)  Receive first character from Console port and execute monitor if ASCII text / command, else start utility mode for programming services.

# Mon12 Monitor Commands

| `AUTO [<Address>]` | Enable Auto start, address is the vector |
| --- | --- |
| `NOAUTO` | Disable Auto start |
| `BF <StartAddress> <EndAddress> [<data>]` | Fill memory with data |
| `BR [<Address>]` | Set/Display user breakpoints |
| `BULK` | Erase entire on-chip EEPROM contents |
| `CALL [<Address>]` | Call user subroutine at <Address> |
| `G [<Address>]` | Begin/continue execution of user code |
| `HELP` | Display the Mon12 command summary |
| `LOAD [P]` | Load S-Records into memory, P = Paged S2 |
| `MD <StartAddress> [<EndAddress>]` | Memory Display Bytes |
| `MM <Address>` | Modify Memory Bytes (8 bit values) |
| `MW <Address>` | Modify memory Words (16 bit values) |
| `MOVE <StartAddress> <EndAddress> <DestAddress>` | Move a block of memory |
| `RD` | Display all CPU registers |
| `RM` | Modify CPU Register Contents |
| `STOPAT <Address>` | Trace until address |
| `T  [<count>]` | Trace <count> instructions |

# MON12 Interrupt Support

All interrupt services under MON12 are provided through the ram vector table, **see Table 2**. Each location in the table is initialized to a value of $0000 to cause the trap of an unscheduled interrupt.  Any nonzero value will allow the interrupt to proceed to the user's service routine that should be located at the provided address value.  Interrupt service delay is plus 21 cycles over standard interrupt service.

To use vectors specified in the table, the user must insert the address of the interrupt service routine during software initialization into the ram interrupt table. For an example, for the IRQ vector, the following is performed:

Example:      IRQ Service routine label = IRQ_SRV
              Ram Vector Table address is defined in table below, IRQ vector definition:
              VIRQ  EQU $3FF2  ; define ram table vector location

Place IRQ service routine address in the table:
              MOVW          #IRQ_SRV,VIRQ

This vector initialization should remain after debug when auto start will be applied for launching the user's application.  Note that the user interrupt service routines must be located in the $4000 - $7FFF address range for correct operation.  See Autostart for more details.

**10**

# MON12 and NOICE Memory Map

| ADDRESS | TYPE MEMORY | MEMORY APPLICATION |
|---------|-------------|--------------------|
| $C000 - $FFFF | FLASH | MON12, NOICE, and Utility firmware located in internal flash, Page $3F. |
| $8000 - $BFFF | External Ram | User Paged Program Memory space, pages $20 - $2E.<br><br>Note: Pages $30 - $3F reside in the internal flash. |
| $4000 - $7FFF | External Ram | User Program Memory, emulate fixed page $3E. |
| $3F8C - $3FFD | Internal Ram | Ram Interrupt Vector Table |
| $3E00 - $3F8B | Internal Ram | Monitor reserved ram memory. Stacks and variables. |
| $1000 - $3DFF | Internal Ram | User Internal Ram memory |
| $0400 - $0FEB | Internal EEprom | User EEprom memory, Monitor reserves $FEC - $FEF for Autostart, user should avoid $FF0 - $FFF memory use. |
| $0000 - $03FF | HCS12 Registers | Monitor or user access to control registers. |

# NOICE OPERATION

NOICE is a development software provided by www.NOICEdebugger.com.  NOICE provides a development environment that is supported by the NOICE host PC software.   This development environment has the capability to provide symbolic debug for C source codes and compilers for a low cost.  A fully functional software version is available on the support CD that will operate in demonstration mode. The user should register the software and download the latest version from the above web site to get full support.  See the NOICE documentation for details.

The CML-12SDP256 provides the NOICE debug monitor kernel as a subset of the MON12 monitor in reserved flash memory.  User may apply the NOICE development system by setting the MON12 Autostart for the $F800 vector, reset the board and launch the NOICE host software on the PC.  The NOICE monitor kernel applies the same resources, memory map, and ram interrupt table as the MON12 monitor.   NOICE operation notes:

**Baud Rate** = 19.2K baud 8/n/1

**E clock frequency** = 24MHz

# BDM OPERATION

The CML-12SDP256 board will emulate supported HC12 device internal flash memory in external ram.  This feature allows BDM (Background Debug Modules) such as the AX-BDM12 to load and control the execution of code being developed without the necessity of the internal flash memory being programmed many times during the development process.  This feature improves updating time and allows the use of may software breakpoints instead of being limited to only 2 hardware breakpoints.

Operation Notes for BDM use:

1)      CML-12SDP256 **MODC Option Jumper** should be installed if a BDM is connected to the BDM Port.  Default Mode is single-chip so the MODC option installed will force Special Single-chip Mode on Reset.

