1 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.
2 REFERENCE DOCUMENTS

<table>
<thead>
<tr>
<th>Description</th>
<th>Reference Document</th>
</tr>
</thead>
<tbody>
<tr>
<td>UT32M0R500 Datasheet</td>
<td><a href="http://ams.aeroflex.com/pagesproduct/datasheets/UT32M0R500_Advanced_Datasheet.pdf">http://ams.aeroflex.com/pagesproduct/datasheets/UT32M0R500_Advanced_Datasheet.pdf</a></td>
</tr>
<tr>
<td>ARMKeil ULINK2 Hardware Debugger</td>
<td><a href="http://www2.keil.com/mdk5/ulink">http://www2.keil.com/mdk5/ulink</a></td>
</tr>
</tbody>
</table>

3 BLOCK DIAGRAM DESCRIPTION/PICTURE

![Diagram of the UT32M0R500 Microcontroller EVB](image-url)
## JUMPER AND SWITCH SETTING SUMMARY

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

6  BOOT CONFIGURATION OPTIONS
The UT32M0R500 device has three different boot modes configured through the BOOTCFG pins. The

<table>
<thead>
<tr>
<th>Boot mode selection pins</th>
<th>Boot Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>0</td>
<td>0: Load image from internal Flash memory into SRAM and execute</td>
</tr>
<tr>
<td>0 1</td>
<td>1</td>
<td>1: Reserved</td>
</tr>
<tr>
<td>1 0</td>
<td>2</td>
<td>2: Load/Update image over UART0 into flash (reset required)</td>
</tr>
<tr>
<td>1 1</td>
<td>3</td>
<td>3: Load/Update image over CAN0 into flash (reset required)</td>
</tr>
</tbody>
</table>

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).


7  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.

<table>
<thead>
<tr>
<th>CLKSEL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Selects internal clock</td>
</tr>
<tr>
<td>1</td>
<td>Selects external clock source.</td>
</tr>
<tr>
<td></td>
<td>External clock of crystal oscillator or clock signal on CLKin support</td>
</tr>
</tbody>
</table>
8 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 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.

8.1 CREATING A PROJECT WITH KEIL UVISION 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 helloworld and click Save, see Figure 1.

![Figure 1: Project Setup]

4) Select Device and click OK, see figure 2.
5) Click the **Manage Run-Time Environment** symbol and under **Software Component**, select the appropriate components and click **OK**, see Figure 3.

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.

**Table 1.1. Helloworld Source Code**

```c
#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");
    }
}
```

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 4**.

![Figure 4: Add source and include files](image-url)
9) Right-click on **Target1** and select **Options for Target 'Target 1'**…. see **Figures 5-9** 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 6** and **Figure 7**. Leave the other Tabs with defaults.
Figure 7: ASM Include Paths

Figure 8: Linker
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 10.
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.

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:

```
Welcome to the Cobham NES 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:

```
Programming complete -- check progress stream for any 'E/F/2/3' (error)
```

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.
Figure 12: 2/6
Figure 14: 4/6

CLOCK I/O

SMA JACKS

CLK IN
SMA EDGE NA
XTAL1_SMA

CLK OUT
SMA EDGE NA
XTAL2_SMA

CRYSTAL OSCILLATOR

COMPONENTS SPECIFIED SUPPRESS OSCUILLATION FREQUENCY

SOURCE ROUTING

INSTALL ZERO OHM RESISTORS TO GROUND SOURCE OF EXTERNAL CLOCK

SMA JACKS: INSTALL RESISTORS BETWEEN 1-3, XTAL1 and XTAL2, SERIER RESISTORS BETWEEN 1-3 AND 4.
MOVE SHADE SHUNT JUMPER TO EXTERNAL

CONTROLS

REMOVE SHUNT JUMPER TO CONTROL CLOCK USING H2 HEADER INPUT

HEADER 3 IMPEDED
CLKSEL

REMOVE BOTH SHUNT JUMPERS TO CONTROL BOOTCF0(1) USING H2 HEADER INPUT

JPT HEADER 2

RESET

NRST LOCATED ON H1 HEADER INPUT

ARDUINO POWER HEADERS

NRST
Figure 15: 5/6
Figure 16: 6/6
## 10 REVISION HISTORY

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Change Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nov 2017</td>
<td>0.0.1</td>
<td>DRAFT</td>
</tr>
<tr>
<td>Dec 2017</td>
<td>0.0.2</td>
<td>DRAFT REVISION</td>
</tr>
<tr>
<td>Dec 2017</td>
<td>0.0.3/4</td>
<td>Added information on setting up a project and running a sample program. Added information on how to program over UART.</td>
</tr>
<tr>
<td>Dec 2017</td>
<td>0.1.0</td>
<td>Draft release.</td>
</tr>
<tr>
<td>Mar 2018</td>
<td>1.0.0</td>
<td>Initial Release</td>
</tr>
</tbody>
</table>
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.