UT32M0R500-EVB Users Guide

PRODUCT NAME	MANUFACTURER PART NUMBER	SMD #	DEVICE TYPE
Arm Cortex M0+	UT32M0R500	5962-17212	01

Table 1: Cross Reference of Applicable Products

1.0 Introduction

The UT32M0R500-EVB Development Board provides a comprehensive and rapid prototyping platform for the UT32M0R500 Microcontroller. The Arduino™ Uno connectivity and full product pinout allow for easy expansion and accessibility. Along with the microcontroller, the subject board supports an external clock, includes JTAG connectors for debugging, and USB-to-UART connectors for communicating from a PC.

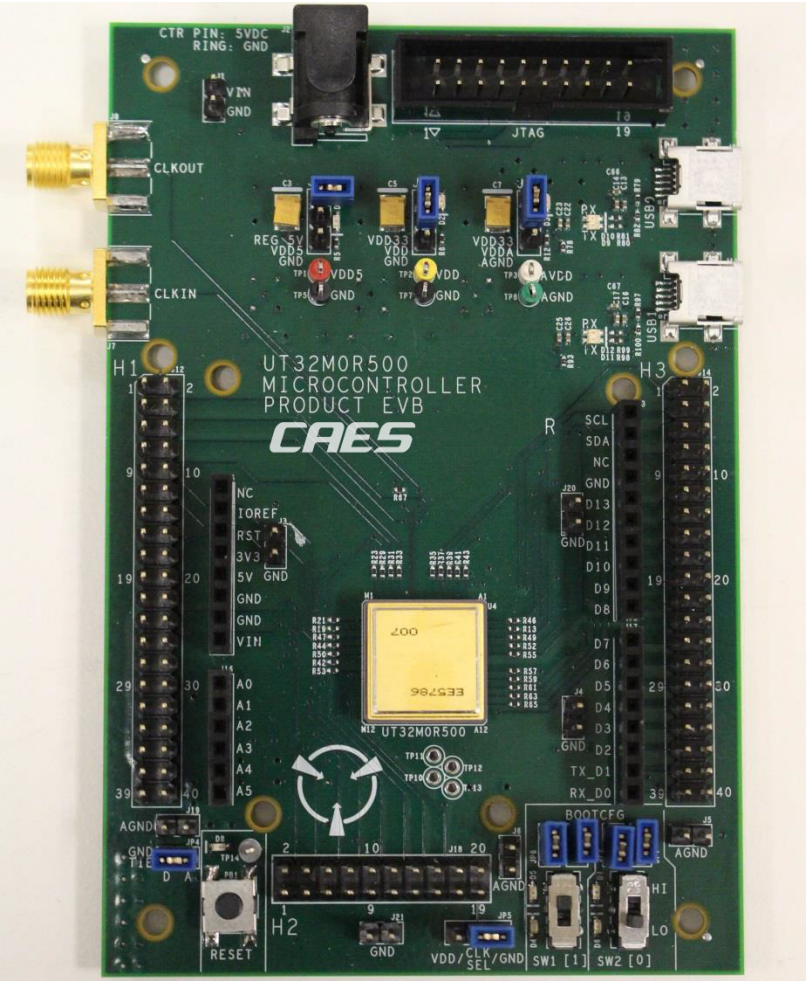


Figure 1: UT32M0R500 Evaluation Board

UT32M0R500-EVB Users Guide

2.0 Reference Documents

Description	Reference Document
UT32M0R500 Datasheet	https://caes.com/sites/default/files/documents/Datasheet-UT32M0R500.pdf
ARM Keil ULINK2 Hardware Debugger	http://www2.keil.com/mdk5/ulink

3.0 Block Diagram Description and Picture

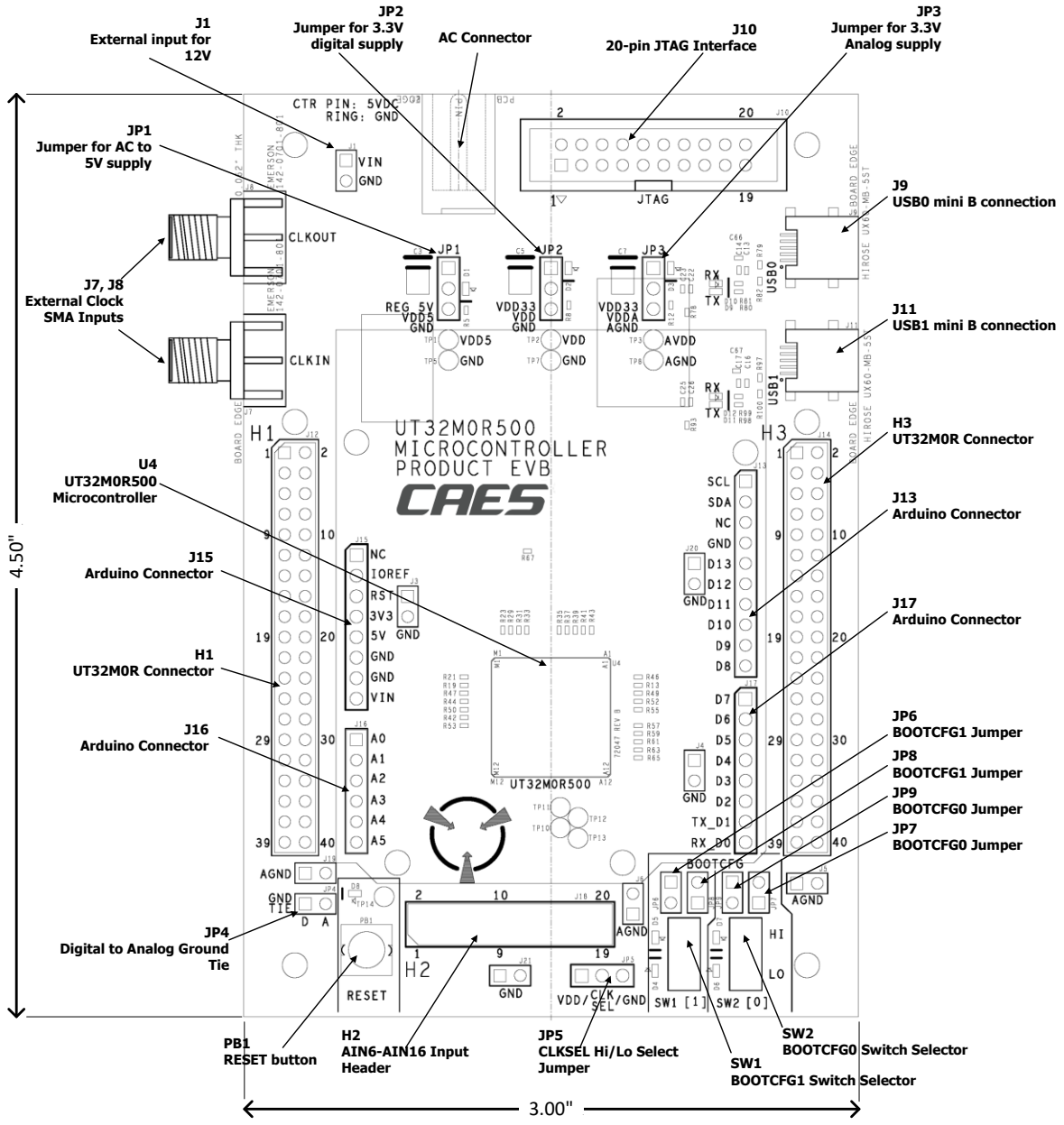


Figure 2: UT32M0R500 EVB Description

UT32M0R500-EVB Users Guide

4.0 Jumper and Switch Setting Summary

Jumper	Setting	Description/Comments
JP1	Shunt Pin 1 to 2 → Connect 3.3V Digital Supply	Shunt to provide power to board from the VIN pin. If implemented, make sure to disconnect the AC wall plug.
JP2	Shunt Pin 1 to 2 → Connect 3.3V Digital Supply	Required for device operation
JP3	Shunt Pin 1 to 2 → Connect 3.3V Analog Supply	Required for device operation
JP4	Shunt Pin D to A	Connects digital and analog grounds
JP5	Shunt Pin 1 to 2 → Connect CLKSEL pin to VDD Shunt Pin 2 to 3 → Connect CLKSEL pin to GND	Shunt required for proper operation, see Clock Source Options
JP6	Shunt for BOOTCFG1 → Connect LEDs to VDD	Provides power to LEDs connected to BOOTCFG1 and power for SW1. JP8 also required.
JP7	Shunt for BOOTCFG0 → Connect SW0 to U4	JP9 also required.
JP8	Shunt for BOOTCFG1 → Connect SW1 to U4	JP6 also required.
JP9	Shunt for BOOTCFG0 → Connect LEDs to VDD	Provides power to LEDs connected to BOOTCFG0 and power for SW1. JP7 also required.
J1	External connection for 12V input to VIN signal	
J7	SMA connector for CLKOUT signal	JP5 must be connected to VDD when applying clock signal to this pin.
J8	SMA connector for CLKIN signal	JP5 must be connected to VDD when applying clock signal to this pin.
J9	USB mini-B connector for USB0	Connection for communicating to U4 over USB through UART0
J10	20-pin JTAG interface connector	
J11	USB mini-B connector for USB1	Connection for communicating to U4 over USB through UART1
J13	Arduino connector	
J14	Arduino connector	
J16	Arduino connector	
J17	Arduino connector	
H1(J12)	Connector for U4	
H2(J18)	20-pin header for AIN6 – AIN15 input	
H3(J14)	Connector for U4	
SW1	Toggle switch for BOOTCFG1 selection	
SW2	Toggle switch for BOOTCFG0 selection	
PB1	RESET Button	
U4	UT32M0R500 device	

5.0 Power Supply and Power Selections

The UT32M0R500-EVB has two options for providing power to the board. The first option is to provide power via the AC wall supply provided with the development board. The second option is to provide 7V – 12V to VIN (J1).