2)      The BDM initialization of the HC12 should set the correct operating MODE (Expanded Wide for memory access).  The EME, EMK, LSTRB, RW, ROMEN and Stretch configuration bits should be set for proper external memory access operation.  The Axiom support CD contains sample set-up macros for the AX-BDM12.

3)      While using the BDM, the user has full control over the memory map and hardware resources of the HCS12.  The no resources are required to be reserved for monitor use and the user can apply the actual HCS12 interrupt vector table located at 0xFF8C - 0xFFFF.

# AUTOSTART

The MON12 Monitor allows an Autostart operation to launch user applications programmed into the HCS12 internal flash fixed page ($3E) addresses 0x4000 - 0x7FFF from Reset.  The Autostart mask and vector are stored in the nonvolatile internal EEprom at addresses $FEC - $FEF.  The monitor provides special commands, **AUTO** and **NOAUTO**, to enable and disable the Autostart on the next Reset sequence.  After an Autostart is enabled with a valid user vector, user application code will be started after Reset instead of the monitor or utility programs.  To recover monitor operation after Autostart has been enabled, the **NOAUTO** (**Spare option on Revision C boards**) option jumper can be installed or a low level applied to the XIRQ signal and Reset applied.

User application must perform all initialization including Stack setting, hardware startup, and external memory bus enable if needed, when the Autostart is applied.  MON12 Ram Interrupt Vector table must also be applied in the same manner as under MON12 supervision or application interrupts will be trapped instead of serviced.  See the CML12S.asm file for sample start-up initialization code.

Developing an application under MON12 or NOICE for Autostart should follow these steps:

1) Follow the MON12/ NOICE memory map and apply startup initialization and interrupt service routines in the 0x4000 - 0x7FFF memory area.

2) After development by applying ram memory program pages $20 - $2D, user should relocate the paged program code to internal flash pages $30 - $3D for programming into the flash memory.   The user code in memory area 0x4000 - 0x7FFF will translate to the lower fixed flash page $3E for programming operations.  User variables and stack, as well as interrupt vectors should stay in the internal ram area 0x1000 - 0x3E80 (monitor stack and variables not needed).  Then set the Autostart vector for application launching.

3) If the Autostart application fails to start after programming, user should review all initialization and memory mapping first.  Make sure the NOAUTO (or Spare) Option jumper is idle or open.  If the application applies the XIRQ interrupt, the interrupt must be idle (high level) during any Reset sequence.  Hardware may need to be applied if XIRQ signal level cannot be guaranteed high during Reset.

4) To perform a test Autostart and apply the external ram for program space the following precaution should be observed:

> Expanded Wide Mode bus operation must be enabled from internal Ram space before access to the external ram can be performed.  Use CML12S.asm file for an example and locate the PEAR/MODE Register write in internal ram space 0x1000 - 0x3F80. Program pages $20 - $2D should be applied.  Code must be loaded and tested without powering down the development board (use Reset Switch).

# OPTIONS and JUMPERS

## MEM_EN

The MEM_EN option jumper is installed by default and enables the external ram memory on the expanded HCS12 address and data bus.  Removing the MEM_CS option jumper will allow single-chip I/O port type operation of HCS12 ports A, B, E, and K (no bus enabled) without external memory interference.  Note that the default monitor will enable the Expanded Bus no matter the MEM_EN option setting.   User can defeat this operation by installing a simple program patch to be AUTO started:

```
          org     $4000
START_SS:  jmp     $FC25   ; back to the monitor, no bus enabled for single-chip mode
```

## ECS

The ECS option installed enables the Emulation Chip Select signal from the HCS12 to drive the upper address lines from HCS12 Port K to the external Ram on the CML12S board.  With the option open or idle, only the linear 64K byte address map is available on the external address / data bus.  ECS installed is required to emulate flash program pages in the external ram memory.

## MODC

The MODC option jumper provides Special Mode enable during Reset.  This option must be open or idle when operating with the MON12 or NOICE monitors. If a BDM cable is applied to BDM port, the MODC option must be installed to enable Special Mode.  Failure to install the MODC jumper during BDM application may cause communication problems with the host.

# NOAUTO (Spare)

The NOAUTO (Spare on REV C. board) option jumper installed defeats the Autostart operation so the MON12 monitor will provide a command prompt. The jumper applies a ground potential to the XIRQ* interrupt line. User should only install the jumper to restore monitor operation, perform the MON12 NOAUTO command to disable the Autostart, and remove or idle the jumper. MON12 operation will then be provided on subsequent Reset conditions.

