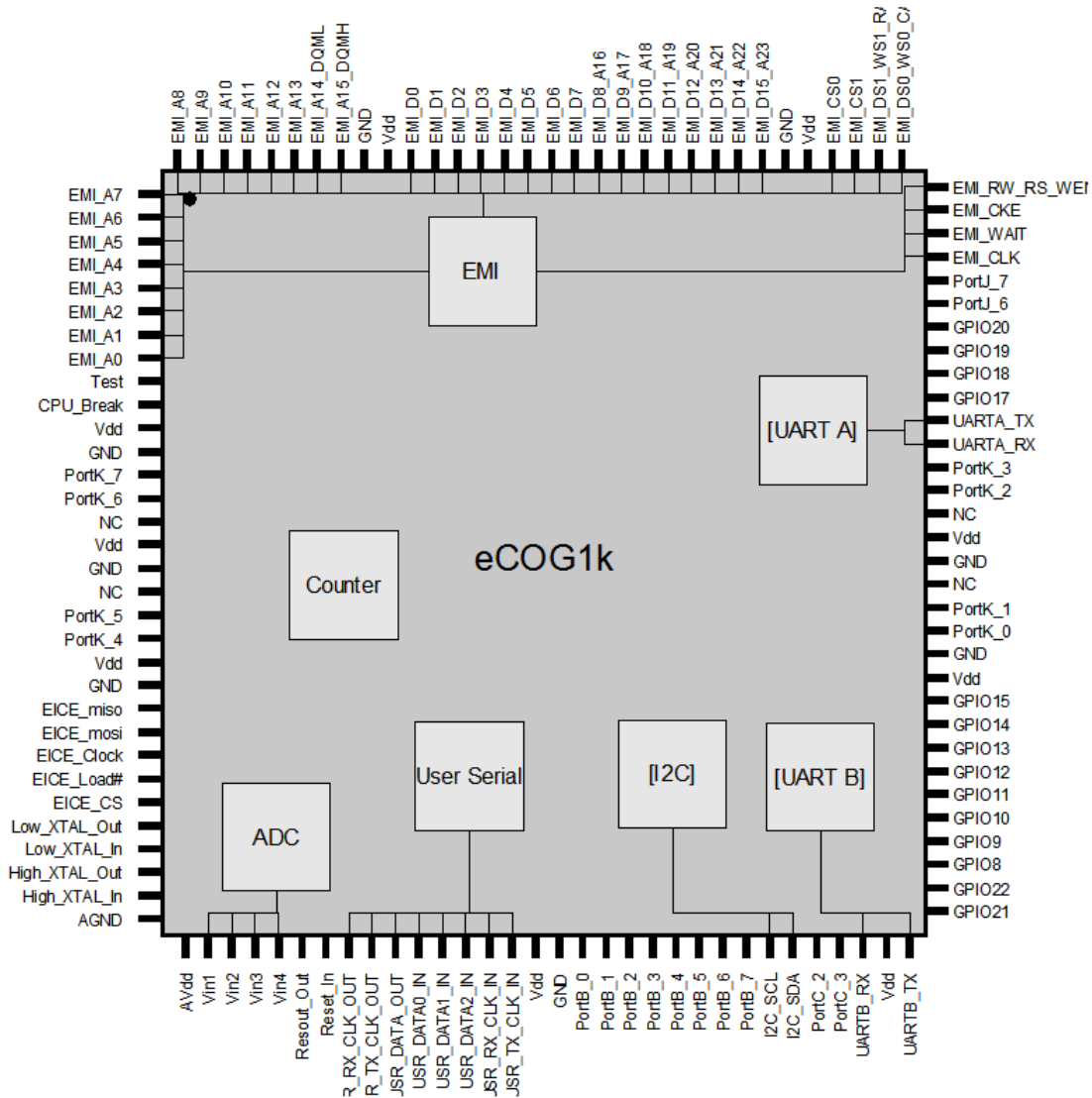




AN009 – Installing the PicOS Operating System for the eCOG1k

Version 1.5

This application note describes how to install the PicOS operating system for the eCOG1k microcontroller on the Cyan development board.



