Homebrew Embedded Systems for Amateur Radio

Mike, G8CUL/F4VRB

(mikeg8cul@gmail.com)

RALRT 2025

- What is an embedded system?
- About me
- So why not apply some of what I did professionally to Amateur Radio roles?

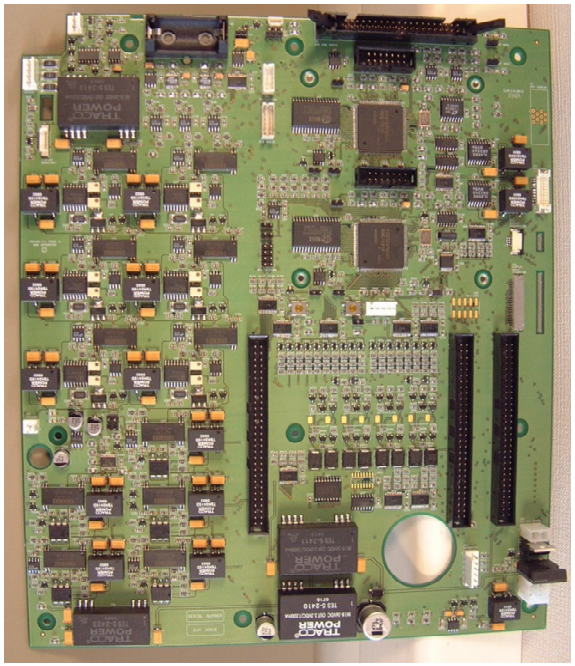
- System design, PCB design –now unpaid, but for me!
- However, not all have an embedded processor – but most do
- Embedded microcontroller device, or a single board computer

- Instrumentation I worked on professionally
- Some complex multi-processor designs, hopefully not often needed in an Amateur Radio context


iCAM
ALPHA/BETA AIR MONITOR
CANBERRA


DOSE RATE
CANBERRA


CANBERRA
RMS RATEMETER



- My first ever embedded system was designed in 1978
- Motorola 6800 + ancillary devices

HARWELL

UNCLASSIFIED

Technical Specification

Harwell 6000 Series Microcomputer (MOUSE)

95/6161-1/6

Issue No. 1

Introduction

The Harwell 6000 Series microcomputer (MOUSE) (Microcomputer Organised for Users) has been designed to make low cost computing facilities available to users of the Harwell 6000 Series. It allows computer-based systems for experimental and dedicated applications to be developed rapidly and economically for both laboratory and plant use. It is compatible with both the Systems and Laboratory series of 6000 Series units and can be used in 2000 Series systems with appropriate adaptors.

MOUSE is a complete microcomputer in 6000 Series format with a number of facilities and a standard internal program which controls the operations of these facilities.

MOUSE includes:

- a Motorola M6800 microprocessor
- 2K control program with provision for up to 14K of additional program
- 1K RAM (expandable to 2K)

The facilities in MOUSE include:

- 8 × 32 bit scalers (capacity 4 × 10^9 counts each)
- 1 timer (internal or external clock) (capacity approximately 20 days)
- 16 parallel inputs with full handshaking
- 16 parallel outputs with full handshaking
- 1 pulse output
- IEEE-488 interface

General Description

Methods of Operation

MOUSE can be used:

(a) as a stand-alone computer, holding all of the applications programs required for data acquisition, data reduction and control functions. These programs must be developed on a microcomputer development system.

(b) in conjunction with another computer which acts as an IEEE-488 controller. In this case the applications programs in the controller pass simple instructions to the standard control program in MOUSE which operates the facilities of MOUSE for data acquisition and output tasks, with control and data reduction operations being carried out in the IEEE-488 controller.

The controller can be one of a number of computers with IEEE-488 controlling functions and programmed in appropriate languages, such as the Commodore PET (BASIC), DEC LSI-11, DEC MINC (BASIC) and Hewlett Packard calculators and computers. A controller can operate with up to 16 MOUSE modules in a system.