# MODE

The MODE option jumper is not installed on the CML12S-DP256 board and is hard connected by circuit copper trace for Single-chip Mode operation of the CPU. Both the MODA and MODB signals are terminated by this option. Due to the restriction that the HCS12 internal flash memory is the only nonvolatile program memory provided on the board, single Chip Mode is default. All other Modes can be enabled under software control from this mode of operation.

The MODE option jumper may be installed by the user by cutting the hard trace and applying 2 header pins with a shunt jumper. With the shunt jumper removed, the Reset mode will then be Normal Expanded Wide. (Note: mask set 1K79X and earlier will not fetch the Reset vector from external memory in this mode).

# OSC_SEL

The OSC_SEL option jumper is not installed on the CML12S-DP256 board. The default configuration is for the provided 4MHz reference crystal to provide the HCS12 oscillator. If the user requires an external clock to be applied, two header pins and shunt jumper can be applied to select the alternate clock source. User should refer to the HCS12 User Guide and CML12S board schematic for proper application of the external clock.

# ROM_OFF

The ROM_OFF option jumper is not installed on the CML12S-DP256 board. The default configuration is that the internal flash memory of the HCS12 is enabled at Reset. The user must add external nonvolatile memory to the CML12S board to take advantage of this option. If the external memory is applied, the user may install the two header pins and shunt jumper to select internal or external memory use form Reset.

# JP1 and JP2

JP1 and 2 option jumpers provide an easy method of connecting or isolating the HCS12 SCI0 and SCI1 serial channel RXD pins respectfully from the provided on-board RS232 transceiver. To apply the RXD pins on the SCI channels for other user applications requires that the transceiver driver be removed from the HCS12 pin. User may then apply signals to the respective pins at the MCU PORT connector without driver conflict. Please note that the on-board monitor(s) require HCS12 SCI channel 0 (JP1 installed) for user interface.

# CUT-AWAY OPTIONS 1 - 6

CUT-AWAY options allow the user to disconnect dedicated HCS12 I/O port resources from development board peripherals.  The CUT-AWAY options also allow for establishing the connection again by installing surface mount 1206 size 0 ohm resistors or mod wire with the use of a soldering iron.  Normal operation of the development board generally does not require any manipulation of the CUT_AWAY options.

**#1 Cut-Away:** HCS12 Port S5/MOSI signal to the LCD_PORT shift register.

**#2 Cut-Away:** HCS12 Port S7/SS0 signal to the LCD_PORT shift register.

**#3 Cut-Away:** HCS12 Port S6/SCK signal to the LCD_PORT shift register.

**#4 Cut-Away:** HCS12 Oscillator Crystal ground, if another crystal is applied by the user this connection may require a capacitor to be installed. Refer to the HCS12 CGM module information.

**#5 Cut-Away:** HCS12 Port M0/CAN_RXD0 signal to the CAN port transceiver.

**#6 Cut-Away:** CAN Port Transceiver enable connection to ground.  This connection enables the CAN Port transceiver output to the CAN bus at all times.  If the user wants to apply output enable or slew rate control to the transceiver, this option should be cut and 1206 size resistor applied for slew rate or a HCS12 I/O port applied for output enable control.  See the PCA82C250 data sheet for application information.

# PORTS AND CONNECTORS

## TB1 and J1 Power

The TB1 and J1 connectors provide power input to the board or if J1 is used for input, TB1 maybe used to source additional circuitry.  The J1 power jack accepts a standard 2.0 ~ 2.1mm center barrel plug connector (positive voltage center) to provide the +VIN supply of +7 to +20 VDC @ 80ma minimum (+9VDC nominal).  TB1 provides access to the +VIN, GND (power ground), HCS12 core VDD, and +5V power supplies.  The CML12Sxxx power supply will provide 50ma of +5V for user application.  +VIN input power should only be applied by J1 or TB1, not both or a supply conflict may occur and the CML12Sxxx board could be damaged.  The VDD supply is for reference or external 2.5V input only and should not be loaded by external circuitry or damage to the HCS12 device may occur.