Confidential and Proprietary Information

©Cyan Technology Ltd, 2008

This document contains confidential and proprietary information of Cyan Technology Ltd and is protected by copyright laws. Its receipt or possession does not convey any rights to reproduce, manufacture, use or sell anything based on information contained within this document.

Cyan Technology™, the Cyan Technology logo and Max-eICE™ are trademarks of Cyan Holdings Ltd. CyanIDE® and eCOG® are registered trademarks of Cyan Holdings Ltd. Cyan Technology Ltd recognises other brand and product names as trademarks or registered trademarks of their respective holders.

Any product described in this document is subject to continuous developments and improvements. All particulars of the product and its use contained in this document are given by Cyan Technology Ltd in good faith. However, all warranties implied or expressed, including but not limited to implied warranties of merchantability, or fitness for purpose, are excluded.

This document is intended only to assist the reader in the use of the product. Cyan Technology Ltd shall not be liable for any loss or damage arising from the use of any information in this guide, any error or omission in such information, or any incorrect use of the product.

This product is not designed or intended to be used for on-line control of aircraft, aircraft navigation or communications systems or in air traffic control applications or in the design, construction, operation or maintenance of any nuclear facility, or for any medical use related to either life support equipment or any other life-critical application. Cyan Technology Ltd specifically disclaims any express or implied warranty of fitness for any or all of such uses. Ask your sales representative for details.



Revision History

Version	Date	Notes
V1.0	10/03/2004	First release
V1.1	09/06/2004	Document format updated
V1.2	23/09/2004	Updated list of trademarks
V1.3	02/11/2004	Changed font sizes
V1.4	22/02/2005	Updated for CyanIDE
V1.5	07/04/2005	Updated file title and page headers

Contents

1	Introduction	5
2	Glossary	5
3	Requirements	5
4	Method.....	6
4.1	PicOS Installation	6
4.2	Application Installation.....	6
4.3	Opening the Project.....	6
4.4	Compiling the Application	7
4.5	Loading and Running the Application.....	7
4.6	Results.....	7
Appendix A	File Listings	8
A.1	Project Files.....	8
A.2	cstartup.asm	8
A.3	irq.asm.....	12
A.4	app.c.....	14
A.5	options.sys.....	15
A.6	internal.map.....	15

1 Introduction

The PicOS operating system is a third party small footprint operating system for the eCOG1k microcontroller. It is designed to be installed and run on the eCOG1k development board. The software and this application note are intended to serve as a guide for the PicOS operating system on the eCOG1k development board, and to verify the correct compilation, installation and operation of the system through the use of simple test code. The software is compiled and then downloaded into the eCOG1k on the development board using the tools provided in the CyanIDE software development package. This application note also serves a description of the compilation and installation method for the PicOS operating system.

The majority of the instructions specific to this application are directly applicable to other applications and form a basic method for compiling code and downloading it to the eCOG1k microcontroller.

2 Glossary

A table of abbreviations used in this document.

eCOG1	Cyan Technology target micro controller
LCD	Liquid Crystal Display
PicOS	OlsoNet operating system designed for the eCOG1k
UART	Universal Asynchronous Receiver/Transmitter

3 Requirements

- PC running CyanIDE V1.0 or later.
- An eCOG1k development board, connected to the PC with a download cable.

The PicOS software and application code can be downloaded as a zip archive file from the Cyan Technology web site at www.cyantechology.com. The following instructions are based on using this zip file, named <AN009 Installing PicOS.zip>.

