

# BACKWARDS IS FORWARD

Register access defines functionality

```
UART:5002000C UART_UART_LCR_REG % 4
UART:5002000C
UART:5002000C
```

; DATA XREF: sub\_264C+14+r  
; sub\_264C+1A+rw ...  
; RW: Line Control Register

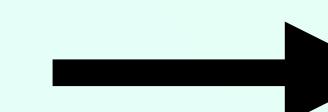
```
1 int __fastcall sub_264C(int result)
2 {
3     int v1; // r3
4
5     UART_UART_SRR_REG = 7;
6     v1 = UART_UART_LCR_REG;
7     UART_UART_LCR_REG = v1 | 0x80;
8     UART_UART_DLF_REG = (unsigned __int8)result;
9     UART_UART_RBR_THR_DLL_REG = BYTE1(result);
10    UART_UART_IER_DLH_REG = BYTE2(result);
11    UART_UART_LCR_REG = 3;
12    UART_UART_IIR_FCR_REG = 7;
13    UART_UART_IER_DLH_REG = BYTE2(result) & 0xFE;
14    return result;
15 }
```

```
1 int __fastcall UART_reset_and_configure_uart(int result)
2 {
3     int v1; // r3
4
5     UART_UART_SRR_REG = 7; // reset UART
6     v1 = UART_UART_LCR_REG;
7     UART_UART_LCR_REG = v1 | 0x80; // set Divisor Latch Access Bit, required to set
8                                // baud rate via DLL/DLH reg
9     UART_UART_DLF_REG = result; // set the rate divisor fractional part - 0x1106
10    UART_UART_RBR_THR_DLL_REG = BYTE1(result); // set the low byte of divisor
11    UART_UART_IER_DLH_REG = BYTE2(result); // set the hibyte of the divisor
12    UART_UART_LCR_REG = 3; // datalen select b11 is 8bits
13    UART_UART_IIR_FCR_REG = 7; // setup read interrupt register
14    UART_UART_IER_DLH_REG = BYTE2(result) & 0xFE;
15    return result;
16 }
```



Function Folders

Function name	Segment	Start	Length
sub_258C	Code	0000258C	00000056
sub_65F4	Code	000065F4	0000005A
sub_BE2	Code	00000BE2	0000005C
sub_264C	Code	0000264C	0000005C
sub_1F46	Code	00001F46	0000005C
sub_2052	Code	00002052	00000064
sub_26AC	Code	000026AC	0000006A
sub_417A	Code	0000417A	00000010
sub_2A9E	Code	00002A9E	00000010
sub_D78	Code	00002D78	00000002
sub_452C	Code	0000452C	00000002



Function name	Segment	Start	Length	Locals
-				
UART				
sub_2720	Code	00002720	0000003A	00000010
sub_258C	Code	0000258C	00000056	00000010
sub_264C	Code	0000264C	0000005C	00000010
sub_26AC	Code	000026AC	0000006A	00000018
NMI_handler	Code	000024D0	00000002	
SVCAll_handler	Code	000024D2	00000002	