# MCU_PORT

| | | | |
|---:|:---:|:---:|:---|
| GND | **60** | **59** | +5V |
| PT7 | **58** | **57** | PT6 |
| PT5 | **56** | **55** | PT4 |
| PT3 | **54** | **53** | PT2 |
| PT1 | **52** | **51** | PT0 |
| ** PK0 | **50** | **49** | PK1 ** |
| ** PK2 | **48** | **47** | PK3 ** |
| ** PK4 | **46** | **45** | PK5 ** |
| GND | **44** | **43** | +5V |
| PP1 | **42** | **41** | PP0 |
| PP3 | **40** | **39** | PP2 |
| PP5 | **38** | **37** | PP4 |
| PP7 | **36** | **35** | PP6 |
| ** PM1 | **34** | **33** | PM0 ** |
| PM3 | **32** | **31** | PM2 |
| PM5 | **30** | **29** | PM4 |
| PM7 | **28** | **27** | PM6 |
| PJ1 | **26** | **25** | PJ0 |
| PJ7 | **24** | **23** | PJ6 |
| ** PS7 | **22** | **21** | PS6 ** |
| ** PS5 | **20** | **19** | PS4 ** |
| ** PS3 | **18** | **17** | PS2 ** |
| ** PS1 | **16** | **15** | PS0 ** |
| GND | **14** | **13** | +5V |
| GND | **12** | **11** | VREGEN |
| ** PB7/D7 | **10** | **9** | PB6/D6 ** |
| ** PB5/D5 | **8** | **7** | PB4/D4 ** |
| ** PB3/D3 | **6** | **5** | PB2/D2 ** |
| ** PB1/D1 | **4** | **3** | PB0/D0 ** |
| GND | **2** | **1** | +5V |

The **MCU_PORT** provides access to the peripheral features and I/O lines of the HCS12.

** Note signals with alternate connections on the development board:

PB0 - 7 [D0 - 7] provide address / data on the expanded HCS12.

PK0 - 5 [XA14 - XA19] provide high order paged address lines on the expanded HCS12.

PM0 - 1 [CAN RXD0, TXD0] CAN channel  0 to CAN Port transceiver.

PS0 - 1 [COM Port RXD0, TXD0]

PS2 - 3 [JP3 Port RXD1, TXD2]

PS4 - 7 [SPI Port] provides LCD_PORT serial interface.

# ANALOG PORT

| | | | |
|---:|:---:|:---:|:---|
| PAD0/AN0 | **1** | **2** | PAD8/AN8 |
| PAD1/AN1 | **3** | **4** | PAD9/AN9 |
| PAD2/AN2 | **5** | **6** | PAD10/AN10 |
| PAD3/AN3 | **7** | **8** | PAD11/AN11 |
| PAD4/AN4 | **9** | **10** | PAD12/AN12 |
| PAD5/AN5 | **11** | **12** | PAD13/AN13 |
| PAD6/AN6 | **13** | **14** | PAD14/AN14 |
| PAD7/AN7 | **15** | **16** | PAD15/AN15 |
| VRH | **17** | **18** | VRL |
| VDDA | **19** | **20** | GND |

The **ANALOG** port provides access to the Port AD0 and Port AD1 Analog-to-Digital input lines.

**PAD0 – PAD15**  HC12 Port AD0-15 is an input port or AN0 - AN15 A/D Converter inputs.

**VRH / VRL**  HC12 A/D Converter Reference Pins. See HCS12 A/D User guide.  To provide an external reference voltage, R3 and R4 need to be removed to apply external VRH or VRL respectfully.  See schematic.

# BUS_PORT

| | | | |
|---|---|---|---|
| GND | **1** | **2** | D11/PA3 |
| PA2/D10 | **3** | **4** | D12/PA4 |
| PA1/D9 | **5** | **6** | D13/PA5 |
| PA0/D8 | **7** | **8** | D14/PA6 |
| A0 | **9** | **10** | D15/PA7 |
| A1 | **11** | **12** | A2 |
| A10 | **13** | **14** | A3 |
| OE* | **15** | **16** | A4 |
| A11 | **17** | **18** | A5 |
| A9 | **19** | **20** | A6 |
| A8 | **21** | **22** | A7 |
| A12 | **23** | **24** | A13 |
| WE* | **25** | **26** | A14 |
| PE3/LSTRB* | **27** | **28** | A15 |
| PE5/MODA | **29** | **30** | PE7/NOACC ** |
| PE6/MODB | **31** | **32** | PE1/IRQ* |
| +5V | **33** | **34** | PE0/ XIRQ* |
| PE2/RW | **35** | **36** | RESERVED |
| PE4/ECLK | **37** | **38** | RESERVED |
| GND | **39** | **40** | RESET* |

The **BUS_PORT** supports off-board memory devices while the HCS12 is in expanded mode.