NOTE: Make sure you only use one of the mentioned methods

UT32M0R500-EVB Users Guide

6.0 Boot Configuration Options

The UT32M0R500 device has three different boot modes configured through the BOOTCFG pins. The UT32M0R500-EVB supports the control of the BOOTCFG pins through two methods. The first method is via SW1 and SW0. To use SW1 and SW0, jumpers JP6 – JP9 must be in place. The second method that allows for control of the BOOTCFG pins is through H2 (pins 17 and 19).

Boot Mode Selection Pins		Boot Mode	Description
BOOTCFG1	BOOTCFG2		
0	0	0	Load image from internal Flash memory into SRAM and execute
0	1	1	Reserved
1	0	2	Load/Update image over UART0 into flash (reset required)
1	1	3	Load/Update image over CAN0 into flash (reset required)

NOTE: For control through H2, make sure to remove jumper JP6 – JP9.

7.0 Clock Source Options

The UT32M0R500-EVB supports all clocking options for the UT32M0R500 microcontroller. There is the option to use the internal clock source or use an external source. This is determined by the CLKSEL pin, which is controlled by JP5. For the external clock source, the UT32M0R500 can utilize a clock signal (square wave with 50% duty cycle) or crystal oscillator input. In the case of the external clock source, the UT32M0R500-EVB can support a clock source by connecting to the SMA connectors (J7 and J8). Another option is populate the board with a crystal oscillator and support circuitry.

CLKSEL	Description
0	Selects internal clock
1	Selects the External Crystal Source External clock of crystal oscillator or clock signal on CLKIN support

8.0 Programming and Debugging Interface

The UT32M0R500-EVB supports programming and communicating with the microcontroller over UART. For programming the microcontroller, the UART0 peripheral is used. To facilitate communicating over UART from a PC, the EVB includes to USB-to-UART converters connected on USB0 and USB1 for UART0 and UART1 respectively. Where both UARTs can be utilized for communication, only UART0 (via USB0) can be used for programming.

The UT32M0R500-EVB supports debugging through the 20-pin JTAG (J10) interface. To program the UT32M0R500 over JTAG only the ARMKeil ULINK2 hardware debugger is officially supported.

UT32M0R500-EVB Users Guide

8.1 Creating a Project with Keil µVision IDE

1) Launch Keil uVision
2) From the Project menu, select New uVision Project….
3) Under the directory of choice, specify the project name as helloword and click Save, see Figure 3.

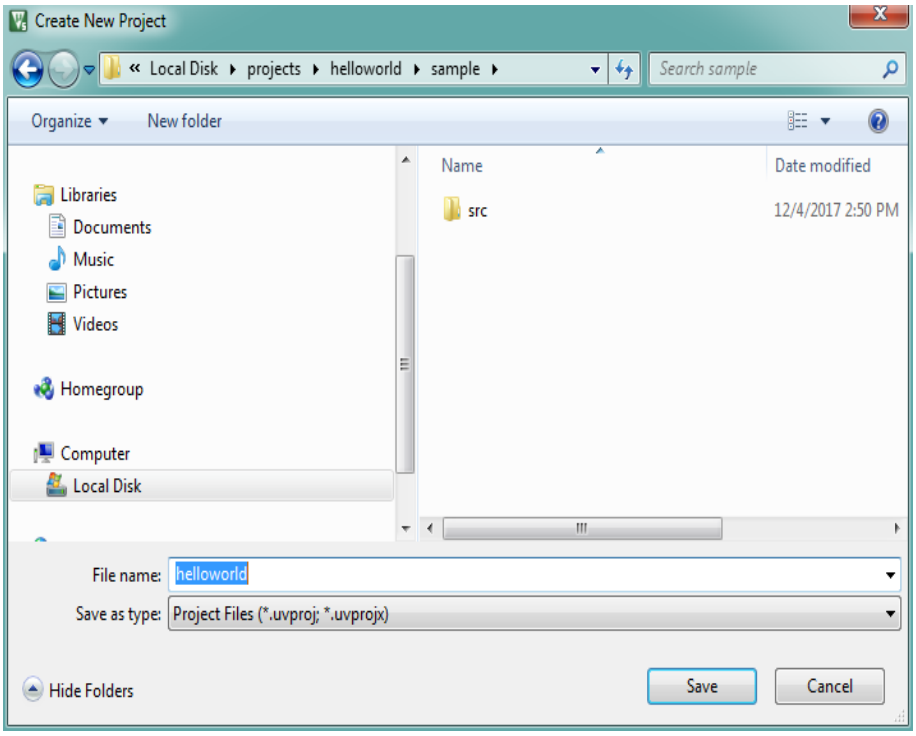


Figure 3: Project Setup

4) Select **Device** and click **OK**, see figure 4.

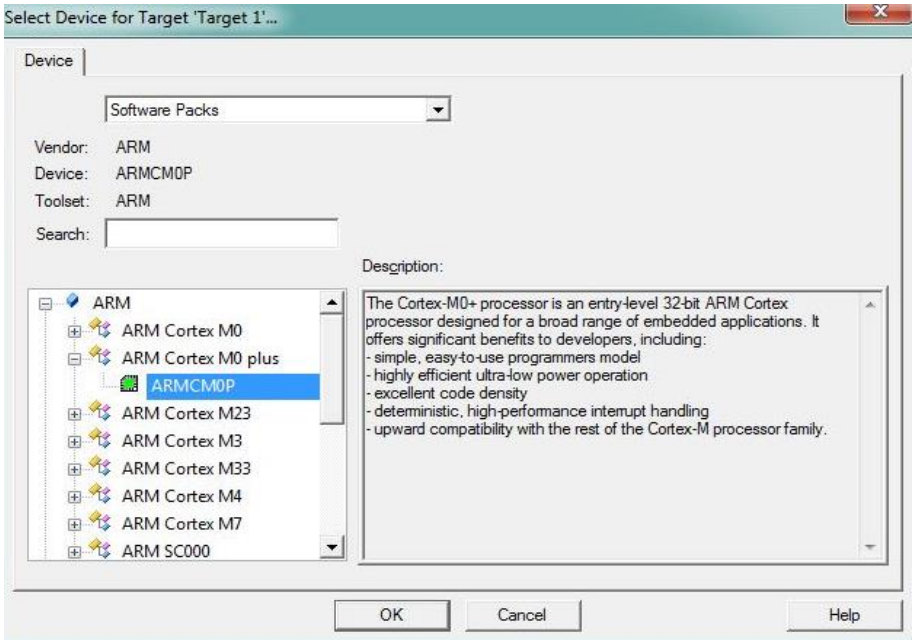


Figure 4: Select Device

UT32M0R500-EVB Users Guide

5) Click the **Manage Run-Time Environment** symbol and under **Software Component**, select the appropriate components and click **OK**, see Figure 5.

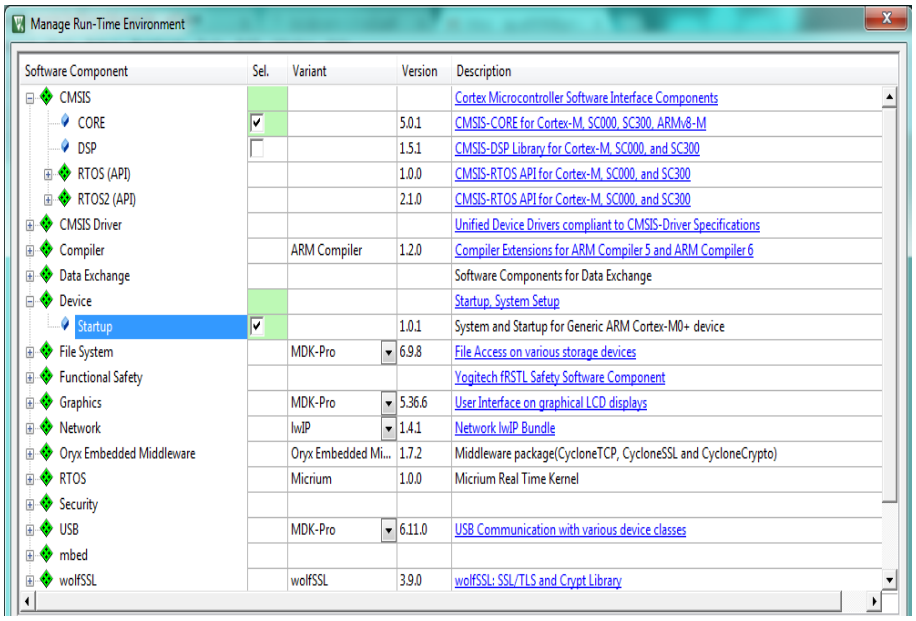


Figure 5: Software Components

6) Under the folder where the project was created, create a **src** folder for the **.c** files. In the **Project**, double-click **Source Group 1** and rename it to **hello_src**.