# UNICORNS

```
% python booter.py secure_img.bin
invalid register name CYCLECNT, skipping
invalid reg config value: XPSR - 69000000:APSR
invalid reg config value: IPSR - 000(NoException)
invalid register name CFBP, skipping
invalid register name MSPLIM, skipping
invalid register name PSPLIM, skipping
[-] Tracing basic block at Entry - Booter_Flow (0x15e8) - LR 0x1b6 - block size = 0x
[-] Tracing basic block at Entry - CLK_Enable_RC32M (0x155c) - LR 0x15ee - block siz
[+] Read: CRG_TOP + 0x0044 (4 bytes)
[+] Read: CRG_TOP + 0x0014 (4 bytes)
[+] Read: CRG_TOP + 0x0014 (4 bytes)
[+] Read: CRG_TOP + 0x0020 (4 bytes)
[+] Read: CRG_TOP + 0x0028 (4 bytes)
[-] Tracing basic block at Entry - WDOG_feed_ff (0x1544) - LR 0x1638 - block size = 0x
[+] Read: CRG_TOP + 0x0020 (4 bytes)
[+] Read: CRG_TOP + 0x0028 (4 bytes)
[+] Read: CRG_TOP + 0x00f0 (4 bytes)
[+] Read: CRG_TOP + 0x00f0 (4 bytes)
[+] Read: CRG_TOP + 0x0020 (4 bytes)
[+] Read: CRG_TOP + 0x0028 (4 bytes)
[-] Tracing basic block at Entry - OTPC_enable_clock_and_reset (0x1cac)
[+] Read: CRG_TOP + 0x0000 (4 bytes)
[-] Tracing basic block at Entry - OTPC_set_read_mode (0x1dac) - LR 0x1
[-] Tracing basic block at Entry - QSPIC_set_manual_mode (0x2052) - LR
[+] Read: CRG_TOP + 0x0000 (4 bytes)
[+] Read: CRG_TOP + 0x0000 (4 bytes)
[+] Read: CRG_TOP + 0x0020 (4 bytes)
[+] Read: CRG_TOP + 0x0028 (4 bytes)
[-] Tracing basic block at Entry - QSPIC_Software_Reset_peripheral (0x2
[-] Tracing basic block at Entry - QSPIC_Enable_CS_active_low (0x1ee0)
[-] Tracing basic block at Entry - QSPIC_Disable_CS_active_low (0x1ef4)
[-] Tracing basic block at Entry - QSPIC_Enable_CS_active_low (0x1ee0) - LR 0x2192 - block size = 0x14
[-] Tracing basic block at Entry - QSPIC_WriteData_manual_mode (0x2460) - LR 0x2198 - block size = 0x20
QSPI: Release Power-down / Device ID
[-] Tracing basic block at Entry - QSPIC_Disable_CS_active_low (0x1ef4) - LR 0x219c - block size = 0x14
[-] Tracing basic block at Entry - QSPIC_Enable_CS_active_low (0x1ee0) - LR 0x21ba - block size = 0x14
[-] Tracing basic block at Entry - QSPIC_WriteData_manual_mode (0x2460) - LR 0x21c0 - block size = 0x20
QSPI: Enable Reset
```

```
def load_jlink_reg_str(mu, raw_str):
    ...
    loads a raw string from a jlink session of the register state. example:
    PC = 000015E8, CycleCnt = 000011CE
    R0 = 00000000, R1 = 00000000, R2 = 00000002, R3 = 00000000
    R4 = 00000000, R5 = 00000000, R6 = 00000000, R7 = 00000000
    R8 = 22F03812, R9 = 54020200, R10= 20030000, R11= 00000000
    R12= 18846521
    SP(R13)= 20040000, MSP= 20040000, PSP= 00000000, R14(LR) = 000001BB
    XPSR = 69000000: APSR = nZCvQ, EPSR = 01000000, IPSR = 000 (NoException)
    CFBP = 00000000, CONTROL = 00, FAULTMASK = 00, BASEPRI = 00, PRIMASK = 00
    MSPLIM = 00000000
    PSPLIM = 00000000
    ...
    regs = raw_str.strip().replace('\n','').replace(' ','').split(',')
```

```
mu.mmio_map(0x50000000, 4096, crg_top_read, None, crg_top_write, None)
mu.mmio_map(0x50010000, 4096, periph2_read, None, periph2_write, None)
mu.mmio_map(0x50020000, 4096, periph_read, None, periph_write, None)
mu.mmio_map(0x50040000, 4096, periph4_read, None, periph4_write, None)
mu.mmio_map(0x30070000, 4096, Otp.mmio_read, otp, Otp.mmio_write, otp)
mu.mmio_map(0x38000000, 4096, Qspi.mmio_read, qspi, Qspi.mmio_write, qspi)
mu.mmio_map(0x10080000, 4096, Otp.mem_read, otp, Otp.mem_write, otp)
```



<https://www.unicorn-engine.org/>