U5 DSP56002 Assembler

Ziel

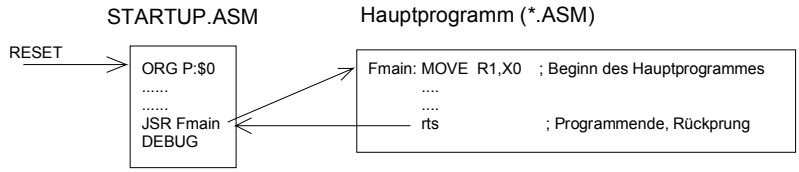
Erstellen und Austesten eines Kleinprogrammes in Assembler mit dem EVM DSP56002 Board unter Verwendung der Tasking Entwicklungsumgebung (EDE).

Umfeld

Der Einsatz von Assembler in einem Programmierprojekt kann durchaus sinnvoll sein um zeitkritische Programmteile zu optimieren, d.h. Flaschenhälse wegzuschaffen. Wo immer möglich sollte aber die Codierung in Hochsprache erfolgen.
Beim Test mit dem Debugger wird der Code bis auf Assemblerniveau aufgelöst. Schwierige Fehler können oft nur mit dem tieferen Verständnis und Analyse des Assemblercodes gefunden und behoben werden.

Startup-Code

Normalerweise muss dem eigentlichen Programm ein Startup-Code beigefügt werden der beim RESET des Prozessors gewisse Initialisierungen vornimmt, wie Stack setzen, Speicherbereiche initialisieren, etc. Dieser „Boot-Code“ wird durch das File `STARTUP.ASM` dargestellt. Es beginnt beim RESET-Vektor `P:$0` den Programmlauf, führt die Initialisierungen durch und ruft das eigentlich auszuführende Programm auf, indem das Symbol `Fmain` als Subroutine angesprungen wird.
Das File `STARTUP.ASM` ist daher dem Projekt im Quell- oder Objektcode zuzufügen.



Vereinfachtere Ablauf nach dem RESET: Das Startup-Modul ruft Fmain als Subroutine auf.

(Das File STARTUP.ASM ist bei den Beispielen unter \asm zu finden)

Aufgaben

1. Aufsetzen eines neuen, leeren Assembler-Projektes (z.B. Uebung5_Add2Sum_1, keine Leerschläge erlaubt. Vgl. hierzu auch Übung 1.
   Codieren des ersten Assembler-Beispieles und Austesten mit Crossview.

2. Codieren des zweiten Assembler Beispiels. Achtung: Beim Linken dürfen keine Fehlermeldungen erscheinen!

3. Codieren Sie das Programm zur Filterung in Kap. 6 des EVM Quick Start Guide so, dass es mit der Tasking-EDE erstellt und getestet werden kann. (Zu finden in Motorola HTML-Hilfe, EVM user’s guide)