4 Method

4.1 PicOS Installation

Download the zip file <AN009 Installing PicOS.zip> from the Cyan web site. Extract the files to a suitable location on the PC hard disk. A suggested location is the CyanIDE installation directory. <C:\Program Files\Cyan Technology\CyanIDE>. The unzip process puts the extracted files into a sub-directory named PicOS, which contains several further sub-directories:

- System PicOS kernel and eCOG1k system code
- Lib Library routines
- Docs PicOS documentation
- Examples Example PicOS projects for use with CyanIDE
- Apps Examples of PicOS application code
- Linux Test code for Linux host systems
- SimNets Windows tools for linking to eCOG1k simulator

4.2 Application Installation

The PicOS installation includes an example CyanIDE project which contains a basic application that can be used to verify the correct operation of PicOS. In the directory <PicOS\Examples> there is another directory <basic> which contains the following files.

app.c	The main C program code
basic.cyp	CyanIDE project file
cstartup.asm	eCOG1k startup and C initialisation code
devboard.cfg	CyanIDE configuration file
internal.map	CyanIDE memory map file
irq.asm	Reset and interrupt vectors
options.sys	Project-specific PicOS options

Note that the project file contains relative path references to PicOS system and library files that are compiled when the application is built. For simplicity, the example application should stay in its place, two directories below the parent directory *PicOS*. The name of the application project directory may be changed if required. If the application project is moved to a different location, it may be necessary to delete the PicOS system and include files from the project, then add them to the project again, so that the project file uses the correct relative path to these files.

4.3 Opening the Project

From the CyanIDE main menu, select *Project->Open*, and navigate to the directory in which the basic example project is stored. Select the project file (*basic.cyp*) and click the *Open* button. CyanIDE opens the project and displays the source files in the navigator pane to the left of the main window.

Double-click on any file in the navigator (or right-click and select *Open*) to open it in a code editor window in the main CyanIDE workspace.

4.4 Compiling the Application

From the CyanIDE main menu, select *Build->Build*, or *Build->Rebuild All*. CyanIDE compiles the C source files, then assembles and links all assembly files to generate the project output binary file. Any compilation or assembly errors are listed in the build tab of the output window. Double-click on any error to open the corresponding source file in the editor, with the cursor at the line which contains the error.

If changes are made to any *<*.c>*, *<*.h>* or *<*.asm>* files within the project, then it is sufficient to use the *Build* command to recompile the project. CyanIDE checks the project for any dependencies on these files, and compiles and links only those files that need to be rebuilt. However, if any change is made to the file *<options.sys>* then use the *Build->Rebuild All* command to ensure that all project files are recompiled with the new PicOS settings.

4.5 Loading and Running the Application

Having compiled the application, the resulting code is downloaded to the eCOG1k development board using CyanIDE. From the main menu, select *Debug->Run*. CyanIDE downloads the file to target memory and starts execution.

The CyanIDE debugger allows the program to be stopped or paused and restarted. It supports single step by source line or by instruction, and provides windows to display the CPU registers, memory and any watched variables.

4.6 Results

When the program is executed, the software performs a binary counting sequence on the development board LEDs, updated every half-second and looping round on completion. This indicates correct operation and verifies that the installation was successful.

Appendix A File Listings

A.1 Project Files

The basic example project includes the following files:

- basic.cyp CyanIDE project file
- app.c main program
- kernel.c, .h PicOS kernel
- main.c PicOS initialisation and driver code for eCOG1k
- sysio.h PicOS definitions
- irq.asm reset and interrupt vectors
- cstartup.asm C startup and initialisation code
- devboard.cfg peripheral configuration (compiles to devboard.asm)
- internal.map memory map
- options.sys PicOS configuration options