7) Right-click on **hello_src** and click on **Add New Item** to **Group 'hello_src'**…. Add a new **C** source file, **hello_test.c** and copy the following code.

```
#include <stdio.h>
#include "UT32M0R500.h"
#include "ut32m0_uart.h"

UART_TypeDef *UART0 = (UART_TypeDef *) UART0_BASE;
UART_InitTypeDef UART_InitStruct;
uint32_t ActualBaudRate;

int main (void){
   UART_StructInit (&UART_InitStruct);
   ActualBaudRate=UART_Init (UART0, &UART_InitStruct);
   UART_Cmd (UART0, ENABLE, ENABLE);

      for(;;){
      printf("Hello World!!!\r\n");
      }

}
```

UT32M0R500-EVB Users Guide

8) Right-click on **Target1** and select **Add Group**… to create groups for source and include files for Cobham's Standard Peripheral Library. Add sources and include files to their respective directories, see **Figure 6**

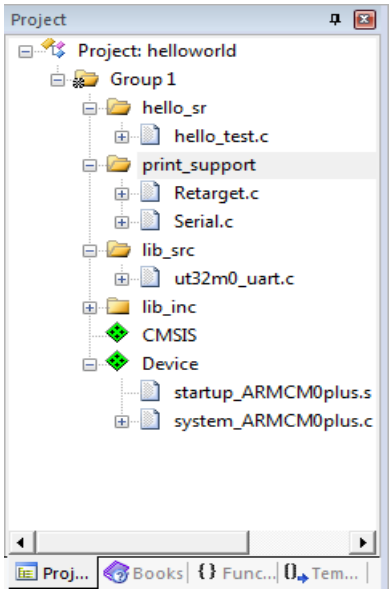


Figure 6: Add Source and Include Files

9) Right-click on **Target1** and select **Options for Target 'Target 1'**…. see **Figures 7-17** for basic settings—

Change setting according to the particular project. For **C/C++** and **Asm** tabs, click  and setup the compiler include paths; see **Figure 11** and **Figure 12**. Leave the other tabs with defaults.