**PA0/D8 - PA7/D15** High Byte Data Bus in Wide Expanded Mode.  Port A in Single Chip Mode.

**A0 – A15**  Latched Memory Addresses 0 to 15.

**OE***  Memory Output Enable signal, Active Low.  Valid with ECLK and R/W high.

**WE***  Memory Write Enable signal, Active Low.  Valid with ECLK high and R/W low.

**RESET***  HCS12 active low RESET signal.

# KEYPAD / PORT H

The KEYPAD / PORT H connector provides interface for the HCS12 port H or applying a keypad such as the Axiom Mfg. HC-KP. When applied as a KEYPAD connector, the interface is for a passive 4 x 4 matrix (16 key) keypad device.

| | |
|---|---|
| **1** | PH0 |
| **2** | PH1 |
| **3** | PH2 |
| **4** | PH3 |
| **5** | PH4 |
| **6** | PH5 |
| **7** | PH6 |
| **8** | PH7 |

This interface is implemented as a software key scan.  Pins PH0-3 are used as column drivers which are active high outputs. Pins PH4-7 are used for row input and will read high when their row is high.

See the file **Key12Dx.ASM** for an example program using this connector.

# P_COM1 and P_COM2

| | | | |
|---|---|---|---|
| 1 | **1** | | 6 |
| TXD0 | **2** | **6** | 7 |
| RXD0 | **3** | **7** | 8 |
| 4 | **4** | **8** | 9 |
| GND | **5** | **9** | X |

The **COM-1** port has a Female DB9 connector that interfaces to the HCS12 internal SCI0 serial port via the U11 RS232 transceiver.  It uses a simple 2 wire asynchronous serial interface and is translated to RS232 signaling levels.
**1,4,6 connected and 7,8 connected**

JP1 will isolate the SCI0 RXD pin from the transceiver.

The 1,4,6,7,8, and 9 pins provide RS232 flow control and status.  These are connected on the on the bottom of the development board to provide NULL status to the host.  User may isolate pins and provide flow control or status connection to the host by applying HCS12 I/O signals and **RS232 level conversion.**

# P_COM2

P_COM2 is a 3 pin header that provides the HCS12 SCI1 serial port translated to RS232 signal levels.  A solder cup DB9 style connector may be installed with wires and connector to apply this channel.  JP2 option will isolate the SCI RXD pin from the transceiver.

**P_COM2 pin connections:**

> Pin 1 = TXD
>
> Pin 2 = RXD
>
> Pin 3 = GND / common

# CAN PORT

This port provides a CAN Bus interface associated with HCS12 CAN channel 0.  The port has a CAN Transceiver (Philips PCA82C250) capable of up to 1M Baud data rate.  The user may isolate the HCS12 CAN channel 0 from the transceiver by CUT-AWAY option 5.

## CAN Port Connections

| 1 | GND | The CAN Port connector provides an interface to the MSCAN12 |
|---|-----|---|
| 2 | CAN-H | channel 0 in the HCS12 microcontroller. |
| 3 | CAN-L | |
| 4 | +5V | |

## CAN BUS TRANSMIT ENABLE

The CAN port transceiver transmit driver is enabled for maximum drive and minimum slew rate by default.  The drive and slew rate may be adjusted by cutting CUT-AWAY #6 and adding a 1206 size surface mount resistor,  see the PCA82C250 data sheet for more information.

CAN Bus transceiver transmit enable control can be applied to the port by the RS tie pad. The user should select an available HCS12 I/O port to perform the transmit enable function and connect it from the MCU_PORT pin to RS pad as required.  **The CUT_AWAY #6 must be open to apply transmit enable control.** The transmit enable signal to the CAN transceivers is active logic low.

## CAN BUS TERMINATION

The CAN port provides RC11,12, and 13 1206 SMT size termination resistors on the bottom of the CML12Sxxx board that are not installed at the factory.  The termination resistors provide optional bias and termination impedance for the CAN bus connected to the CAN Port.  Type of wire media, data rate, length of wire, and number of CAN bus nodes can all effect the requirement or value of the termination for the CAN bus.  User should refer to particular application for termination requirements.

**RC11 CAN-H Bias Resistor:** Provides bias to ground potential.

**RC13 CAN-L Bias Resistor:** Provides bias to +5V potential.

**RC12 CAN Termination Resistor:** Provides end point termination between CAN-H and CAN-L signal.

# P1 - P4 HCS12 Header Ring

P1 - P4 provide a header ring for all I/O of the HCS12 device.  These connectors are not installed.  User should refer to the CML12S board schematic diagram for connector pin connections.  All HCS12 I/O is available from the other I/O Ports on the board.

# LCD_PORT

The LCD_PORT interface is connected to the HCS12 SPI-0 port and applies a serial shift register to convert the data to parallel interface for LCD input. This is required due to the fast timing characteristics of the HCS12 data bus and the slow timing of the standard LCD Modules. Example LCD Port assembly language driver software is provided on the support CD to demonstrate typical LCD module operation using this technique.