A.2 cstartup.asm

```

;
; =====
;
; Copyright (C) Olsonet Communications Corporation, 2002, 2003
;
; Permission is hereby granted, free of charge, to any person obtaining a copy
; of this software and associated documentation files (the "Software"), to
; deal in the Software without restriction, including without limitation the
; rights to use, copy, modify, merge, publish, distribute, sublicense, and/or
; sell copies of the Software, and to permit persons to whom the Software is
; furnished to do so, subject to the following conditions:
;
; The above copyright notice and this permission notice shall be included in
; all copies or substantial portions of the Software.
;
; THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
; IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
; FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
; AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
; LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
; FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
; IN THE SOFTWARE.
;
; =====
;

                MODULE    cstartup
                .ALL

;
; C reserves DATA address 0 for the NULL pointer. The value H'DEAD is put in
; here so that it is easier to spot the effect of a NULL pointer during
; debugging. User memory for constants grows upwards from H'0001. The first
; address is given the equate symbol $??LO_ADDR.
;
                .SEG      C_RESERVED1
                ORG       0
                dc        H'DEAD
$??LO_ADDR     EQU       $

; PG: starting data address
DS_ADDR       EQU       H'E800

;
; DATA addresses H'EFE0-H'FFFF are used for scratchpad RAM in interrupt mode.
; DATA addresses H'EFC0-H'EFDF are used for scratchpad RAM in user mode.
; DATA addresses H'EFB8-H'EFBF are used for register storage in interrupt mode.
; Then follows the interrupt stack, user stack and user heap.
; User memory for variables grows downwards from the end of the user stack.
; This version of cstartup only contains one area of scratchpad RAM

```

```

; which constrains users not to write re-entrant re-interruptable
; code.
; The interrupt stack must start at IY-38 to be compatible with the C compiler.
;
;
; .SEG      C_RESERVED2
; ORG      H'EFB8

$??HI_ADDR    DEQU    $
; ds      8          ; Interrupt register storage

; ds      32        ; User Scratchpad
IY_SCRATCH    DEQU    $

$?irq_scratchpad? DEQU $
; ds      32        ; Interrupt Scratchpad

;
; The registers that control the functional blocks of the eCOG1 are located
; at addresses H'FEA0 to H'FFCF. The C header file <ecog1.h> declares an
; external structure that describes the registers. This variable is defined
; below.
;
;
; .SEG      REGISTERS
; ORG      H'FEA0

$fd:
$rg           ds      304

;
; C requires the following segments:
;   CONST - Constants in ROM. For example:
;           const char c = 'c' ;
;           printf( "Hello World!" ) ;
;   VAR   - Variables in RAM. These are set to zero by the cstartup code.
;           For example:
;           int i ;                (in file scope)
;           static int i ;         (in function scope)
;   INIT  - Initialised variables in RAM. For example:
;           int i = 9 ;           (in file scope)
;           static int i = 9 ;    (in function scope)
;   INITC - Initialisation data for the INIT segment
;   HEAP  - The heap. Required if malloc() etc. are used.
;   STACK - The stack. Always required.
;
; The memory allocated to each segment is defined by the value of
; $??<segment_name>_SIZE as set below. These sizes can be set manually or, if
; the appropriate line is tagged with !PACK and the -pack option is specified
; to ECOGCL, ECOGCL will write in the size actually required for the segment.
; The sizes of the STACK and HEAP segments must be set by the user.
;
$??ISTACK_SIZE =      H'0040
$??STACK_SIZE  =      H'0100
$??HEAP_SIZE   =      H'0000 ; PG - No heap in this application

; ROM segments
$??INITC_SIZE  =      h'0005 ; !PACK
$??CONST_SIZE  =      h'018e ; !PACK

; RAM segments
$??INIT_SIZE   =      h'0005 ; !PACK
$??VAR_SIZE    =      h'013d ; !PACK

; -- Locate DATA segments in memory --
;
; Segments are allocated sequentially by the ??ALLOCATE macro. They may be
; set at fixed addresses by setting ADDR prior to calling ??ALLOCATE.
;
??ALLOCATE     MACRO    seg
;               .SEG    &seg
;               ORG     ADDR

$??&seg!_LO   = ADDR
ADDR          = ADDR + $??&seg!_SIZE
$??&seg!_HI   = ADDR-1
;               ENDMAC

; Allocate DATA ROM