The use of a high level language such as BASIC in the controller permits application programs to be

- Mainly now use Microchip PICs. There is a huge range available

- They range from the smallest – A 6 pin SOT23 device, up to a multi-pin PIC32 - a very powerful processor

- Software development tools are easily available and are FREE

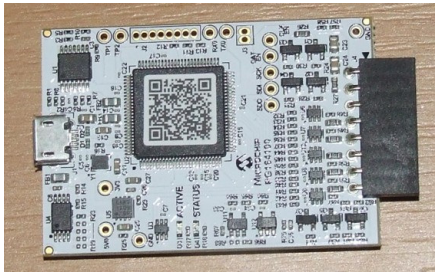
- Development tools are common to the whole range of PIC processors. These tools include MPLAB and the compiler relevant to the processor

- The tools allow source code editing, compiling, programming and ***debugging***

- Programming and debugging require hardware interface -

Programming/debug interface hardware

PICkit2



Newer ‘SNAP’ –

PICkit3 in use debugging a pollution measuring system complete with display and GPS using a PIC32

Latest (low cost) PICkit BASIC device,

- The older tools work with the older devices
- The newer tools are required for newer devices
- High level programming language (C), assembly language available but I wouldn’t recommend it for the inexperienced

- Hardware (PCB) design. I design my own and use Easy-PC but others are available (KiCad and RS's Design Spark are good examples)

- They provide the tools to 'create' the component (schematic symbol and footprint)

- They provide the tools to create the schematic and the PCB layout. Checks exist that ensure that the final copper matches the schematic!

- Boards from China –cost is very low (typically <£1 per board of < 100mm square)
- This is for a double-sided board with plated through holes, solder resist and silk screen both sides
- The quality is good and the turnround time is reasonable

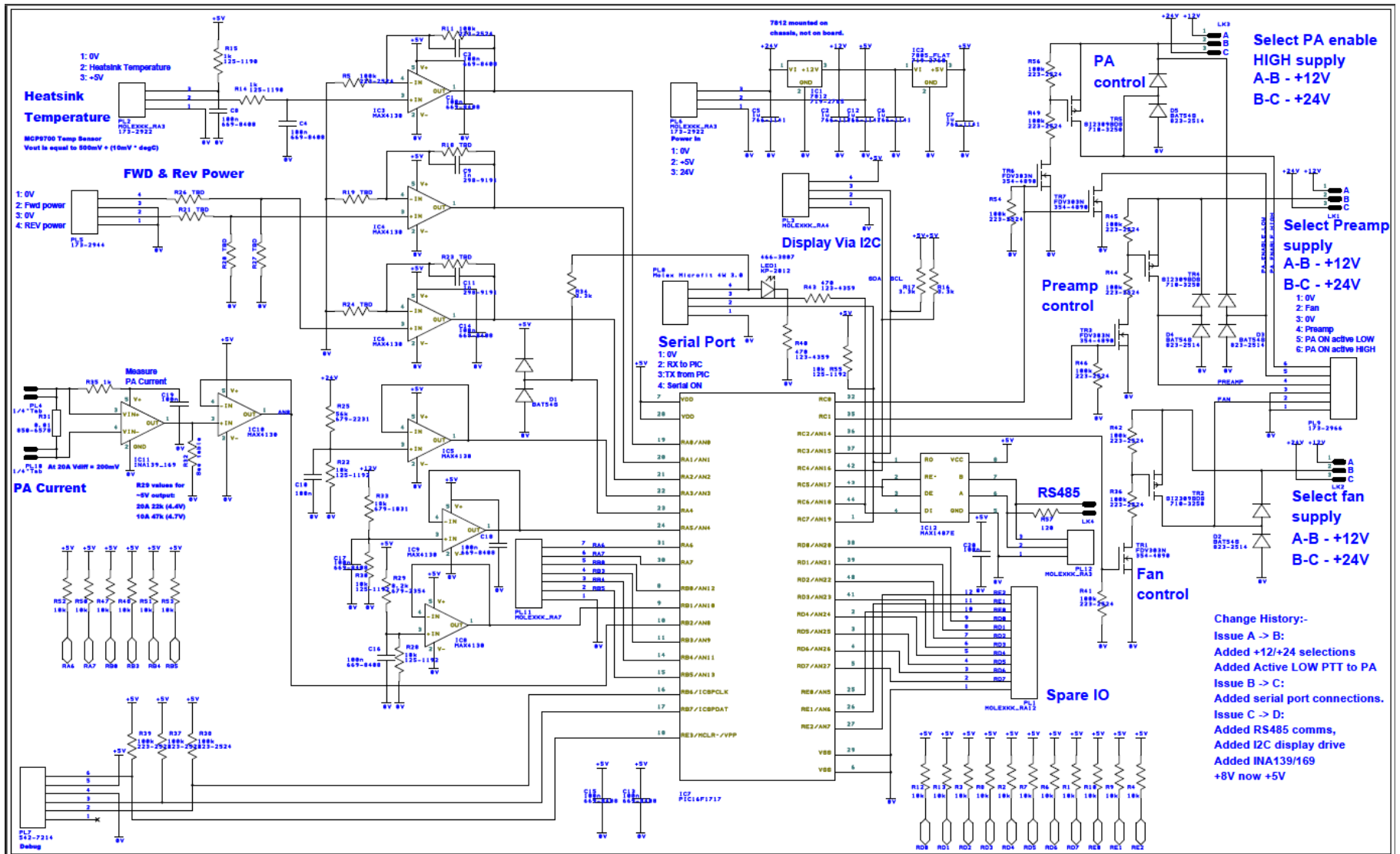
- Boards are hand assembled – but a magnifying lens and a bright light helps