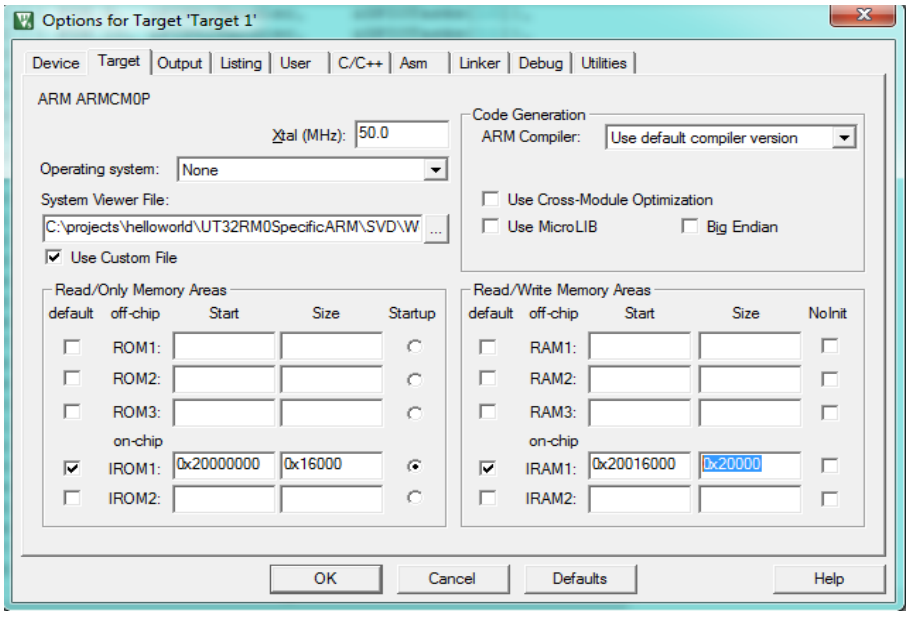


Figure 7: Target

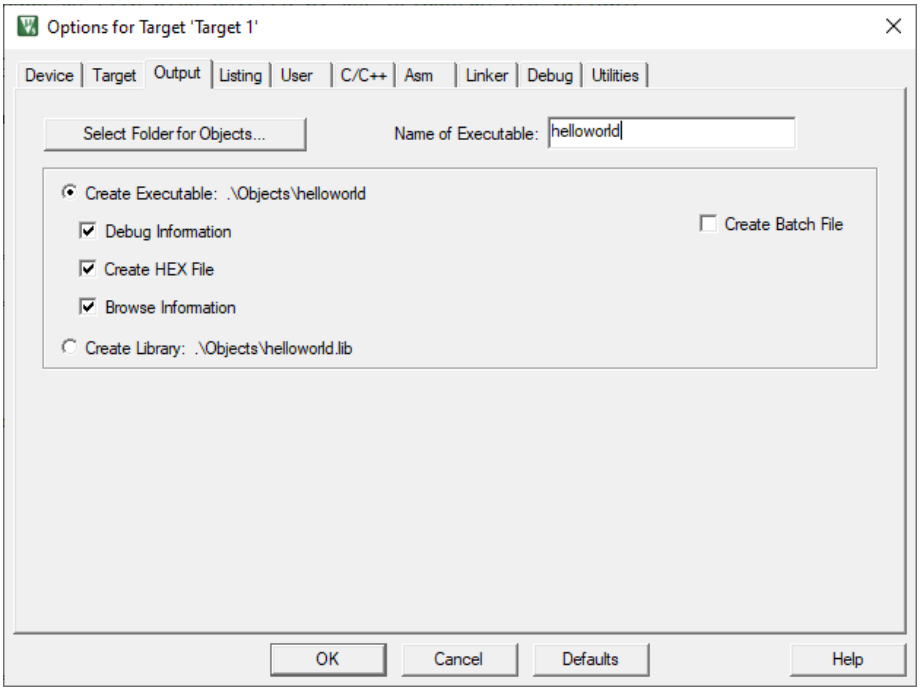


Figure 8: Output

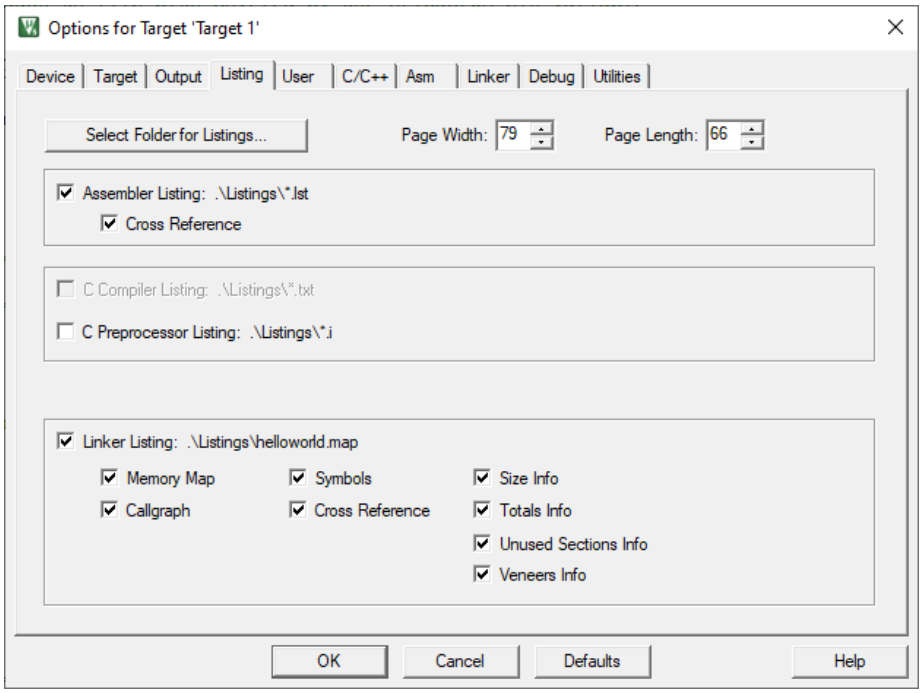


Figure 9: Listing

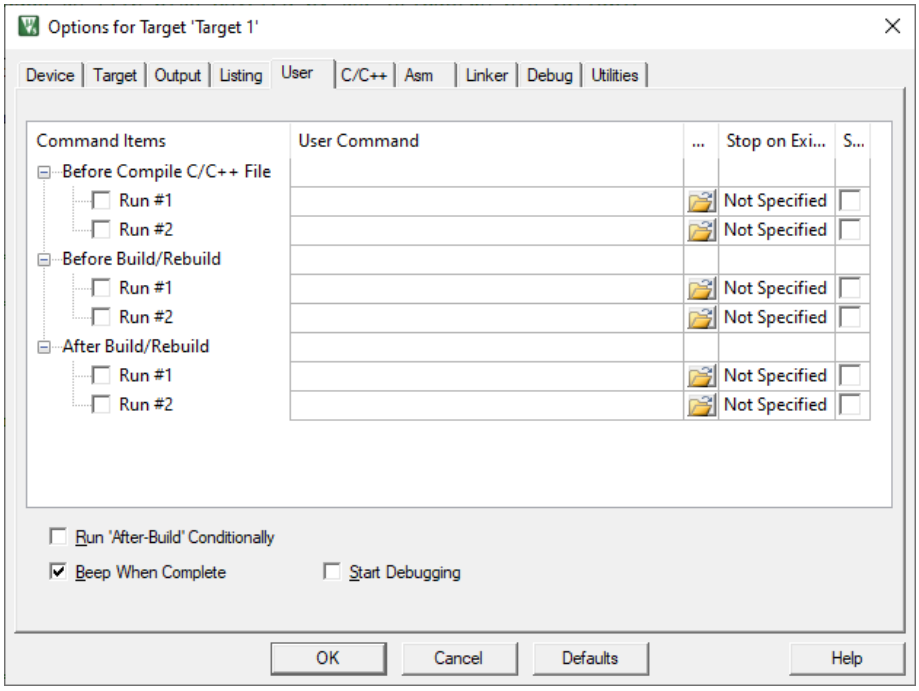


Figure 10: User

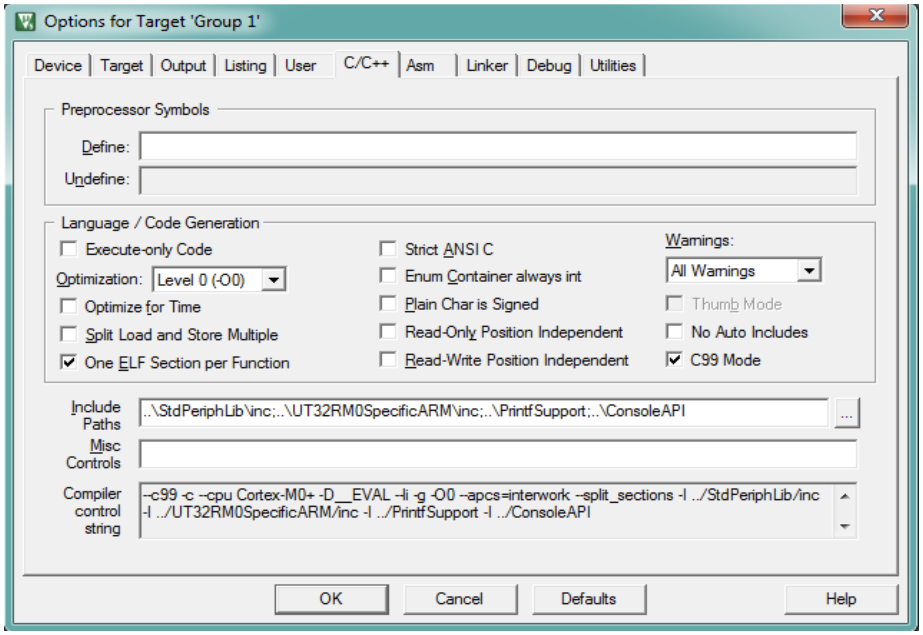


Figure 11: C/C++ Include Paths

UT32M0R500-EVB Users Guide

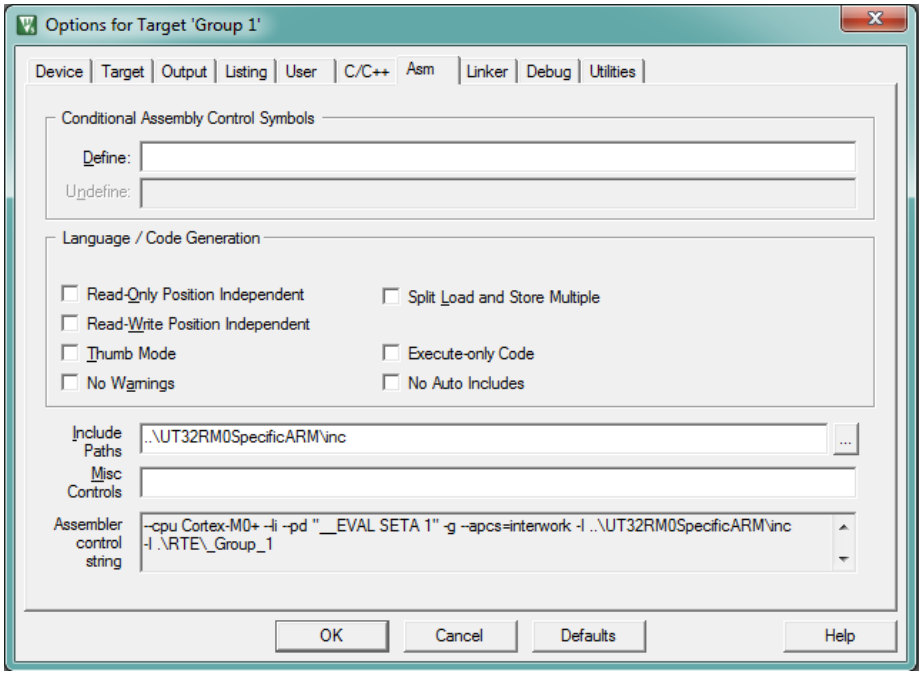


Figure 12: ASM Include Paths

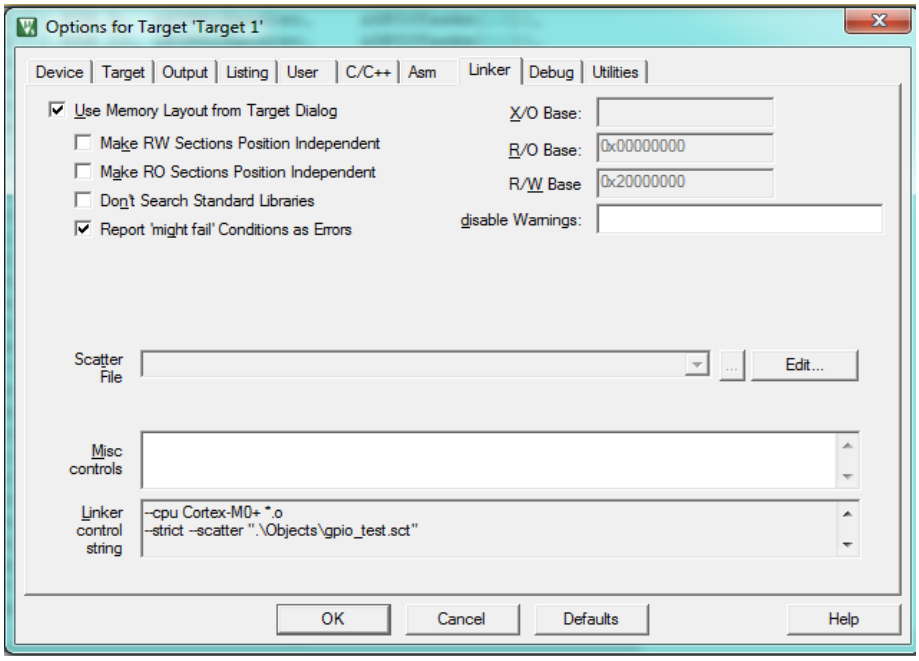


Figure 13: Linker

UT32M0R500-EVB Users Guide

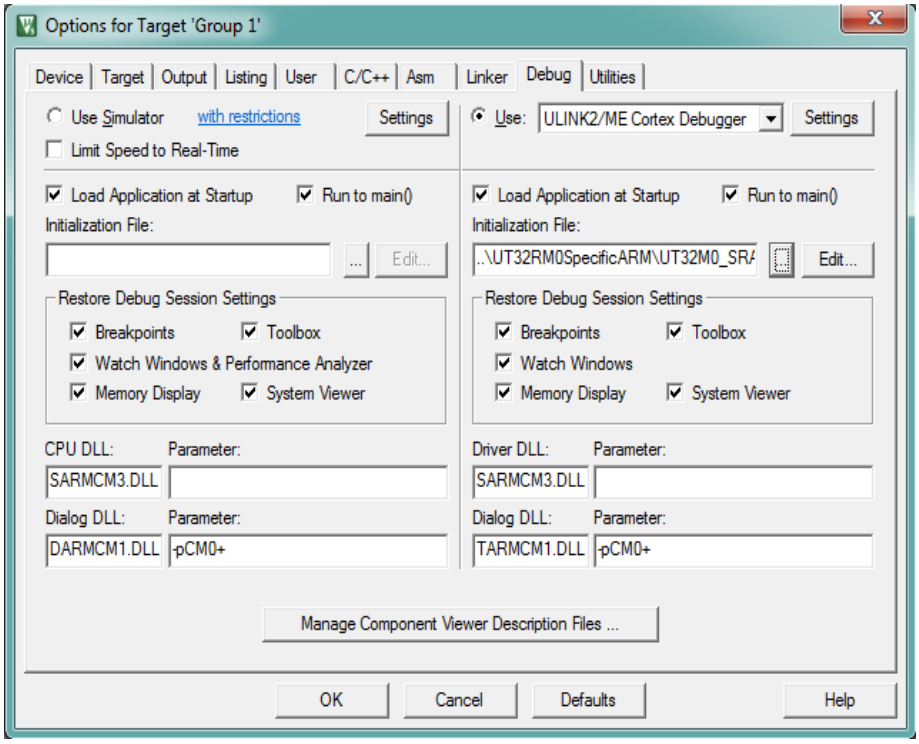


Figure 14: Debug

To get to **Figures 15-17**, click **"Settings"** in the top right of **Figure 14**