```

```

ADDR          = $$$LO_ADDR
              ??ALLOCATE INITC
              ??ALLOCATE CONST

; Allocate DATA RAM
; PG I don't get it. Why going from the bottom? I am fixing it to
; start from the top and leave as much room for the stack as available

ADDR          = DS_ADDR
              ??ALLOCATE INIT
              ??ALLOCATE VAR
              ??ALLOCATE HEAP

EVAR          = ADDR

; PG The stack goes from the bottom
ADDR          = $$$HI_ADDR - $$$ISTACK_SIZE - $$$STACK_SIZE
ESTK          = ADDR
              ??ALLOCATE STACK
              ??ALLOCATE ISTACK

; -- Memory initialisation macros --
;
; Segments may be initialised by filling with a constant value using the
; ??SEGFILL macro. Two symbols are passed, the segment name and the value to
; fill with. A third symbol (the size) is assumed.
;
??SEGFILL     MACRO    seg, value
              LOCAL   fill_loop
              IF      $$$&seg!_SIZE
              ld      x, $$$&seg
              ld      al, $$$&seg!_SIZE
              ld      ah, &value
&fill_loop:  st      ah, @(0,x)
              add     x, #1
              sub     al, #1
              bne     &fill_loop
              ENDF
              ENDMAC

;
; Segments may be initialised by copying an initialisation segment with
; the ??SEGCOPY macro. Two symbols are passed, the source and destination
; segment names.
;
??SEGCOPY     MACRO    src, dest
              IF      $$$&src!_SIZE NE $$$&dest!_SIZE
              .ERR    "Copy segments different sizes"
              ENDF
              IF      $$$&src!_SIZE
              ld      x, $$$&src
              ld      y, $$$&dest
              ld      al, $$$&src!_SIZE
              bc
              ENDF
              ENDMAC

;
; Fills a block of memory with a value. Three values are passed, the start
; address for the block, the number of addresses to write to and the value
; to be written.
;
??MEMFILL     MACRO    start, length, value
              LOCAL   fill_loop
              ld      x, &start
              ld      al, &length
              ld      ah, &value
&fill_loop:  st      ah, @(0,x)
              add     x, #1
              sub     al, #1
              bne     &fill_loop
              ENDMAC

;
; Start of Code.

```

```

;
; .CODE
; ORG H'40

$?cstart_code:
bra $ecog1ConfigMMU ; configure MMU and Cache Banks

$ecog1ConfigContinue:
;
; Initialise segments. The HEAP and STACK are filled with H'9999 and H'aaaa
; respectively so that their maximum runtime extents can be checked. The
; INIT segment is set from the ROM initialisers in the INITC segment. The non
; initialised RAM segment VAR is set to zero (compiler puts 0 initialised
; variables in these segments as well as uninitialised ones,x).
;
; ??SEGFILL HEAP, #h'9999
; ??SEGFILL STACK, #h'AAAA
; ??SEGFILL ISTACK, #h'BBBB
; ??SEGCOPY INITC, INIT
; ??SEGFILL VAR, #h'0

; Set interrupt stack pointer.
ld y, #IY_SCRATCH

; Set user mode flag to allow interrupts.
st flags, @(-1,y)
ld al, @(-1,y)
or al, #h'10
st al, @(-1,y)
ld flags, @(-1,y)

; Set usermode stack pointer
ld y, #$$$STACK_HI

; Call ecog1Config to setup eCOG1 peripherals
; Defined in module produced by configuration compiler
bsr $ecog1Config

; Call scheduler
bra $zzz_sched

;
; Main may exit by returning or by explicitly calling $exit. In either case
; exit code will be in AL.
;
$exit:
brk ; Alert the user if in debug mode
bra 0 ; Restart

;
; This is the minimal interrupt routine. The contents of FLAGS is restored
; as the program counter is restored using rti.
;
$minimal_handler:
st flags,@(-33,y) ; Store Flags
st al, @(-34,y) ; Store AL

ld al, @(-33,y) ; Put Flags into AL
or al, #h'0010 ; Set usermode
st al, @(-33,y) ; Store the value to be restored to Flags

brk ; Alert the user if in debug mode

ld al, @(-34,y) ; Restore AL
rti @(-33,y) ; Restore PC and Flags

;
; The address exception can happen often during development. A handler
; is put here to catch the exception.
;
$address_error:
st flags,@(-33,y) ; Store Flags
st al, @(-34,y) ; Store AL

ld al, @(-33,y) ; Put Flags into AL