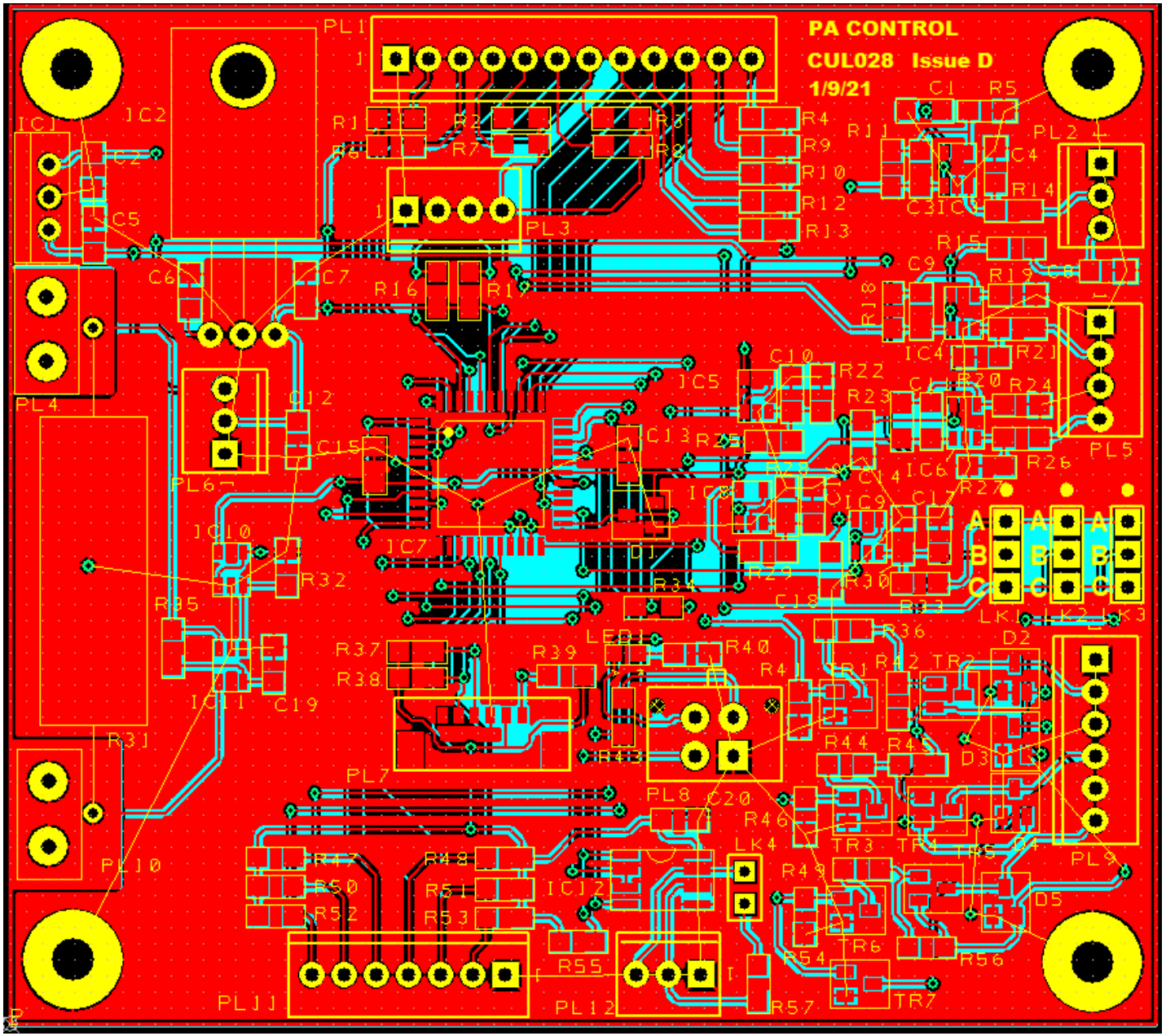
- I nearly always use Surface Mount components normally 1206 size (3mm x 1.5mm). With practice, these are easy to hand solder