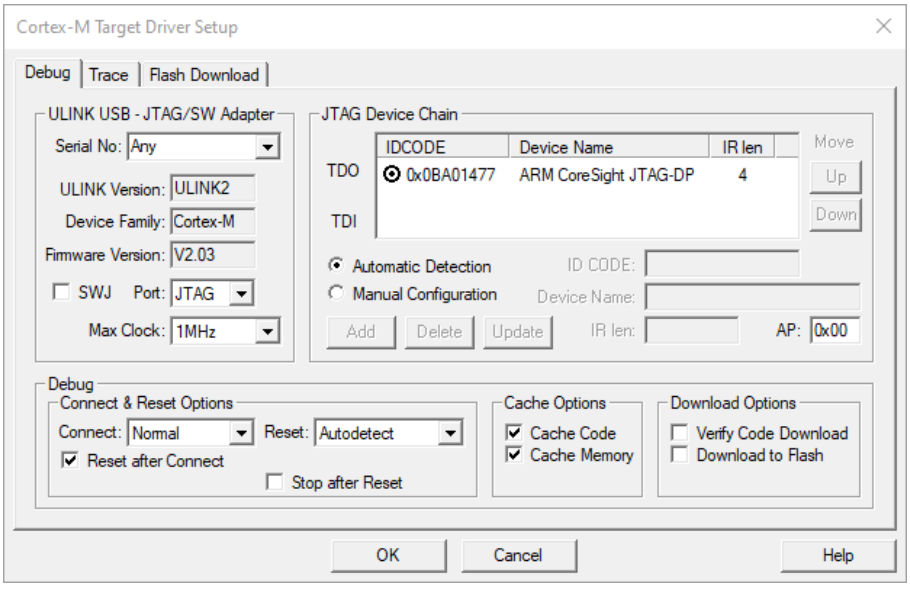


Figure 15: Debug JTAG Settings

UT32M0R500-EVB Users Guide

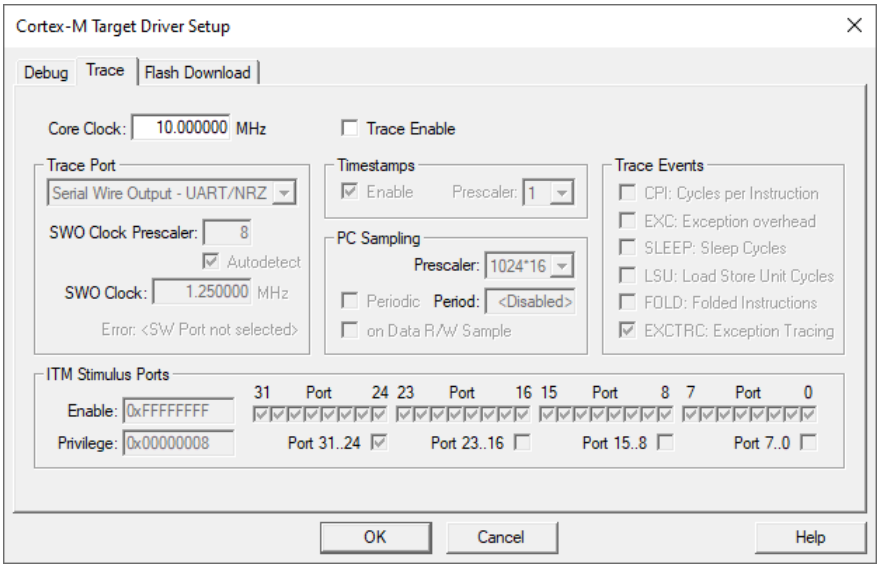


Figure 16: Trace JTAG Settings

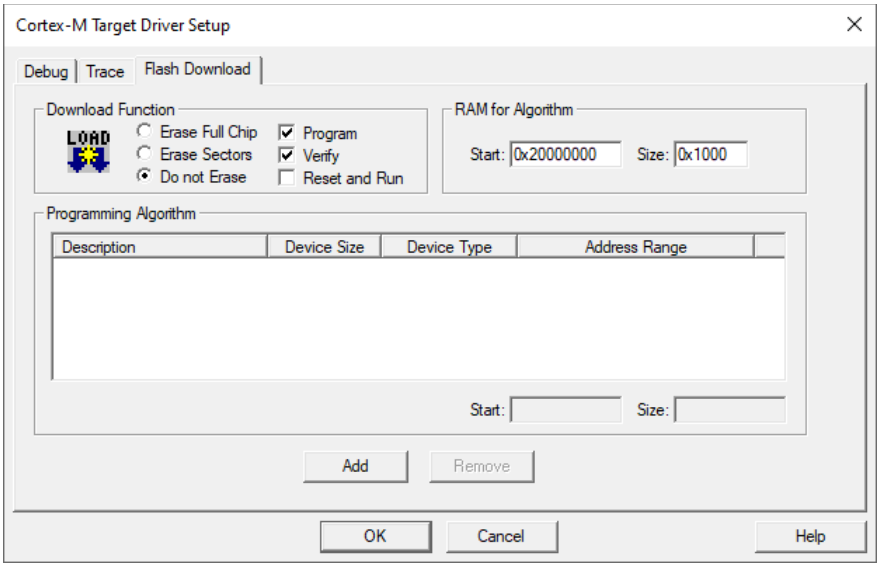


Figure 17: Flash Download JTAG Settings

10) In the **Project Explorer** view, click on  and **Build Project**.

11) Start the debugger  and run  the application. Display the output using a Terminal, see **Figure 12**.

UT32M0R500-EVB Users Guide

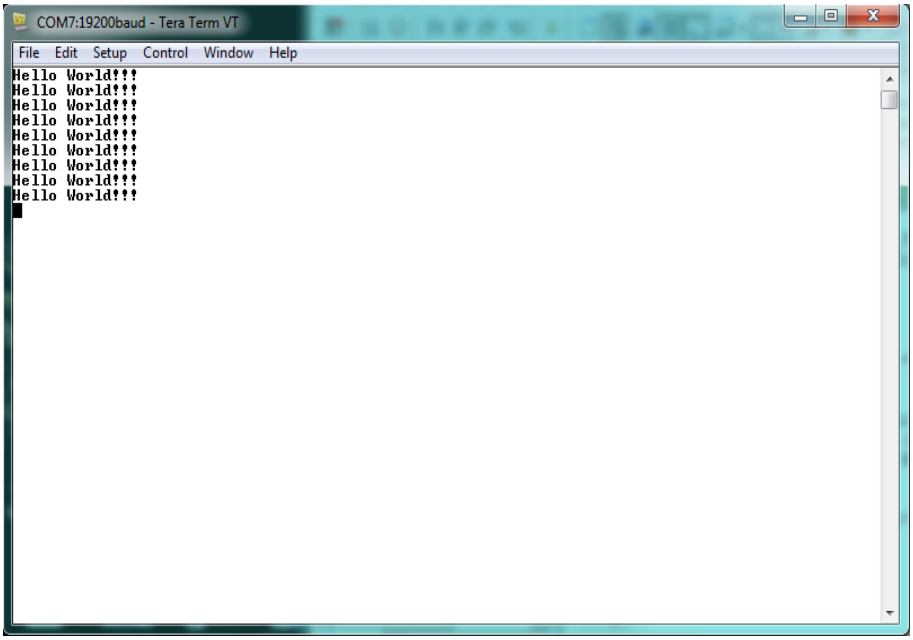


Figure 18: Hello World Display

8.2 UART Terminal Configuration

To program the UT32M0R500 via UART, confirm that the switches for BOOTCFG are in the b'10 position. This enables the UART interface (UART0: UART0_RXD and UART0_TXD) peripheral. The UART0 is initialized to operate at 19200 baud, with 8 databits, 1 stop bit, no parity, and flow control off.