The interface supports all OPTREX™ DMC series and similar displays with up to 80 characters in 4 bit bus mode and provides the most common pinout for a dual row rear mounted display connector.   The LCD module VEE or contrast potential is 0 Volts on this board.  The LCD module type should be TN (Standard Twist) style and Reflective to support this VEE potential. The Axiom Mfg. HC-LCD is also compatible.  The LCD Module is configured in a Write only mode, it is not possible to read current cursor position or the busy status back from the module.

**LCD_PORT Connector**

| | | | | |
|---|---|---|---|---|
| +5V | 2 | 1 | GND | SPI data bit definitions to LCD Port: |
| RS | 4 | 3 | VEE-GND | D0 - D3 = DB4 - 7, LCD data |
| EN | 6 | 5 | R/W-GND | D4 - D5 = Spare pins S1 and S2, not connected |
| DB1 | 8 | 7 | DB0 | D6 = RS, 0 = LCD Command, 1 = LCD Data |
| DB3 | 10 | 9 | DB2 | D7 = EN, 1 = LCD enable. |
| DB5 | 12 | 11 | DB4 | DB0 -DB3 are not applied and have 10K pull-down |
| DB7 | 14 | 13 | DB6 | resistance. |

NOTES:

1) The LCD write requires 3 SPI transfers.  Transfer 1 provides data 0 - 3 and RS (register select) value.  Transfer 2 provides the same data with the EN (D7) bit set.  Transfer 3 provides same data with the EN bit clear.

2) Resistor R25 can be removed to apply and external VEE potential.

3) CUT-AWAY 1 - 3 provide a means to isolate the LCD Port from the HCS12 SPI channel.

# BDM PORT

The BDM port is a 6 pin header compatible with the Freescale Background Debug Mode (BDM) Pod.  This allows the connection of a background debugger for software development, programming and debugging in real-time without using HC12 I/O resources.

BGND | **1** | **2** | GND

**3** | **4** | /RESET

**5** | **6** | +5V

See the HC12 Technical Reference Manual for complete documentation of the BDM.

A Background Debug Module is available from the manufacturer.

# TEST POINTS

The following test points are provided on the development board:

**EXTAL :** HCS12 oscillator or external clock input pin.

**XTAL :** HCS12 oscillator output pin.

**XFC :** HCS12 PLL reference voltage and filter.

**VDDPLL :** HCS12 PLL voltage source test point.

# TROUBLESHOOTING

The CML12SXXX board is fully tested and operational before shipping.  If it fails to function properly, inspect the board for obvious physical damage first.  Ensure that all IC devices in sockets are properly seated.  Verify the communications setup as described under GETTING STARTED and see the **Tips and Suggestions** sections following for more information.

The most common problems are improperly configured communications parameters, and attempting to use the wrong COM port.

1.  Verify that your communications port is working by substituting a known good serial device or by doing a loop back diagnostic.

2.  Verify option jumpers JP1 is installed and MODC is open / idle.

3.  Verify Autostart is not enabled.  Install the NOAUTO option on both pins.

4.  Verify the power source.  You should measure a minimum of 9 volts between the GND and +VIN connections on the TB1 power connector with the standard power supply provided.

5.  If no voltage is found, verify the wall plug connections to AC wall plug outlet and the power connector.

6.  Verify the logic power source.  You should measure +5 volts between the GND and +5V connections on the TB1 power connector.  If the +VIN supply is good and this supply is not +5V, immediately disconnect power from the board.  Contact support@axman.com by email for instructions and provide board name and problem.

7.  Disconnect all external connections to the board except for COM1 to the PC and the wall plug.

8.  Make sure that the RESET line is not being held low.
    Check for this by measuring the RESET Signal on the BUS_PORT.

9.  Verify the presence of a 4MHz square wave at the EXTAL pin or 8MHz E clock signal if possible.

10. Contact support@axman.com by email for further assistance.  Provide board name and describe problem.

# Tips and Suggestions

Following are a number of tips, suggestions, and answers to common questions that will solve many problem users have with the CML12SXXX development system. You can download the latest software from the Support section of our web page at:

    www.axman.com

## Utilities

- If you're trying to program memory or start the utilities, make sure all jumpers are correct.

- Be certain that the data cable you're using is bi-directional and is connected securely to both the PC and the board. Also, make sure you are using the correct serial port.

- Make sure the correct power is supplied to the board. You should only use a 9 volt, 200mA minimum adapter or power supply. If you're using a power strip, make sure it is turned on.

- Make sure you load your code to an address space that actually exists. See the Memory Map if you're not sure. The MEM_EN and ECS options change the memory map.

- If debugging under Mon12, make sure you're not over-writing internal RAM used by it.

- If you're running in a multi-tasking environment (such as Windows™) close all programs in the background to be certain no serial conflict occurs.

## Code Execution

- Under Mon12, breakpoints may not be acknowledged if you use the CALL command. You should use one of the GO command instead.