- Board design. Easy-PC, others are available. KiCad seems to be popular. RS used to promote DesignSpark, (same stable as Easy-PC)

- This is used to do 3 things

1. Create the schematic
2. Create the PCB artwork
3. But first, ‘create’ the components!


Heatsink Temperature
1: 0V
2: Heatsink Temperature
3: +5V
MCP9700 Temp Sensor
Vout is equal to 500mV + (10mV * degC)
FWD & Rev Power
1: 0V
2: Fwd power
3: 0V
4: REV power
Measure PA Current
PA Current
At 20A Vdiff = 200mV
R29 values for ~5V output:
20A 22k (4.4V)
10A 47k (4.7V)
Serial Port
1: 0V
2: RX to PIC
3:TX from PIC
4: Serial ON
7812 mounted on chassis, not on board.
Power In
1: 0V
2: +5V
3: 24V
Display Via I2C
PA control
Preamp control
Fan control
RS485
Spare IO
Select PA enable
HIGH supply
A-B - +12V
B-C - +24V
Select Preamp supply
A-B - +12V
B-C - +24V
1: 0V
2: Fan
3: 0V
4: Preamp
5: PA ON active LOW
6: PA ON active HIGH
Select fan supply
A-B - +12V
B-C - +24V
Change History:-
Issue A -> B:
Added +12/+24 selections
Added Active LOW PTT to PA
Issue B -> C:
Added serial port connections.
Issue C -> D:
Added RS485 comms,
Added I2C display drive
Added INA139/169
+8V now +5V
IC7 PIC16F1717
Debug
Issue D 20/7/21
Issue C 5/12/19
Issue B 24/12/18
Issue A 26/4/17
DRN GBCUL
CHK
APP
Title PA Controller
Drawing No. CUL028
Current Issue Issue D
Sheet 1 of 1

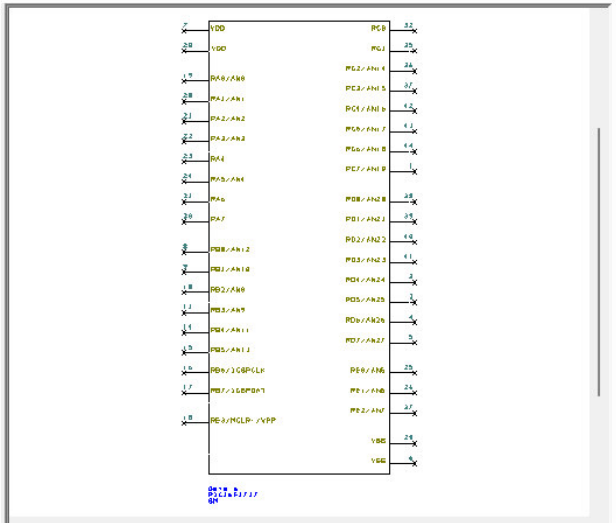

PA CONTROL
CUL028 Issue D
1/9/21

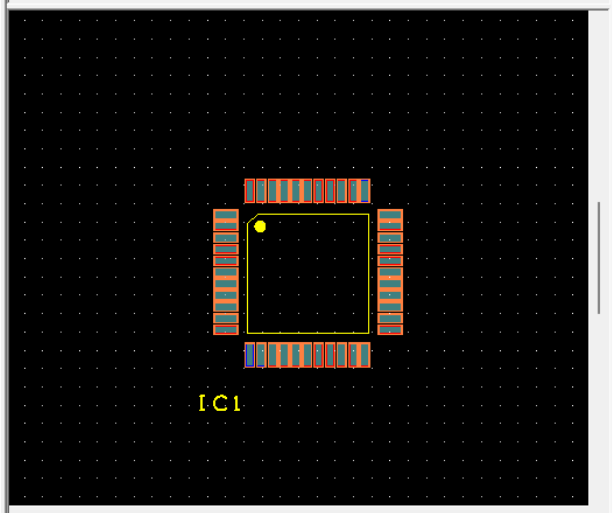
Easy-PC 1000 by Number One Systems - [C:\MyStuff\work\easyPC\Library_processors.cml - PIC16F1717 - Untitled]