```

```

    or al, #'0010      ; Set usermode
    st al, @(-33,y)   ; Store the value to be restored to Flags

    brk                ; Alert the user if in debug mode

    ld al, #'a
    st al, @h'ff69    ; Clear status in mmu.address_exception

    ld al, #'200
    st al, @h'ff7a    ; Clear status in emi.ctrl_sts

    ld al, @(-34,y)   ; Restore AL
    rti @(-33,y)     ; Restore PC and Flags

; The release operation - to return from a process directly to the scheduler
; loop
$zzz_set_release:
    st Y,@release_sp
    st X,@release_rt
; this assumes that the release address is within the first 64K of memory,
; which, of course, is going to be the case, as we are talking about an
; address within the scheduler loop
$zzz_release:
    ld Y,@release_sp
    ld AL, #'0
    mov XH,AL
    bra @release_rt

; End of startup code
$??CSTARTUP_END EQU $

        .SEG VAR
release_sp ds 1
release_rt ds 1

        .SEG CONST
; Make the boundaries of data and stack available to the program
$sevar_   dc EVAR
$sestk_   dc ESTK

        ENDMOD

```

A.3 irq.asm

```

; =====
;
; Copyright (C) Olsonet Communications Corporation, 2002, 2003
;
; Permission is hereby granted, free of charge, to any person obtaining a copy
; of this software and associated documentation files (the "Software"), to
; deal in the Software without restriction, including without limitation the
; rights to use, copy, modify, merge, publish, distribute, sublicense, and/or
; sell copies of the Software, and to permit persons to whom the Software is
; furnished to do so, subject to the following conditions:
;
; The above copyright notice and this permission notice shall be included in
; all copies or substantial portions of the Software.
;
; THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
; IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
; FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
; AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
; LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
; FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
; IN THE SOFTWARE.
;
; =====

        MODULE kernelirq
        .ALL

```

```

;
; Interrupt vectors are sign-extended 24 bit absolute branch addresses.
; For example a vector containing H'1234 will start executing at H'1234
; and a vector containing H'8765 will start executing at H'ff8765.
; All unused interrupts are handled by the default code in minimal_handler.
;
                ORG     H'0
                bra     $?cstart_code

                ORG     H'4

ex_debug                DC $minimal_handler
ex_tim_wdog_ufl         DC $minimal_handler
ex_adr_err              DC $address_error
reserved                DC $minimal_handler
ex_tim                  DC $minimal_handler
ex_reserved             DC $minimal_handler
ex_usart_a              DC $minimal_handler
ex_usart_b              DC $minimal_handler
ex_uart_a               DC $uart_a_int??
ex_uart_b               DC $uart_b_int??
int_tim_tmr_ufl         DC $timer_int??
int_tim_cnt1_ufl        DC $minimal_handler
int_tim_cnt2_ufl        DC $minimal_handler
int_tim_cnt1_match      DC $minimal_handler
int_tim_cnt2_match      DC $minimal_handler
int_tim_pwm1_ufl        DC $minimal_handler
int_tim_pwm2_ufl        DC $minimal_handler
int_tim_pwm1_match      DC $minimal_handler
int_tim_pwm2_match      DC $minimal_handler
int_tim_cap_of1         DC $minimal_handler
int_tim_cap1            DC $minimal_handler
int_tim_cap2            DC $minimal_handler
int_tim_cap3            DC $minimal_handler
int_tim_cap4            DC $minimal_handler
int_tim_cap5            DC $minimal_handler
int_tim_cap6            DC $minimal_handler
int_tim_ltmr_ufl        DC $minimal_handler
int_reserved1           DC $minimal_handler
int_reserved2           DC $minimal_handler
int_reserved3           DC $minimal_handler
int_reserved4           DC $minimal_handler
int_reserved5           DC $minimal_handler
int_reserved6           DC $minimal_handler
int_reserved7           DC $minimal_handler
int_reserved8           DC $minimal_handler
int_usart_a_rx          DC $minimal_handler
int_usart_a_tx          DC $minimal_handler
int_usart_b_rx          DC $minimal_handler
int_usart_b_tx          DC $minimal_handler
int_sci_tx_done         DC $minimal_handler
int_sci_tx_err         DC $minimal_handler
int_sci                 DC $minimal_handler
int_ifr_tx_done         DC $minimal_handler
int_ifr_rx_done         DC $minimal_handler
int_ifr_rx_err         DC $minimal_handler
int_ifr_frame_done     DC $minimal_handler
int_uart_a_tx_rdy       DC $uart_a_int??
int_uart_a_rx_rdy       DC $uart_a_int??
int_uart_b_tx_rdy       DC $uart_b_int??
int_uart_b_rx_rdy       DC $uart_b_int??
int_ehi                 DC $minimal_handler
int_gpio                DC $gpio_int??
int_adc                 DC $minimal_handler