During the UART0 firmware update process, the UT32M0R500's BootRom expects an Intel Hex record file to be uploaded. Depending on the host terminal emulator, the carriage return (0x0D) and line feed (0x0A) characters may be deleted in each line. These characters are required for successful upload. To avoid this, it is recommended that the Intel Hex record be uploaded in the 'binary' mode (as opposed to ASCII).

To prevent overrunning the UT32M0R500's UART receiver, two features need to be enabled prior to file upload. First, "line pacing" should be set to 10 milliseconds (ms). Second, XON/XOFF (software) flow control should be enabled during the update.

UT32M0R500-EVB Users Guide

8.3 Uploading a HEX File Via UART

Before sending an Intel Hex record file, ensure you have a proper connection established by pushing the RESET button (PB1). You should see the following output:

```
Updating from UART...

Welcome to the Cobham AES UT32M0R500 BootROM
Enter a '?' for the user menu

:>
```

You can now see the menu of commands by sending '?' or just hitting return. For this example, the Intel Hex record file will be written into the NOR Flash in image 0. To access the NOR Flash, send 'DEV -tN' to set your target device. Then, send 'IMG -n0' to select image 0.

```
:>IMG -n0

NFC init SUCCESS!
```

To ensure previous uploads don't interfere with this upload, send the command 'ERS' to erase the current image. To check if 'ERS' was successful, send the command 'VFY'. Don't worry about any ERROR message, just make sure that Embedded = 0xFFFF. You can now program your board with 'PGM –fH'. You will see:

```
:>PGM -fH

Send/upload image (hex) file now.
Be sure to enable all THREE of the following features:
  (a) XON/OFF software flow control
  (b) 10ms line pacing
  (c) binary mode
```

You can upload your file (check the above features are enabled). If the file uploads successfully, you will see:

If the upload has no errors, send VFY again. Take note of the Calculated value.

```
:>VFY

ERROR: CRC mismatch. Calculated = 0xCDF1, Embedded = 0xFFFF
```

Using the Calculated value, send 'CRC –c<value>'.

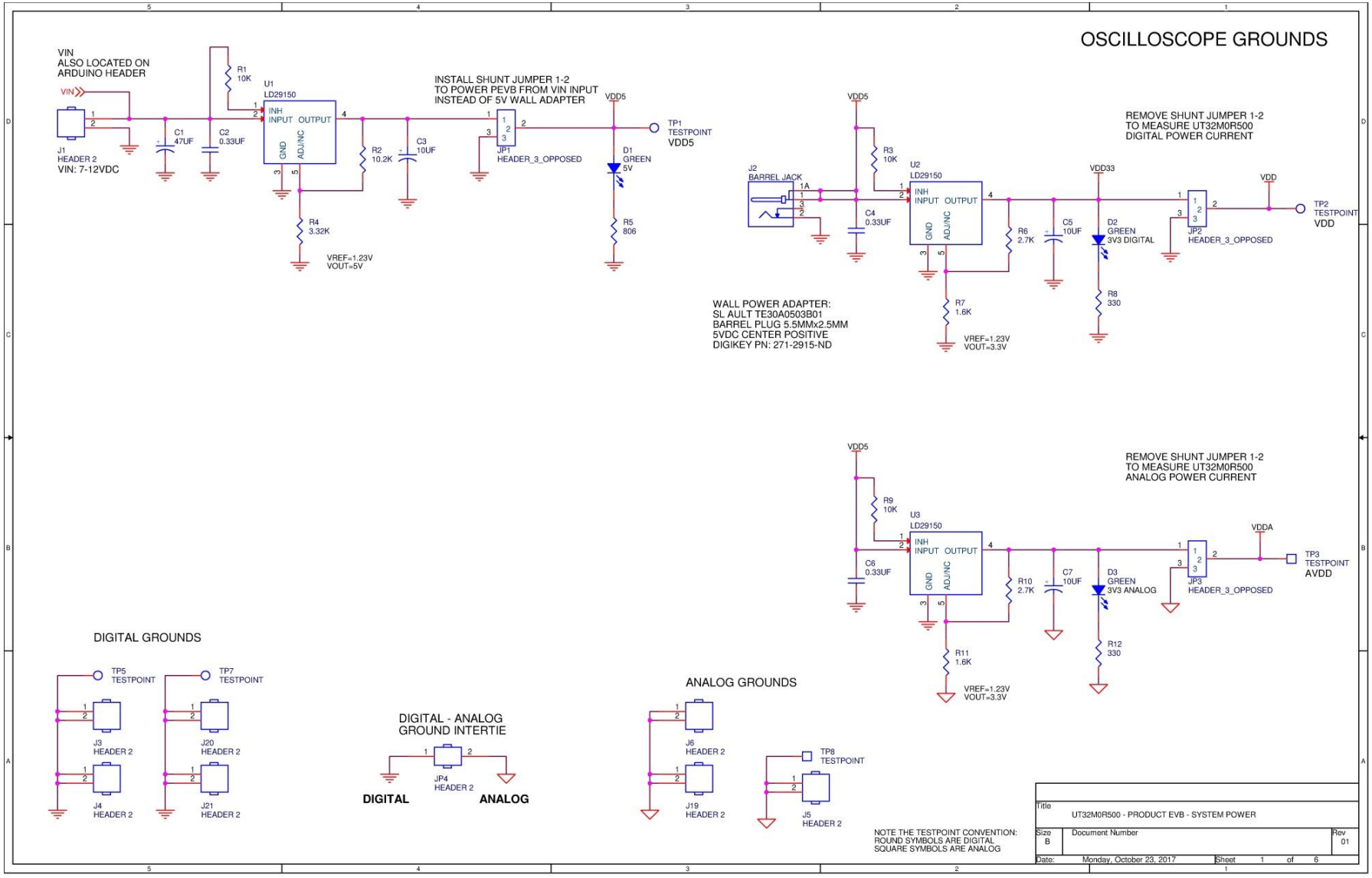
```
:>CRC -cCDF1

SUCCESS!!  CRC programmed correctly
```

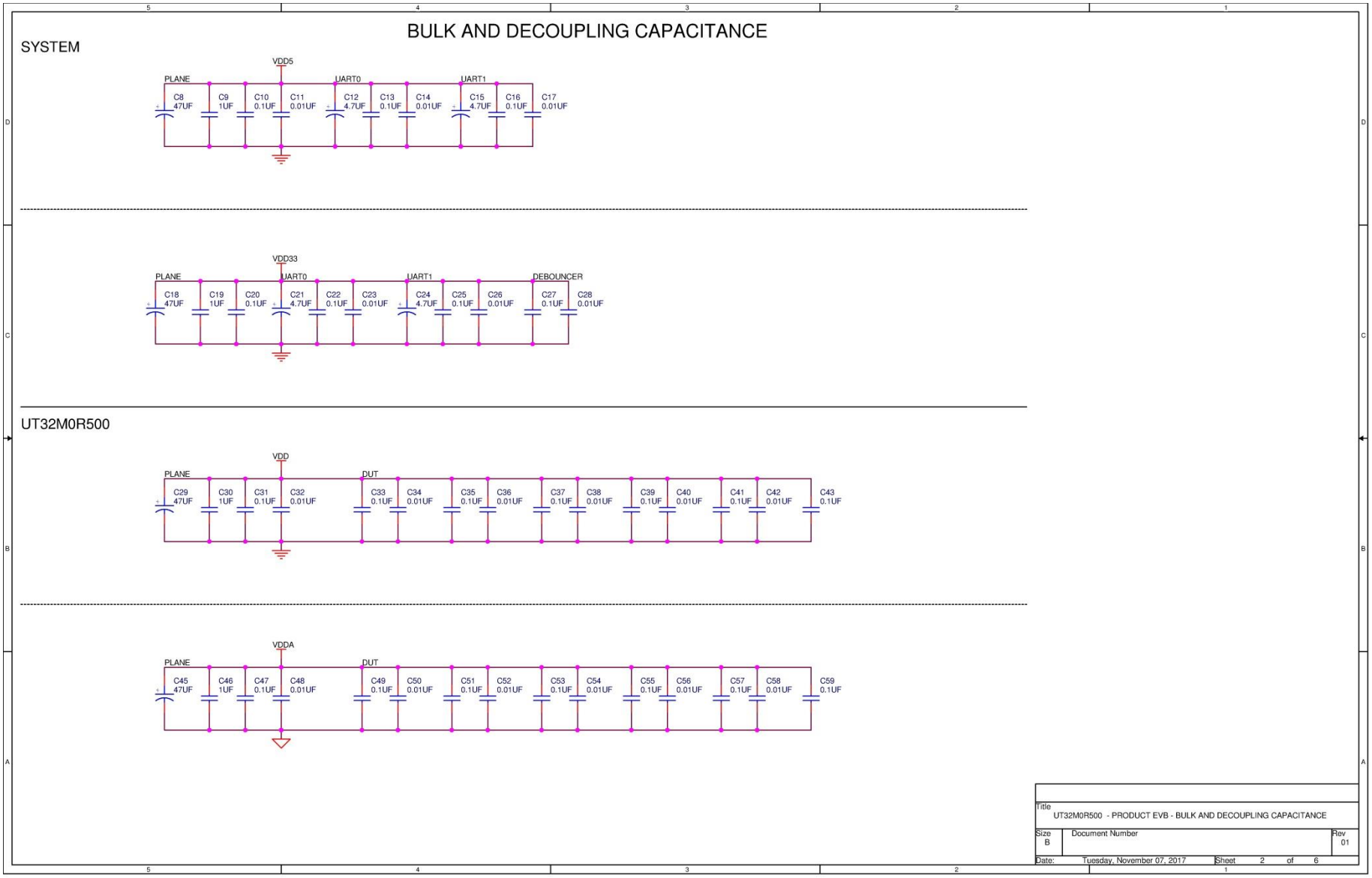
Finally send 'VFY' again. This time, you should see that CRC matches the expected value. You may now change BOOTCFG to b'00 and hit RESET (PB1) to run your program.