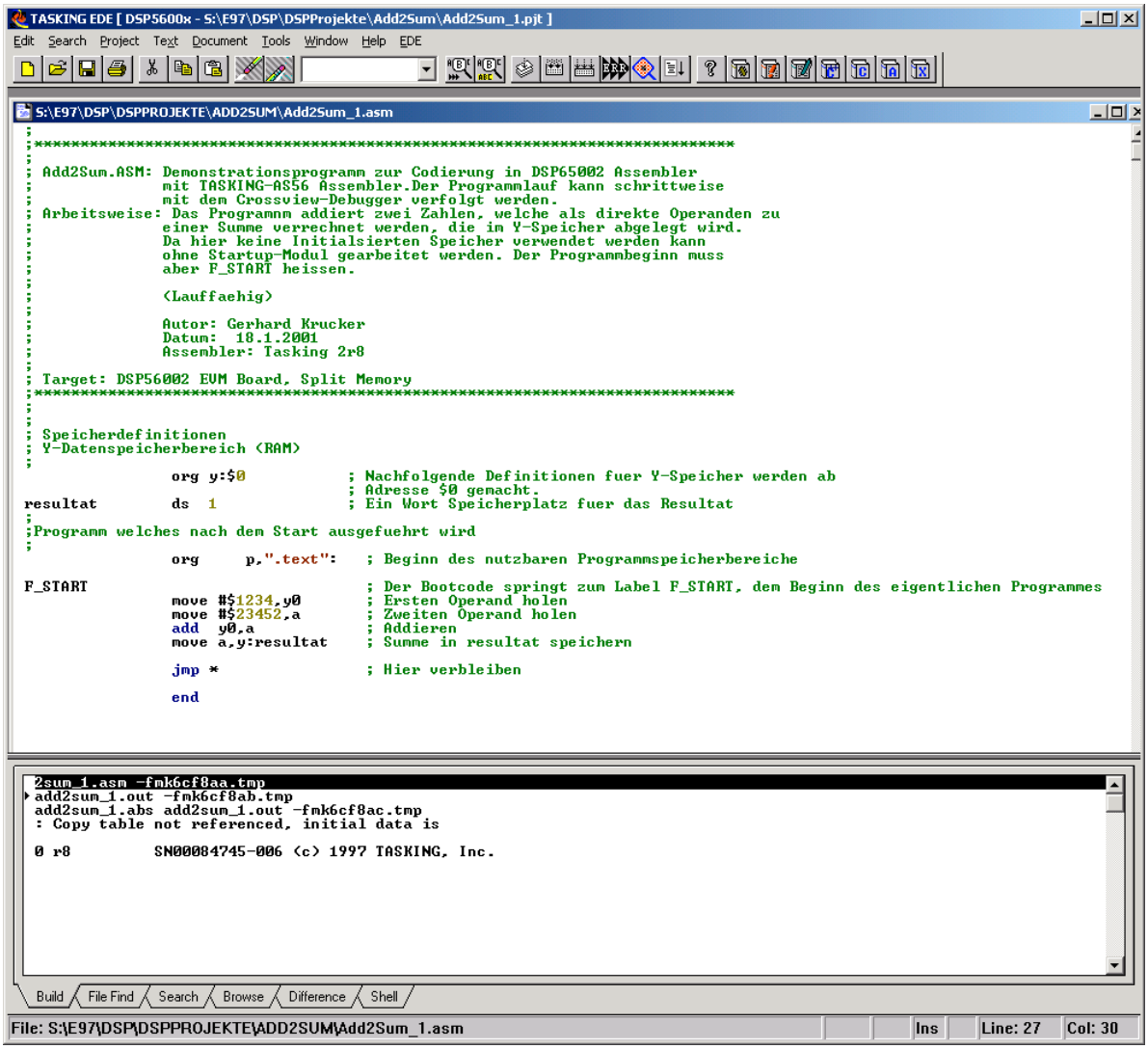
Beispiele

Assemblerprogramme können direkt in der Tasking EDE erstellt werden. Die Assmbler Quelltexte erhalten die Extension `.asm` worauf sie automatisch vom as56 Assembler erkannt und bearbeitet werden. Der Tasking Assembler ist weitgehend kompatibel zum Motorola Assembler, hat aber zahlreiche Erweiterungen. Die Online-Hilfe ist ausgezeichnet, wenn gewisse Grundkenntnisse bestehen.

Für einfache reine Assemblerprogramme sind nur in den Linkeroptionen die C-Libraries vom Linken auszuschalten (vgl. Screenshoot in Beispiel 2)

