

Code:

```
.INCLUDE "m128def.inc"
.EQU ZEROS = 0x00
.EQU ONES = 0xFF

.EQU INTN_START = 0x0100      ; starting address for the stack
.EQU INTN_END = 0x10FF

.EQU EXTN_START = 0x1100
.EQU EXTN_END = 0x18FF      ; ending address for the stack
.EQU addr = 0x0000          ; defining an initial address at the
beginning to put the entered address into it later

.CSEG
.ORG 0x0050                  ; starting address for the program

start:
    LDI R16, ZEROS
    OUT DDRA, R16            ; A - input (3-bit data input sequence)
    OUT DDRB, R16            ; B - input (8-bit binary input sequence)
    OUT DDRC, R16            ; C - input (save button)
    OUT DDRE, R16            ; E - input (high byte of the address)
    STS DDRF, R16            ; F - input (low byte of the address)

    LDI R16, ONES
    OUT DDRD, R16            ; D - output (error detection led)
    STS DDRG, R16            ; G - output (External SRAM)

    IN R17, PINA
    IN R18, PINB
    LDI R19, ZEROS
    ADD R19, R17
    LDI R16, ZEROS
    ADD R16, R18

    ANDI R19, 0b11100000      ; masking the 3-input input in R19
    LDI R20, 0b00110101      ; CRC polynomial given (X5+X4+X2+1) filled with 2 0's
at the end
    EOR R18, R20              ; subtraction operation by using XOR
    EOR R18, R20              ; subtraction operation by using XOR
    ANDI R18, 0b00011111      ; masking the value in R18, remainder, to combine with
input A                      ; shift right 3 times in total to get 3-bits at the least
significant bits

    OR R18, R19               ; R18 gets the value that filled with the value in R17
    CP R16, R18               ; compare the entered input and the transmitted one at
the end
```

```

BRNE L1 ; if they are not equal, go to branch L1
RJMP L2 ; else, no error so branch to L1

L1:
SBI PORTD, 1
L2:
SBI PORTD, 0
RJMP NO_ERROR

NO_ERROR:
IN R17, PINE ; high byte for the memory - 8 bits
IN R19, PINF ; low byte for the memory - 8 bits

LDI R21, HIGH(INTN_START) ; R21 = 0x01
LDI R22, HIGH(EXTN_END) ; R22 = 0x18
CP R17, R22
BRLT CONTROL_ADDR ; if valid for the high limit,
branch to CONTROL_ADDR to check the low limit ; if not, go to INVALID_ADDR
RJMP INVALID_ADDR ; if not, go to INVALID_ADDR

INVALID_ADDR:
RJMP start ; go back to the beginning
of the program

CONTROL_ADDR: ; check the low limit and branch
to VALID_ADDR for further operations
CP R17, R21
BRGE VALID_ADDR

VALID_ADDR:
LDI R16, 0x01
IN R20, PINC ; save button
CP R16, R20 ; check if save button is pushed
BREQ EXT_MEMORY ; if yes, go to EXT_MEMORY for further
checks and to save
RJMP INT_MEMORY ; if not, go to INT_MEMORY for further
checks and to save

EXT_MEMORY: ; X register will be used for the external memory
LDI R16, HIGH(EXTN_START)
CP R17, R16 ; compare the high of start
address with the high of entered memory address (in port E - R17)
BRLT INVALID_ADDR ; invalid address for external memory
CP R17, R22 ; comparing the high of end
address with the high of entered memory address (R17)
BRGE INVALID_ADDR

STS addr, R19 ; storing the value in R19 into addr as
the low bytes
STS addr+1, R17 ; storing the value in R17 into
addr+1 as the high bytes
LDS XL, addr ; loading the value in addr into XL which
is located at R26