- Check the Autostart mask and reset vector located in EEprom at 0xFEC - 0xFEF. These 2 words contain the enable mask and address where user application execution will begin when the unit is powered on.

- When running your code stand-alone, you must initialize ALL peripherals used by the micro, including the Stack, Serial Port, and pseudo Interrupt vectors etc.

- You must either reset the COP watchdog timer in the main loop of your code or disable it when not running under Mon12 or BDM mode. The micro may enable this by default and if you don't handle it your code will reset every few 100ms.

# TABLE 1:  LCD Command and Character Codes

Command codes are used for LCD setup and control of character and cursor position. All command codes are written to LCD panel address $B5F0.  The BUSY flag (bit 7) should be tested before any command updates to verify that any previous command is completed.  A read of the command address $B5F0 will return the BUSY flag status and the current display character location address.

| Command | Code | Delay |
|---|---|---|
| Clear Display, Cursor to Home | $01 | 1.65ms |
| Cursor to Home | $02 | 1.65ms |
| Entry Mode: | | |
|     Cursor Decrement, Shift off | $04 | 40us |
|     Cursor Decrement, Shift on | $05 | 40us |
|     Cursor Increment, Shift off | $06 | 40us |
|     Cursor Increment, Shift on | $07 | 40us |
| Display Control: | | |
|     Display, Cursor, and Cursor Blink off | $08 | 40us |
|     Display on, Cursor and Cursor Blink off | $0C | 40us |
|     Display and Cursor on, Cursor Blink off | $0E | 40us |
|     Display, Cursor, and Cursor Blink on | $0F | 40us |
| Cursor / Display Shift: (nondestructive move) | | |
|     Cursor shift left | $10 | 40us |
|     Cursor shift right | $14 | 40us |
|     Display shift left | $18 | 40us |
|     Display shift right | $1C | 40us |
| Display Function (default 2x40 size) | $3C | 40us |
| Character Generator Ram Address set | $40-$7F | 40us |
| Display Ram Address and set cursor location | $80- $FF | 40us |

# LCD Character Codes

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $20 | Space | $2D | - | $3A | : | $47 | G | $54 | T | $61 | A | $6E | n | $7B | { |
| $21 | ! | $2E | . | $3B | ; | $48 | H | $55 | U | $62 | B | $6F | o | $7C | \| |
| $22 | " | $2F | / | $3C | { | $49 | I | $56 | V | $63 | C | $70 | p | $7D | } |
| $23 | # | $30 | 0 | $3D | = | $4A | J | $57 | W | $64 | D | $71 | q | $7E | > |
| $24 | $ | $31 | 1 | $3E | } | $4B | K | $58 | X | $65 | E | $72 | r | $7F | < |
| $25 | % | $32 | 2 | $3F | ? | $4C | L | $59 | Y | $66 | F | $73 | s | | |
| $26 | & | $33 | 3 | $40 | Time | $4D | M | $5A | Z | $67 | G | $74 | t | | |
| $27 | ' | $34 | 4 | $41 | A | $4E | N | $5B | [ | $68 | H | $75 | u | | |
| $28 | ( | $35 | 5 | $42 | B | $4F | O | $5C | Yen | $69 | I | $76 | v | | |
| $29 | ) | $36 | 6 | $43 | C | $50 | P | $5D | ] | $6A | J | $77 | w | | |
| $2A | * | $37 | 7 | $44 | D | $51 | Q | $5E | ^ | $6B | K | $78 | x | | |
| $2B | + | $38 | 8 | $45 | E | $52 | R | $5F | _ | $6C | L | $79 | y | | |
| $2C | , | $39 | 9 | $46 | F | $53 | S | $60 | ` | $6D | M | $7A | z | | |