File Edit View Settings Window Help

Gate Name	Sch Symbol Name	Sch Symbol Terminal Name	Sch Terminal Number	Pcb Symbol Pad Number	Component Pin Name/Number	Net (Class) Name	No Connect
a	PIC16F887	VDD	1	7	7		
		VDD	2	28	28		
		RA0/AN0	3	19	19		
		RA1/AN1	4	20	20		
		RA2/AN2	5	21	21		
		RA3/AN3	6	22	22		
		RA4	7	23	23		
		RA5/AN4	8	24	24		
		RA6	9	31	31		
		RA7	10	30	30		
		RB0/AN12	11	8	8		
		RB1/AN10	12	9	9		
		RB2/AN8	13	10	10		
		RB3/AN9	14	11	11		
		RC0	15	32	32		
		RC1	16	35	35		
		RC2/AN14	17	36	36		
		RC3/AN15	18	37	37		
		RC4/AN16	19	42	42		
		RC5/AN17	20	43	43		
		RC6/AN18	21	44	44		
		RC7/AN19	22	1	1		
		RD0/AN20	23	38	38		
		RD1/AN21	24	39	39		
		RD2/AN22	25	40	40		
		RD3/AN23	26	41	41		
		RD4/AN24	27	2	2		
		RD5/AN25	28	3	3		
		RB4/AN11	29	14	14		
		RB5/AN13	30	15	15		
		RB6/ICSPCLK	31	16	16		
		RB7/ICSPDAT	32	17	17		
		RE3/MCLR~/VPI	33	18	18		
		RD6/AN26	34	4	4		
		RD7/AN27	35	5	5		
		RE0/AN5	36	25	25		
		RE1/AN6	37	26	26		
		RE2/AN7	38	27	27		

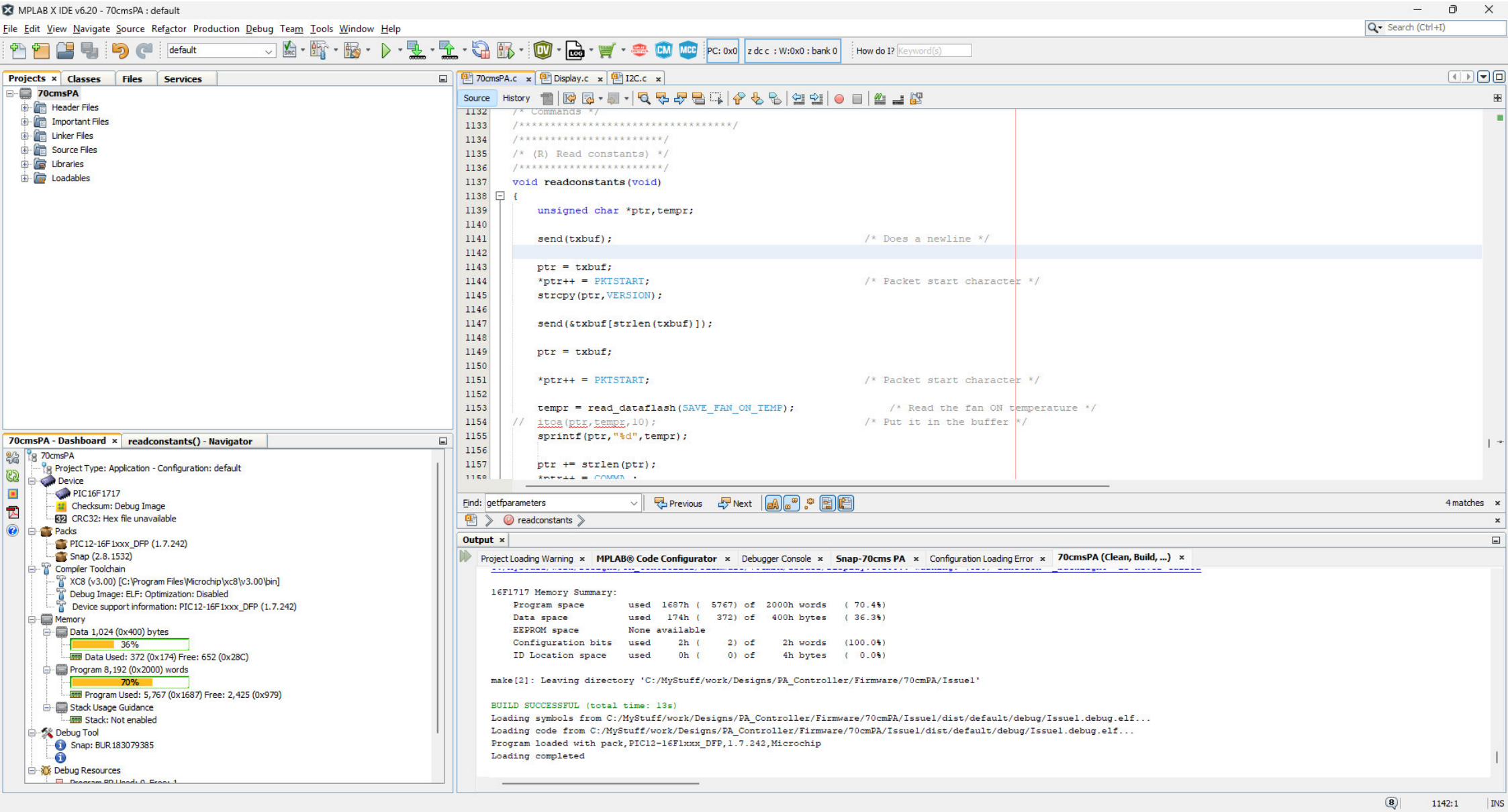
Pin Assignments | Packages | Gates | Values | Properties | Associated Parts