UT32M0R500-EVB Users Guide

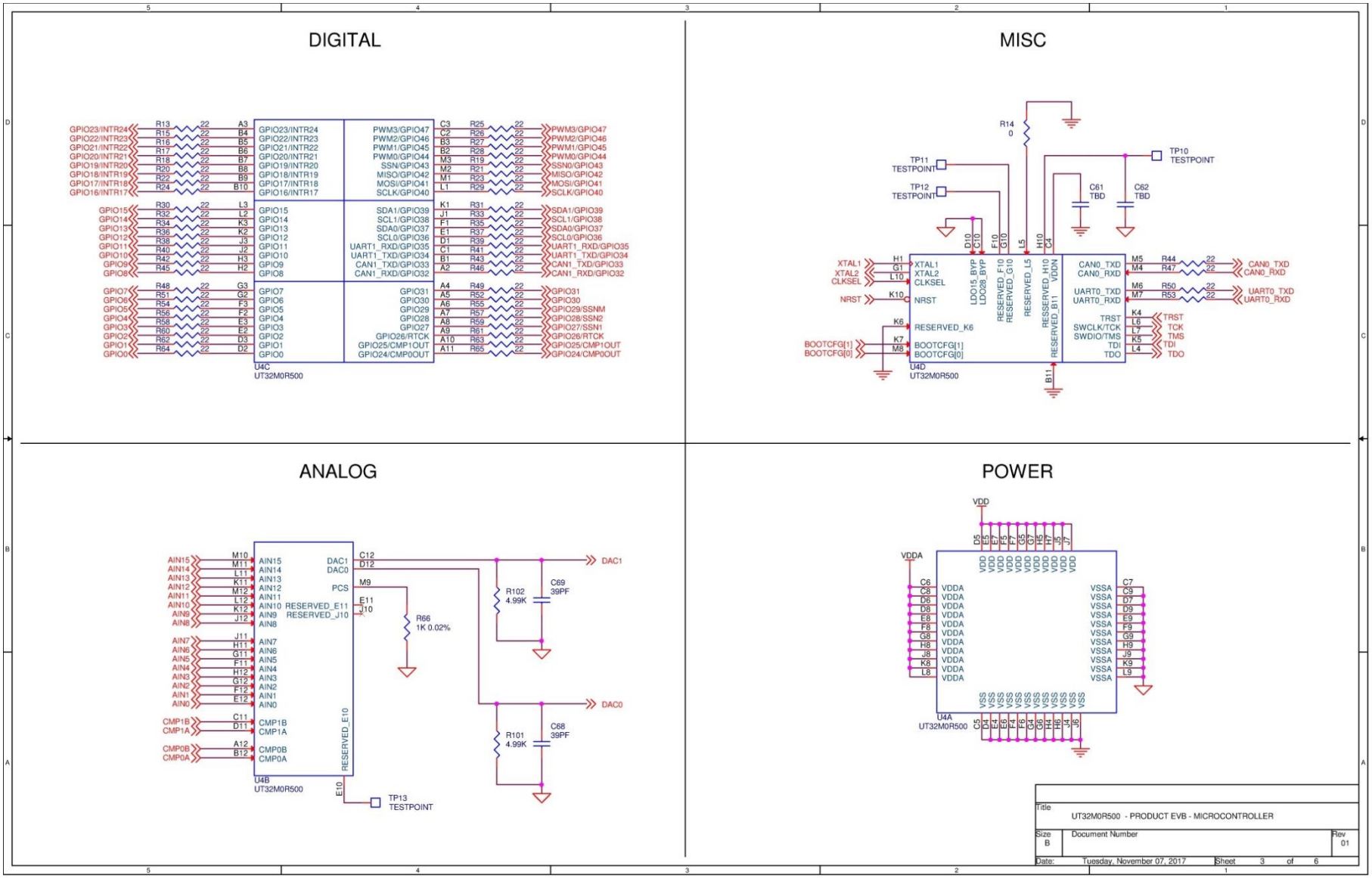
9.0 UT32M0R500-EVB Electrical Schematics