```

```

                                LDS XH, addr+1                ; loading the value in addr into
XH which is located at R27
                                ST X, R18                   ; storing the value into the X
                                RJMP start

                                INT_MEMORY:                 ; Y register will be used for the internal memory
                                LDI R16, HIGH(INTN_END)
                                CP R17, R21                 ; compare the low of start address
with the low of entered memory address (in port E - R17)
                                BRLT INVALID_ADDR            ; invalid address for internal memory
                                CP R16, R17                 ; comparing the high of end
address with the high of entered memory address (R17)
                                BRLT INVALID_ADDR

                                STS addr, R19                ; storing the value in R19 into addr as
the low bytes
                                STS addr+1, R17              ; storing the value in R17 into
addr+1 as the high bytes
                                LDS YL, addr                 ; loading the value in addr into YL which
is located at R28
                                LDS YH, addr+1               ; loading the value in addr into
YH which is located at R29
                                ST Y, R18                   ; storing the value to the Y

                                RJMP start

```

Debugging Outputs:

(Last case – address out of range)

Case where the values are;

A=101(0xA0), B=10111110(0xBE), C=0 (internal memory save), E=0x1F, F=0x00

R16->B, R17->E, R18->transmitted output (equals to B if no error), R19->F, R20->CRC polynomial (G), R21->save button, R22->high bytes of ending address of the stack

is tested:

Address = 0x1F00 (no data as the address is out of range)

The screenshot displays the Code3 (Debugging) - Microchip Studio interface. The main window shows the disassembly of the `NO_ERROR` routine. The routine starts with `RJMP NO_ERROR` and then enters a loop that checks if the address 0x1F00 is within the valid range (0x1F00 to 0x1F00). Since the address is out of range, the program branches to the `INVALID_ADDR` routine, which jumps back to the start of the program. The registers R00 through R27 are shown on the left, and the memory dump on the right shows data from 0x1E6A to 0x1F8C.

Registers:

Name	Value
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00
R07	0x00
R08	0x00
R09	0x00
R10	0x00
R11	0x00
R12	0x00
R13	0x00
R14	0x00
R15	0x00
R16	0xBE
R17	0x1F
R18	0xBE
R19	0x00
R20	0x35
R21	0x01
R22	0x18
R23	0x00
R24	0x00
R25	0x00
R26	0x00
R27	0x00

Disassembly:

```
RJMP NO_ERROR

NO_ERROR:

; Activating SRAM
; LDI R18, (1<<XMM1)|(1<<XMM0) ; releasing pins(4-7) for more efficient use of PORTC.
; STS_XMCRB, R18
; LDI R18, 0x80
; OUT MCUCR, R18 ; setting the SRE bit to activate the SRAM

IN R17, PINE ; high byte for the memory - 8 bits
IN R19, PINF ; low byte for the memory - 8 bits

LDI R21, HIGH(INTN_START) ; R21 = 0x01
LDI R22, HIGH(EXTN_END) ; R22 = 0x18
CP R17, R22
BRLT CONTROL_ADDR ; if valid for the high limit, branch to CONTROL_ADDR to check the
RJMP INVALID_ADDR ; if not, go to INVALID_ADDR

INVALID_ADDR:
RJMP start ; go back to the beginning of the program

CONTROL_ADDR:
; check the low limit and branch to VALID_ADDR for further operation
CP R17, R21
BRGE VALID_ADDR

VALID_ADDR:
```

Memory:

Address	Value
0x1E6A	?? ?? ?? ?? ?? ?? ?? ??
0x1E74	?? ?? ?? ?? ?? ?? ?? ??
0x1E7E	?? ?? ?? ?? ?? ?? ?? ??
0x1E88	?? ?? ?? ?? ?? ?? ?? ??
0x1E92	?? ?? ?? ?? ?? ?? ?? ??
0x1E9C	?? ?? ?? ?? ?? ?? ?? ??
0x1EA6	?? ?? ?? ?? ?? ?? ?? ??
0x1EB0	?? ?? ?? ?? ?? ?? ?? ??
0x1EBA	?? ?? ?? ?? ?? ?? ?? ??
0x1EC4	?? ?? ?? ?? ?? ?? ?? ??
0x1ECE	?? ?? ?? ?? ?? ?? ?? ??
0x1ED8	?? ?? ?? ?? ?? ?? ?? ??
0x1EE2	?? ?? ?? ?? ?? ?? ?? ??
0x1EEC	?? ?? ?? ?? ?? ?? ?? ??
0x1EF6	?? ?? ?? ?? ?? ?? ?? ??
0x1F00	?? ?? ?? ?? ?? ?? ?? ??
0x1F0A	?? ?? ?? ?? ?? ?? ?? ??
0x1F14	?? ?? ?? ?? ?? ?? ?? ??
0x1F1E	?? ?? ?? ?? ?? ?? ?? ??
0x1F28	?? ?? ?? ?? ?? ?? ?? ??
0x1F32	?? ?? ?? ?? ?? ?? ?? ??
0x1F3C	?? ?? ?? ?? ?? ?? ?? ??
0x1F46	?? ?? ?? ?? ?? ?? ?? ??
0x1F50	?? ?? ?? ?? ?? ?? ?? ??
0x1F5A	?? ?? ?? ?? ?? ?? ?? ??
0x1F64	?? ?? ?? ?? ?? ?? ?? ??
0x1F6E	?? ?? ?? ?? ?? ?? ?? ??
0x1F78	?? ?? ?? ?? ?? ?? ?? ??
0x1F82	?? ?? ?? ?? ?? ?? ?? ??
0x1F8C	?? ?? ?? ?? ?? ?? ?? ??

Registers:

R00 = 0x00 R01 = 0x00 R02 = 0x00 R03 = 0x00 R04 = 0x00 R05 = 0x00 R06 = 0x00 R07 = 0x00 R08 = 0x00 R09 = 0x00 R10 = 0x00 R11 = 0x00 R12 = 0x00 R13 = 0x00 R14 = 0x00
R15 = 0x00 R16 = 0xBE R17 = 0x1F R18 = 0xBE R19 = 0x00 R20 = 0x35 R21 = 0x01 R22 = 0x18 R23 = 0x00 R24 = 0x00 R25 = 0x00 R26 = 0x00 R27 = 0x00 R28 = 0x00 R29 = 0x00
R30 = 0x00 R31 = 0xB1

No data is saved in any of the memory, either internal or external, as the address is out of range.

Case where the values are;