CULK.sch | Start Page | Dolphin_Controller.prj | Dolphin_Controller.pcb* | Dolphin_Controller.sch* | PIC16F1717

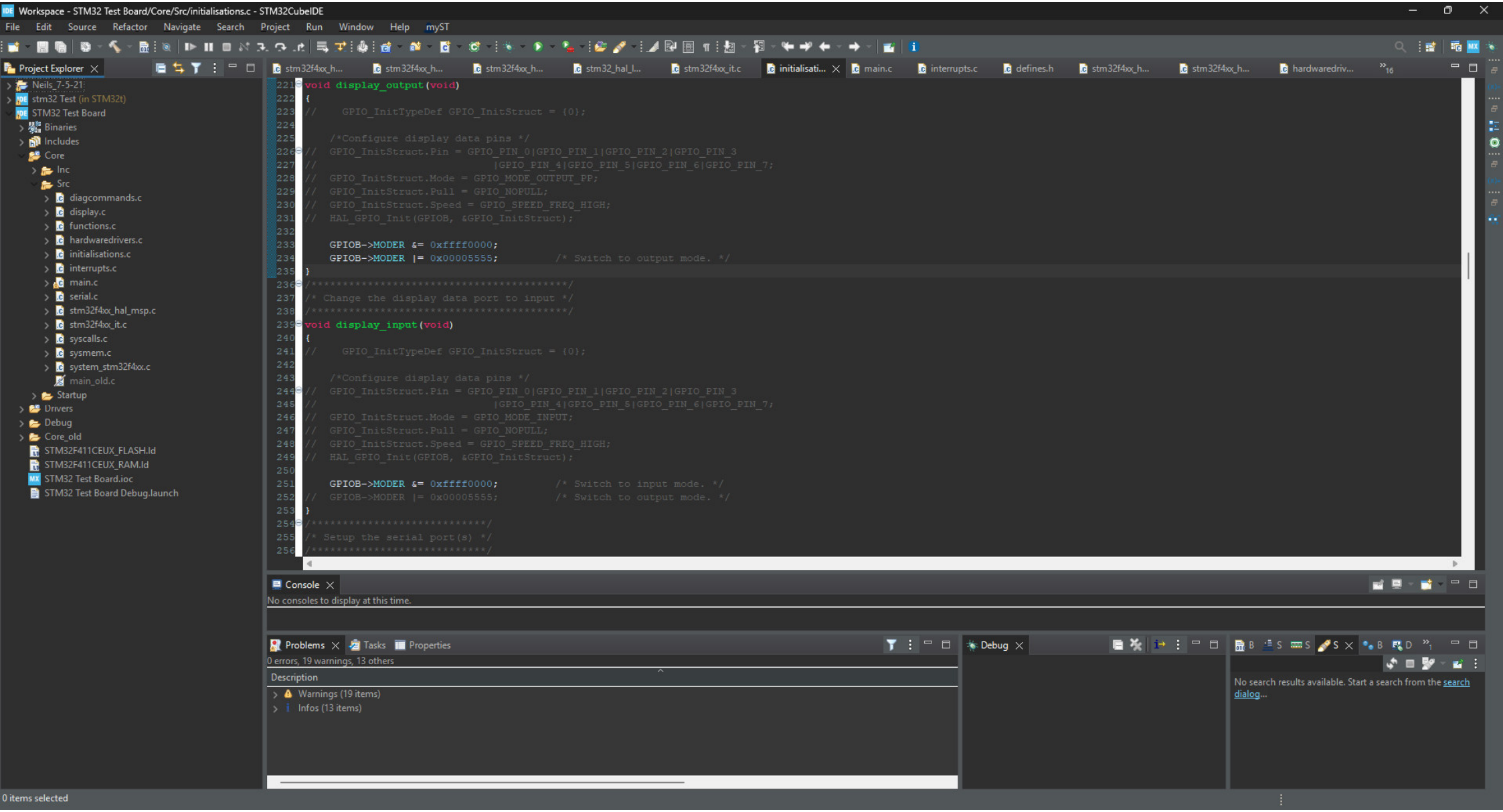
- The new board may need its own firmware
- I write the code in standard K&R ‘C’
- For PICs, using the MPLAB IDE from Microchip works well and includes the editor and compiler but you need the interface to the hardware as in slide 8 above