# TABLE 2: MON12 Service Routine Jump Table

**ADDRESS**
```
ff10        JMP     MAIN              ; warm start
ff13        JMP     BPCLR             ; clear breakpoint table
ff16        JMP     RPRINT            ; display user registers
ff19        JMP     HEXBIN            ; convert ascii hex char to binary
ff1c        JMP     BUFFARG           ; build hex argument from buffer
ff1f        JMP     TERMARG           ; read hex argument from terminal
ff22        JMP     CHGBYT            ; modify memory byte at address in x
ff25        JMP     CHGWORD           ; modify memory word at address in x
ff28        JMP     READBUFF          ; read character from buffer
ff2b        JMP     INCBUFF           ; increment buffer pointer
ff2e        JMP     DECBUFF           ; decrement buffer pointer
ff31        JMP     WSKIP             ; find non-whitespace char in buffer
ff34        JMP     CHKABRT           ; check for abort from terminal
ff37        JMP     UPCASE            ; convert to upper case
ff3a        JMP     WCHEK             ; check for white space
ff3d        JMP     DCHEK             ; check for delimeter
ff40        JMP     ONSCI0            ; initialize i/o device
ff43        JMP     INPUT             ; low level input routine
ff46        JMP     OUTPUT            ; low level output routine
ff49        JMP     OUTLHLF           ; display top 4 bits as hex digit
ff4c        JMP     OUTRHLF           ; display bottom 4 bits as hex digit
ff4f        JMP     OUTA              ; output ascii character in A
ff52        JMP     OUT1BYT           ; display the hex value of byte at X
ff55        JMP     OUT1BSP           ; out1byt followed by space
ff58        JMP     OUT2BSP           ; display 2 hex bytes (word) at x and a space
ff5b        JMP     OUTCRLF           ; carriage return, line feed to terminal
ff5e        JMP     OUTSTRG           ; display string at X (term with $04)
ff61        JMP     OUTSTRG0          ; outstrg with no initial carr ret
ff64        JMP     INCHAR            ; wait for and input a char from term
ff67        JMP     VECINIT           ; initialize RAM vector table
```

# TABLE 3: MON12 Interrupt Table

| MON12 Ram Interrupt Vector | HCS12 Interrupt Vector Address | MON12 TRAP code | Vector Source |
|---|---|---|---|
| 3F8C | FF8C | 02 | PWME |
| 3F8E | FF8E | 04 | PTPI |
| 3F90 | FF90 | 06 | C4TX |
| 3F92 | FF92 | 08 | C4RX |
| 3F94 | FF94 | 0A | C4ERR |
| 3F96 | FF96 | 0C | C4WU |
| 3F98 | FF98 | 0E | C3TX |
| 3F9A | FF9A | 10 | C3RX |
| 3F9C | FF9C | 12 | C3ERR |
| 3F9E | FF9E | 14 | C3WU |
| 3FA0 | FFA0 | 16 | C2TX |
| 3FA2 | FFA2 | 18 | C2RX |
| 3FA4 | FFA4 | 1A | C2ERR |
| 3FA6 | FFA6 | 1C | C2WU |
| 3FA8 | FFA8 | 1E | C1TX |
| 3FAA | FFAA | 20 | C1RX |
| 3FAC | FFAC | 22 | C1ERR |
| 3FAE | FFAE | 24 | C1WU |
| 3FB0 | FFB0 | 26 | C0TX |
| 3FB2 | FFB2 | 28 | C0RX |
| 3FB4 | FFB4 | 2A | C0ERR |
| 3FB6 | FFB6 | 2C | C0WU |
| 3FB8 | FFB8 | 2E | FEPRG |
| 3FBA | FFBA | 30 | EEPRG |
| 3FBC | FFBC | 32 | SPI2 |
| 3FBE | FFBE | 34 | SPI1 |
| 3FC0 | FFC0 | 36 | I2C |
| 3FC2 | FFC2 | 38 | BDLC |
| 3FC4 | FFC4 | 3A | CRGC |
| 3FC6 | FFC6 | 3C | CRGL |
| 3FC8 | FFC8 | 3E | PACBO |
| 3FCA | FFCA | 40 | MCNT |
| 3FCC | FFCC | 42 | PTHI |
| 3FCE | FFCE | 44 | PTJI |
| 3FD0 | FFD0 | 46 | ADC1 |
| 3FD2 | FFD2 | 48 | ADC0 |
| 3FD4 | FFD4 | 4A | SCI1 |
| 3FD6 | FFD6 | 4C | SCI0 |
| 3FD8 | FFD8 | 4E | SPI0 |
| 3FDA | FFDA | 50 | PACAI |
| 3FDC | FFDC | 52 | PACAO |
| 3FDE | FFDE | 54 | TOF |
| 3FE0 | FFE0 | 56 | TC7 |
| 3FE2 | FFE2 | 58 | TC6 |
| 3FE4 | FFE4 | 5A | TC5 |
| 3FE6 | FFE6 | 5C | TC4 |
| 3FE8 | FFE8 | 5E | TC3 |
| 3FEA | FFEA | 60 | TC2 |
| 3FEC | FFEC | 62 | TC1 |
| 3FEE | FFEE | 64 | TC0 |
| 3FF0 | FFF0 | 66 | RTI |
| 3FF2 | FFF2 | 68 | IRQ |
| 3FF4 | FFF4 | 6A | XIRQ |
| 3FF6 | FFF6 | 6C | SWI |
| 3FF8 | FFF8 | 6E | TRAP |
| 3FFA | FFFA | 70 | COP |
| 3FFC | FFFC | 72 | CLM |
|  | FFFE |  | RESET |