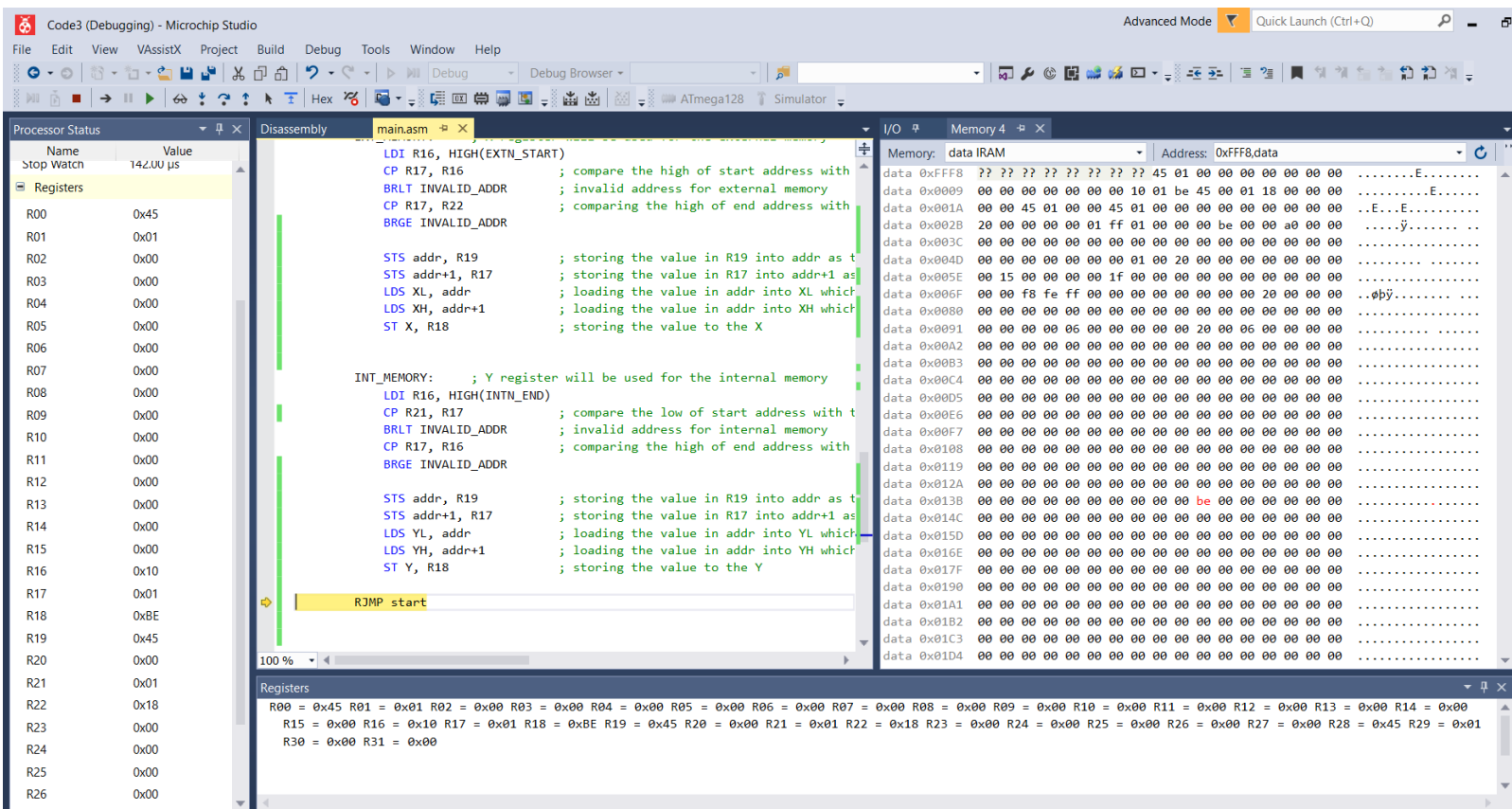
A=101(0xA0), B=10111110(0xBE), C=0 (internal memory save), E=0x01, F=0x45

R16->B, R17->E, R18->transmitted output (equals to B if no error), R19->F, R20->CRC

polynomial (G), R21->save button, R22->high bytes of ending address of the stack

is tested:

Address = 0x0145 (holding the value 0xBE which is in R18 as expected)



As the address entered is valid and the save button is not pushed, the value of R18 will be saved in the internal memory at address entered which is 0x0145. It can be seen from the figure, that address is holding 0xbe as expected.

Case where the values are;