MPLAB X IDE v6.20 - 70cmsPA : default
File Edit View Navigate Source Refactor Production Debug Team Tools Window Help
Search (Ctrl+I)
default
PC: 0x0
z dc c : W:0x0 : bank 0
How do I?
Keyword(s)
Projects
Classes
Files
Services
70cmsPA
Header Files
Important Files
Linker Files
Source Files
Libraries
Loadables
70cmsPA - Dashboard
readconstants() - Navigator
70cmsPA
Project Type: Application - Configuration: default
Device
PIC16F1717
Checksum: Debug Image
CRC32: Hex file unavailable
Packs
PIC12-16F1xxx_DFP (1.7.242)
Snap (2.8.1532)
Compiler Toolchain
XC8 (v3.00) [C:\Program Files\Microchip\xc8\v3.00\bin]
Debug Image: ELF: Optimization: Disabled
Device support information: PIC12-16F1xxx_DFP (1.7.242)
Memory
Data 1,024 (0x400) bytes
36%
Data Used: 372 (0x174) Free: 652 (0x28C)
Program 8,192 (0x2000) words
70%
Program Used: 5,767 (0x1687) Free: 2,425 (0x979)
Stack Usage Guidance
Stack: Not enabled
Debug Tool
Snap: BUR183079385
Debug Resources
70cmsPA.c
Display.c
I2C.c
Source
History
1132 /* Commands */
1133 /*****************************/
1134 /************************/
1135 /* (R) Read constants) */
1136 /************************/
1137 void readconstants(void)
1138 {
1139 unsigned char *ptr,tempr;
1140
1141 send(txbuf); /* Does a newline */
1142
1143 ptr = txbuf;
1144 *ptr++ = PKTSTART; /* Packet start character */
1145 strcpy(ptr,VERSION);
1146
1147 send(&txbuf[strlen(txbuf)]);
1148
1149 ptr = txbuf;
1150
1151 *ptr++ = PKTSTART; /* Packet start character */
1152
1153 tempr = read_dataflash(SAVE_FAN_ON_TEMP); /* Read the fan ON temperature */
1154 // itoa(ptr,tempr,10); /* Put it in the buffer */
1155 sprintf(ptr,"%d",tempr);
1156
1157 ptr += strlen(ptr);
Find: getfparameters
Previous
Next
4 matches
readconstants
Output
Project Loading Warning
MPLAB® Code Configurator
Debugger Console
Snap-70cms PA
Configuration Loading Error
70cmsPA (Clean, Build, ...)
16F1717 Memory Summary:
Program space used 1687h (5767) of 2000h words (70.4%)
Data space used 174h (372) of 400h bytes (36.3%)
EEPROM space None available
Configuration bits used 2h (2) of 2h words (100.0%)
ID Location space used 0h (0) of 4h bytes (0.0%)
make[2]: Leaving directory 'C:/MyStuff/work/Designs/PA_Controller/Firmware/70cmPA/Issue1'
BUILD SUCCESSFUL (total time: 13s)
Loading symbols from C:/MyStuff/work/Designs/PA_Controller/Firmware/70cmPA/Issue1/dist/default/debug/Issue1.debug.elf...
Loading code from C:/MyStuff/work/Designs/PA_Controller/Firmware/70cmPA/Issue1/dist/default/debug/Issue1.debug.elf...
Program loaded with pack,PIC12-16Flxxx_DFP,1.7.242,Microchip
Loading completed
1142:1
INS