```

ENDMOD

A.4 app.c

```

/* ===== */
/*
/* Copyright (C) Olsonet Communications Corporation, 2002, 2003
/*
/*
/* Permission is hereby granted, free of charge, to any person obtaining a copy
/* of this software and associated documentation files (the "Software"), to
/* deal in the Software without restriction, including without limitation the
/* rights to use, copy, modify, merge, publish, distribute, sublicense, and/or
/* sell copies of the Software, and to permit persons to whom the Software is
/* furnished to do so, subject to the following conditions:
/*
/*
/* The above copyright notice and this permission notice shall be included in
/* all copies or substantial portions of the Software.
/*
/*
/* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
/* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
/* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
/* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
/* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
/* FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
/* IN THE SOFTWARE.
/*
/*
/* ===== */

#include "sysio.h"
#include "trc.h"          // Enable tracing

/* ===== */
/* This is the basic installation test of the PicOS operating system.
/* The code will flash the LED's on the eval board to verify install
/*
/*
/* Tony Ward - March 2004
*/
/* ===== */

/* Use 100% memory for the heap */
heapmem {100};

#include "lcd.h"
#include "led.h"

#define INIT_LCD      00
#define INIT_OS       10

process (root, void)
/* ===== */
/* This is the main program of the application
/* ===== */

    static const char *msg = "PicOS:          LED example";
    static const char *pat = "0123456789ABCDEF";
    nodata;

    entry (INIT_LCD)
        dsp_lcd(msg, YES);

    entry (INIT_OS)
        dsp_led(pat, 512);
    finish;

endprocess (1)

```

A.5 options.sys

```
#define ECOG_SIM                0
#define DIAG_MESSAGES          2    /* System messages to UART */
#define UART_DRIVER            2    /* Both UARTs present */
#define LEDS_DRIVER            1    /* LED driver present */
#define LCD_DRIVER             1    /* LCD driver present */
#define TCV_PRESENT            0    /* No TCV */
#define TCV_TIMERS             0    /* */
#define TCV_HOOKS              0    /* */
#define ETHERNET_DRIVER        0    /* No Ethernet */
#define SDRAM_PRESENT          1    /* SDRAM present, malloc using SDRAM */
#define CODE_LONG_INTS         1    /* 'ld' 'lu' 'lx' in formats */
#define MALLOC_SAFE            1
```

A.6 internal.map

```
=====
; Cyan Technology Ltd
; Example application software for eCOG
;
; FILE
; internal.map
;
; DESCRIPTION
; Demonstration architecture description file.
=====
code 0 77ff irom 0000 77ff
data 0 7ff irom 7800 7fff
data e800 efff iram 0 7ff
```