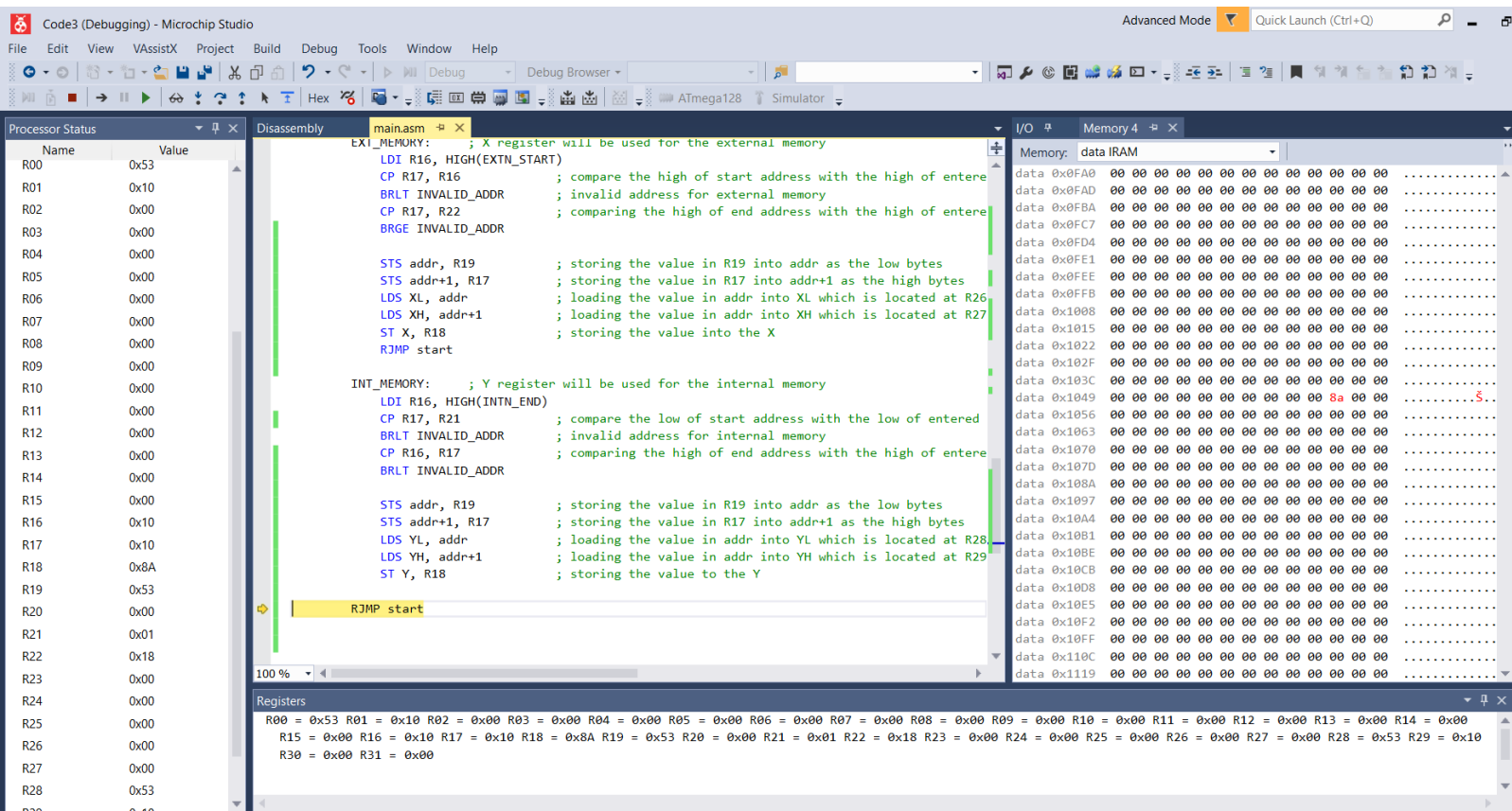
A=100(0x80), B=10001010(0x8A), C=0 (internal memory save), E=0x10, F=0xE0

R16->B, R17->E, R18->transmitted output (equals to B if no error), R19->F, R20->CRC

polynomial (G), R21->save button, R22->high bytes of ending address of the stack

is tested:

Address = 0x10E0 (holding the value 0x8A which is in R18 as expected)



As the address entered is valid and the save button is not pushed, the value of R18 will be saved in the internal memory at address entered which is 0x10E0. It can be seen from the figure, that address is holding 0x8a as expected.

Case where the values are;