UT32M0R500-EVB Users Guide

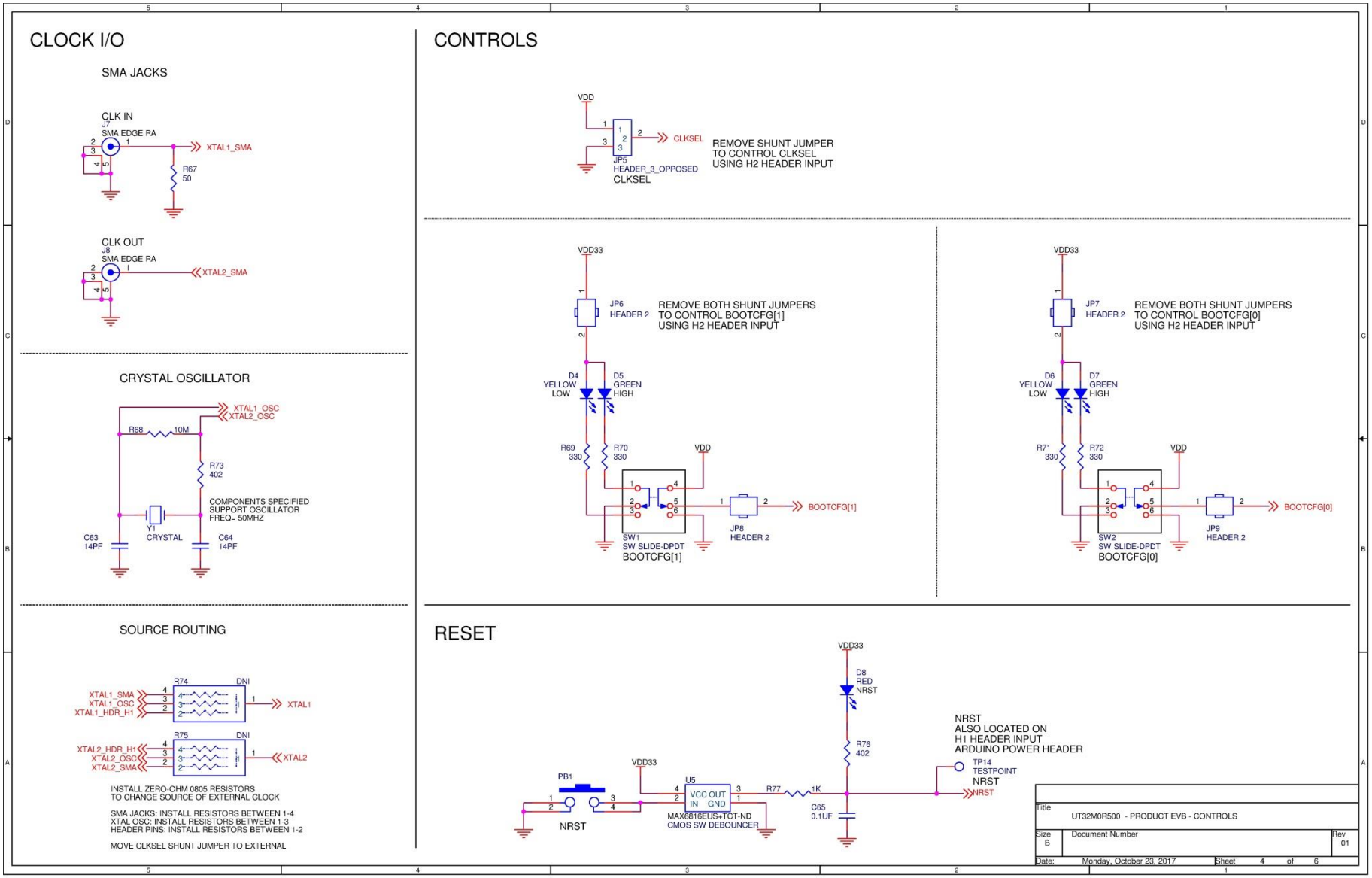


UT32M0R500-EVB Users Guide



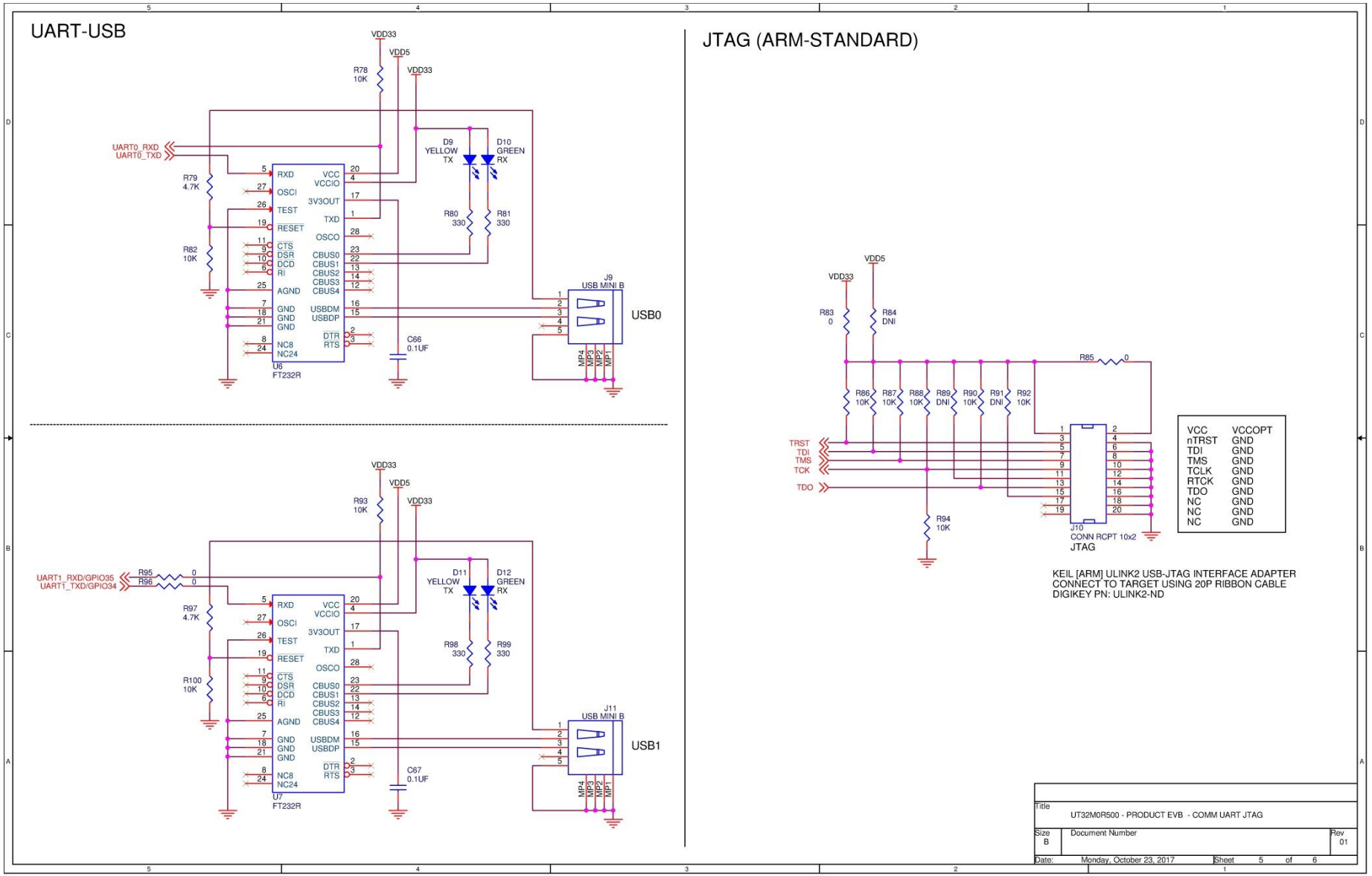
CAES PIONEERING ADVANCED ELECTRONICS

UT32M0R500-EVB Users Guide

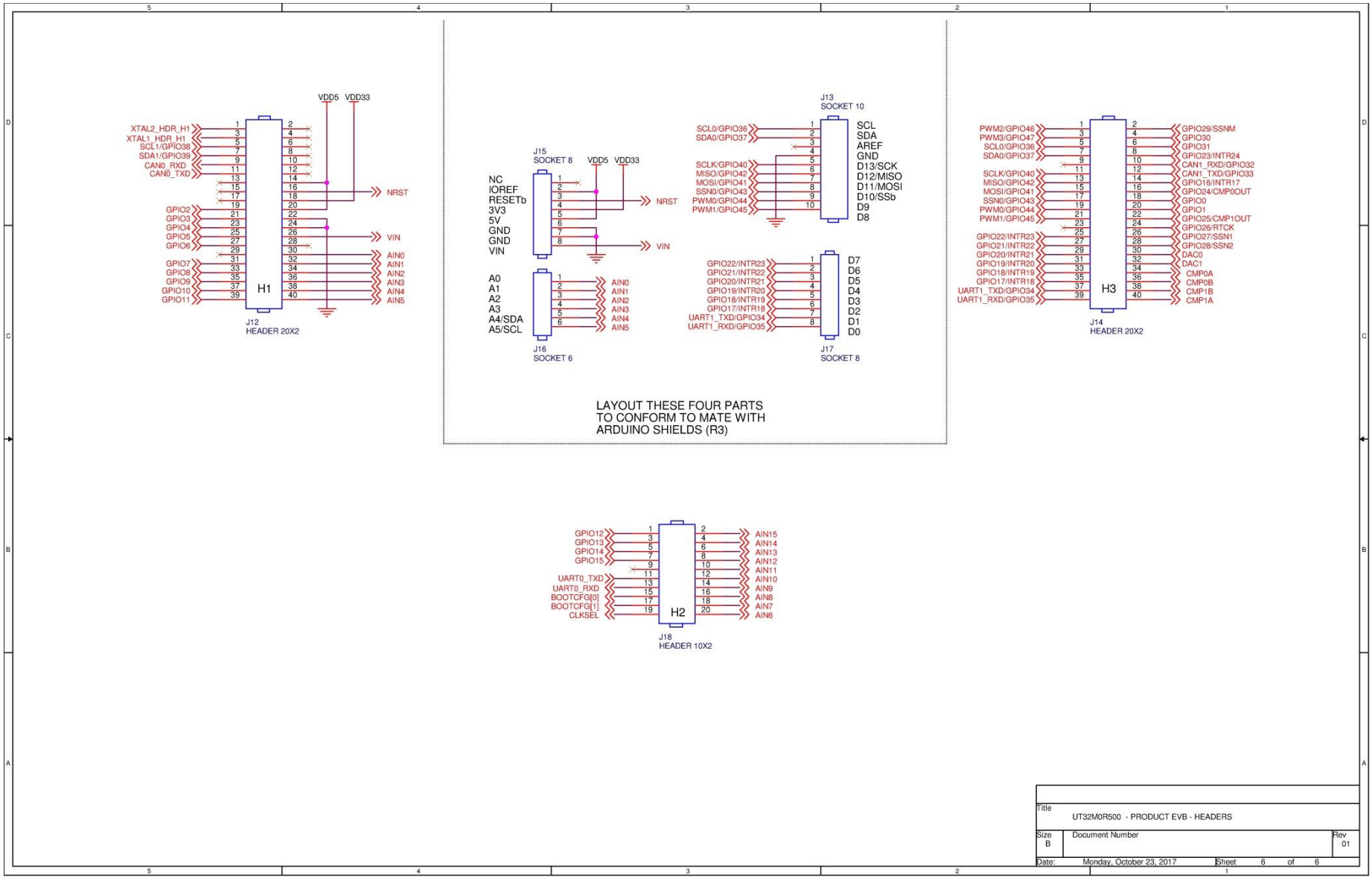


CAES PIONEERING ADVANCED ELECTRONICS

UT32M0R500-EVB Users Guide



UT32M0R500-EVB Users Guide



CAES PIONEERING ADVANCED ELECTRONICS

UT32M0R500-EVB Users Guide

REVISION HISTORY

Date	Revision	Author	Change Description
Nov 2017	0.0.1	OW/AW	DRAFT
Dec 2017	0.0.2	OW/AW	DRAFT REVISON
Dec 2017	0.0.3/4	OW/AW	Added information on setting up a project and running a sample program. Added information on how to program over UART.
Dec 2017	0.1.0	OW/AW	Draft release.
Mar 2018	1.0.0	OW/AW/JA	Initial Release
5/20/2021	1.1.0	OW	Updated template; Added the remaining 'Target 1' generic settings pictures, including the JTAG/Cortex M setup

The following United States (U.S.) Department of Commerce statement shall be applicable if these commodities, technology, or software are exported from the U.S.: These commodities, technology, or software were exported from the United States in accordance with the Export Administration Regulations. Diversion contrary to U.S. law is prohibited.