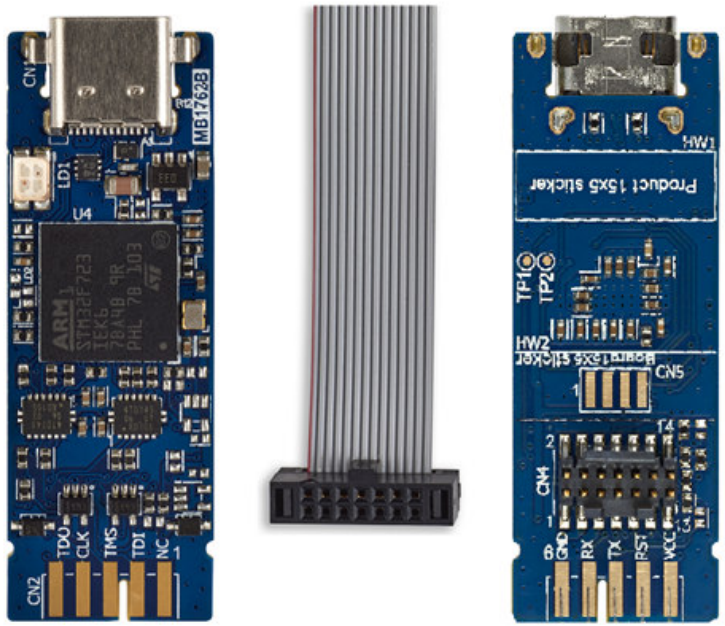
Or use a ready-built single board computer such as a Raspberry Pi, an Arduino or ST ‘Blackpill’.

The latter is a very powerful 32bit device which costs < £10

ST provide the IDE which is similar to the microchip IDE. It has the memorable name of STM32CubeIDE.


Workspace - STM32 Test Board/Core/Src/initialisations.c - STM32CubeIDE
File Edit Source Refactor Navigate Search Project Run Window Help myST
Project Explorer
Neils_7-5-21
stm32 Test (in STM32t)
STM32 Test Board
Binaries
Includes
Core
Inc
Src
diagcommands.c
display.c
functions.c
hardwaredrivers.c
initialisations.c
interrupts.c
main.c
serial.c
stm32f4xx_hal_msp.c
stm32f4xx_it.c
syscalls.c
sysmem.c
system_stm32f4xx.c
main_old.c
Startup
Drivers
Debug
Core_old
STM32F411CEUX_FLASH.ld
STM32F411CEUX_RAM.ld
STM32 Test Board.ioc
STM32 Test Board Debug.launch
stm32f4xx_h...
stm32f4xx_h...
stm32f4xx_h...
stm32_hal_l...
stm32f4xx_it.c
initialisati...
main.c
interrupts.c
defines.h
stm32f4xx_h...
stm32f4xx_h...
hardwaredriv...
void display_output(void)
{
// GPIO_InitTypeDef GPIO_InitStruct = {0};
/*Configure display data pins */
// GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3
// |GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6|GPIO_PIN_7;
// GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
// GPIO_InitStruct.Pull = GPIO_NOPULL;
// GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
// HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
GPIOB->MODER &= 0xffff0000;
GPIOB->MODER |= 0x00005555; /* Switch to output mode. */
}
/**************************************/
/* Change the display data port to input */
/**************************************/
void display_input(void)
{
// GPIO_InitTypeDef GPIO_InitStruct = {0};
/*Configure display data pins */
// GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3
// |GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6|GPIO_PIN_7;
// GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
// GPIO_InitStruct.Pull = GPIO_NOPULL;
// GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
// HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
GPIOB->MODER &= 0xffff0000; /* Switch to input mode. */
// GPIOB->MODER |= 0x00005555; /* Switch to output mode. */
}
/****************************/
/* Setup the serial port(s) */
/****************************/
Console
No consoles to display at this time.
Problems
Tasks
Properties
0 errors, 19 warnings, 13 others
Description
Warnings (19 items)
Infos (13 items)
Debug
No search results available. Start a search from the search dialog...
0 items selected

The interface to the STM32 proecssor is very low cost and available for ~£10. This allows programming and debugging, breakpoints, single stepping and viewing variables.



STLINK-V3MINIE top, bottom, and cable views. Picture is not contractual.

Some of my recent designs –

CUL017 GPSDO and Shack Clock
CUL018 FIM4 Processor
CUL025 ADF Driver (MMRT construction contest winner)
CUL028 PA Controller
CUL029 Transverter Interface (no embedded processor!)
CUL030 Power Meter (Published in RadCom)
CUL033 3.4GHz Transverter Interface
CUL036 Sequencer

UL037 N7DDC ATU remote Display Interface (1) (*)
UL038 N7DDC ATU remote Display Interface (2) (*)
UL043 PIC32 Processor test board
UL048 ‘Flanged’ attenuator board (no processor)
UL052 STM32 processor test board
UL053 Miller GPSDO remake
UL054 GPSDO Small info display
UL056 Rugby Simulator
UL057 GPSDO Large time display

Here are examples for you to look at.