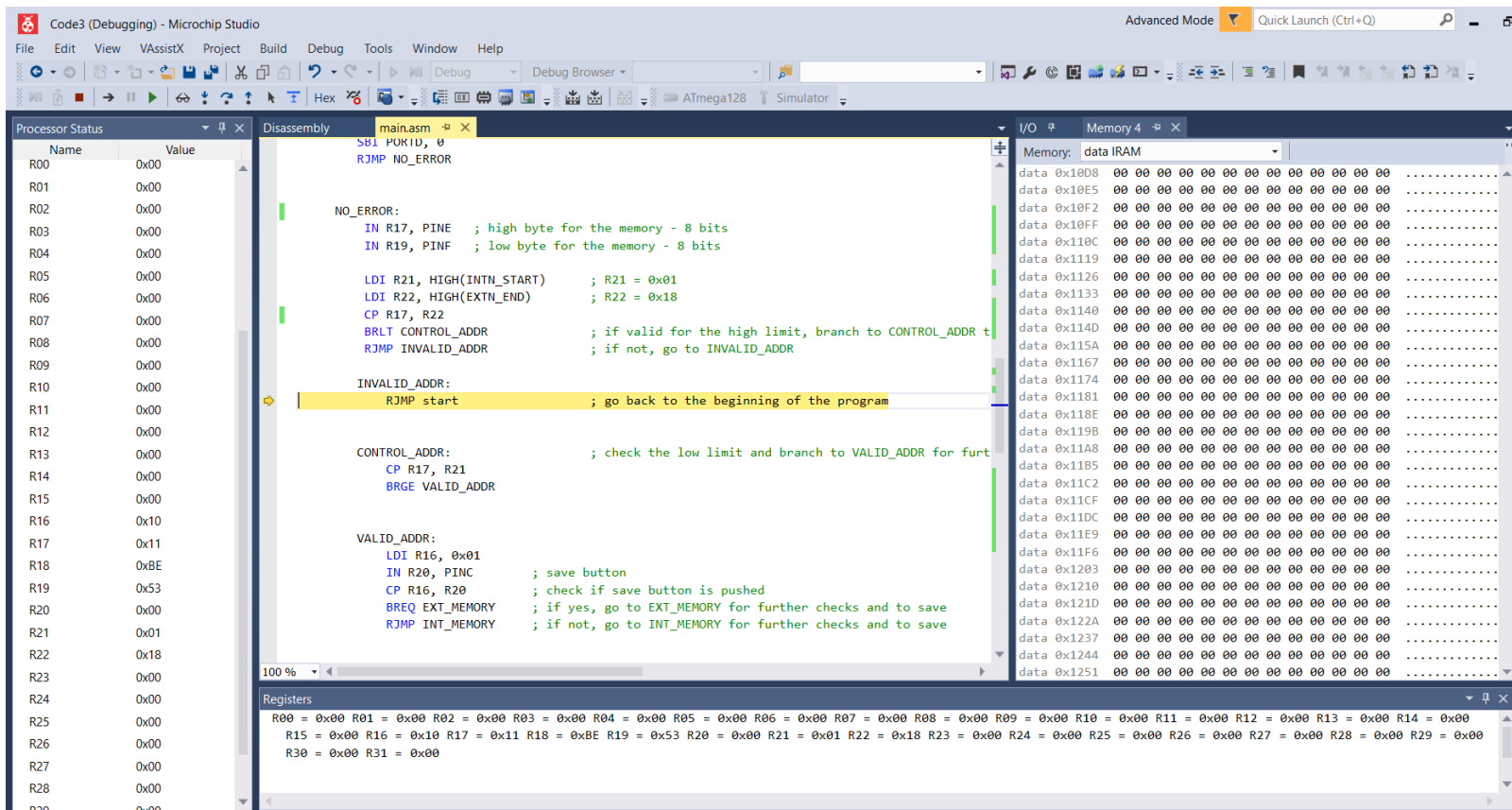
A=101(0xA0), B=10111110(0xBE), C=0 (internal memory save), E=0x11, F=0x53

R16->B, R17->E, R18->transmitted output (equals to B if no error), R19->F, R20->CRC

polynomial (G), R21->save button, R22->high bytes of ending address of the stack

is tested:

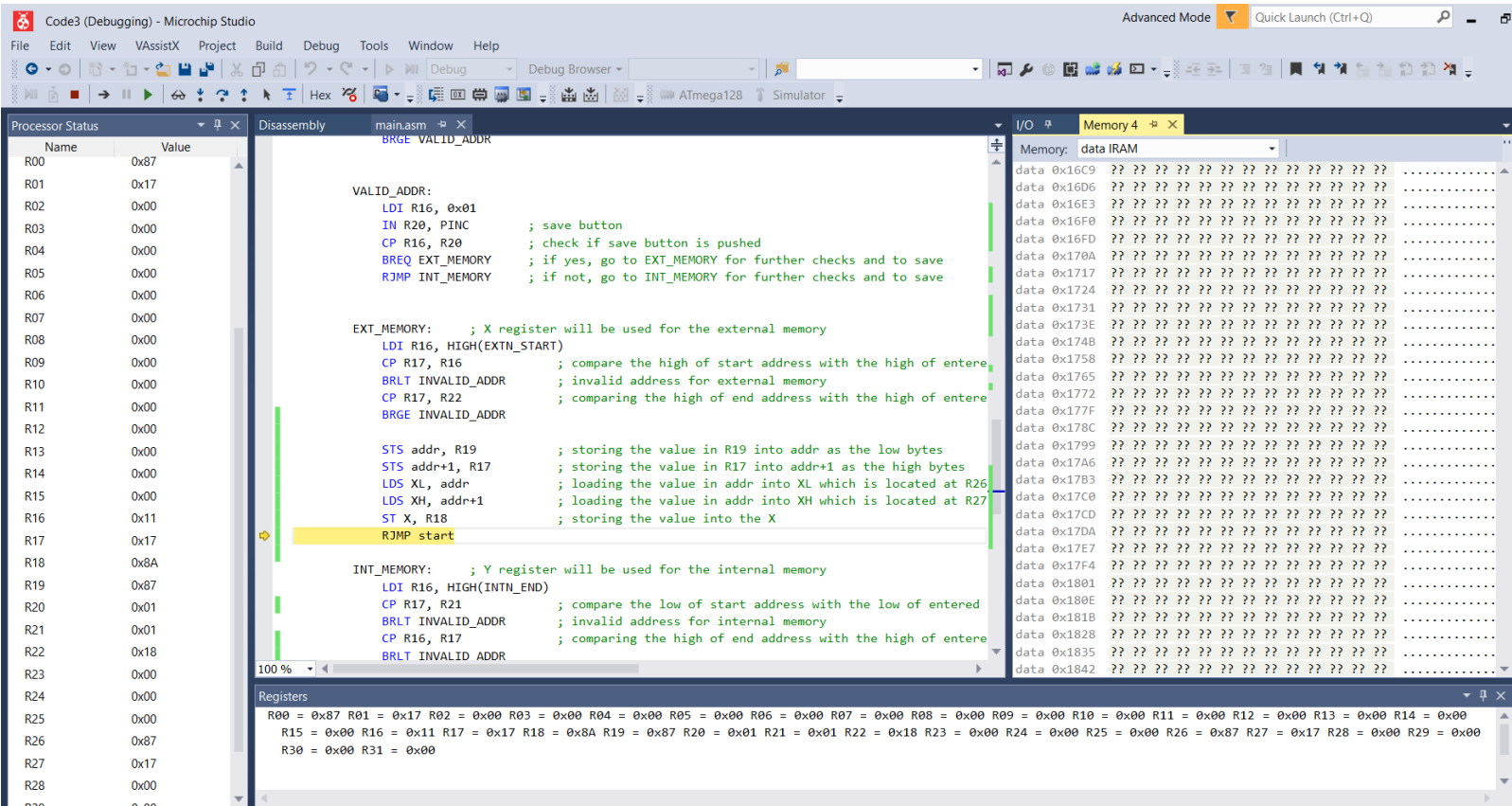
Address = 0x1153 (invalid address for the internal memory save)



The save button is not pushed but the address entered is the address for the external memory, so when the program will treat the address as it is supposed to be in the range for the internal memory but when the checks are done, it will take the address as an invalid address as it is not within the range for the internal memory. So the program will go back to start, and that's why no data will be saved at the address entered, which is 0x1153.

A=100(0x80), B=10001010(0x8A), C=1 (external memory save), E=0x17, F=0x87
R16->B, R17->E, R18->transmitted output (equals to B if no error), R19->F, R20->CRC
polynomial (G), R21->save button, R22->high bytes of ending address of the stack

Address = 0x1787 (needs to hold the value 0x8A which is in R18)



The save button is pushed and the address is valid considering the external memory range, so the value in R18 is supposed to be saved in the entered address, which is 0x1787. It assigns the correct values to the X register pairs (in R26 and R27). R26=0x87 and R27=0x17 which represents the high bytes and low bytes of the address, respectively. Both registers get the correct values but the value in R18 is not saved to that address for some reason that I couldn't solve.