Beispiel 1

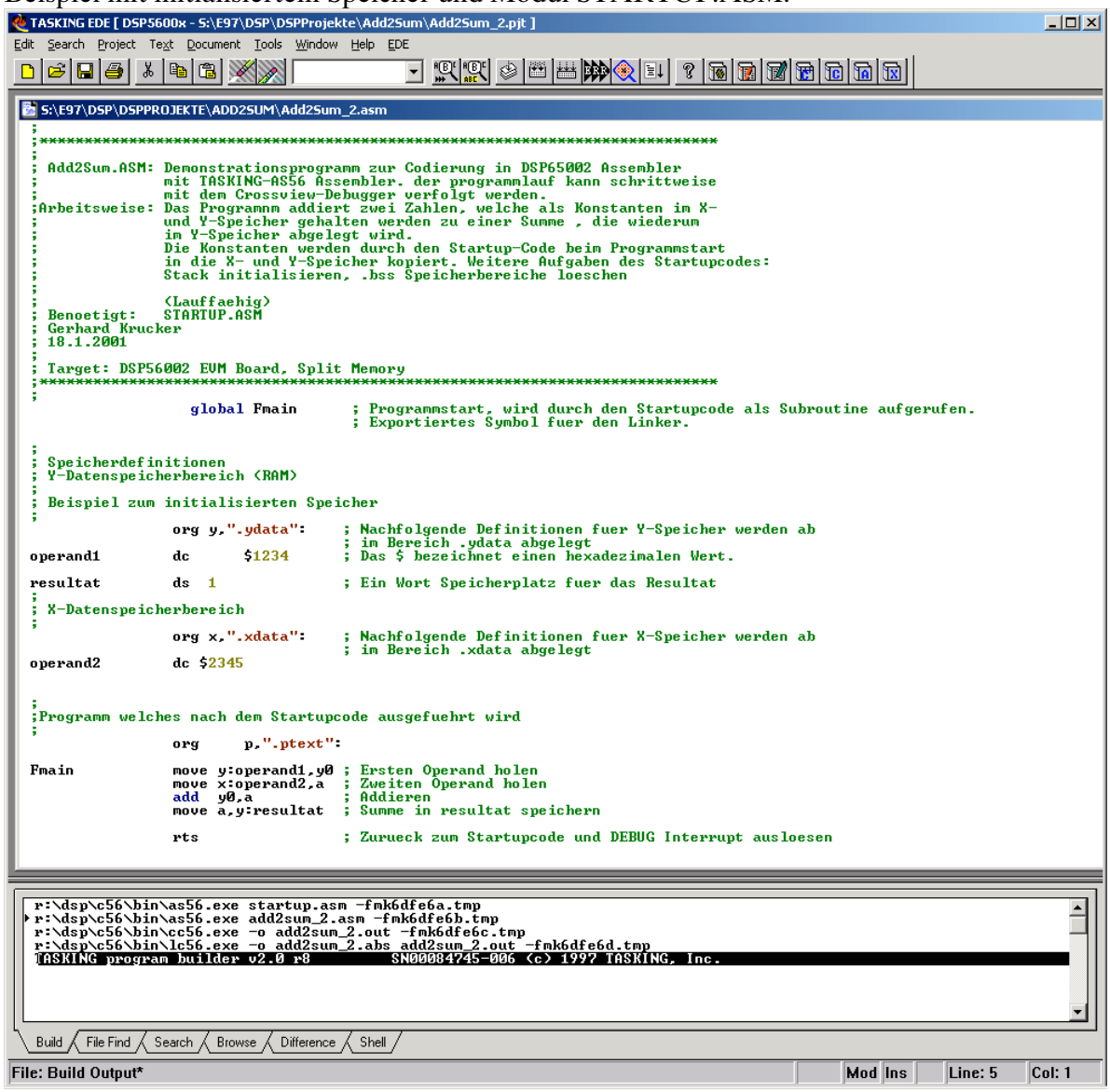
Ein Einfachstbeispiel zum Addieren zweier Zahlen welche im Initialisierten X-/Y-Speicher liegen: Da hier weder Stack, noch initialisierte Speicher benötigt werden, kann auf das Startup-Modul verzichtet werden:



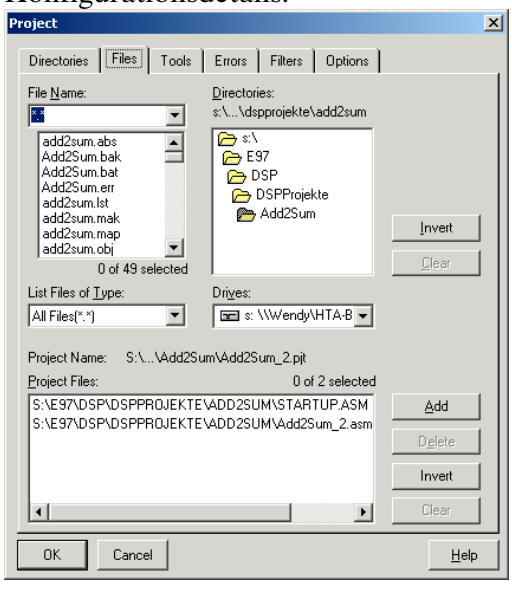
Hinweis:
Dieses Einfachstbeispiel soll nur Illustrieren, mit wie wenig Aufwand ein Kleinprogramm erstellt werden kann und ist als „Murks“ anzusehen. Man beachte, dass F_START die Programmstelle P:$0 verkörpert und der Code hier im Interruptvektorbereich liegt. Daher ist für alle Programme die etwas aufwändigere Struktur des folgenden Beispiels in jedem Fall vorzuziehen.

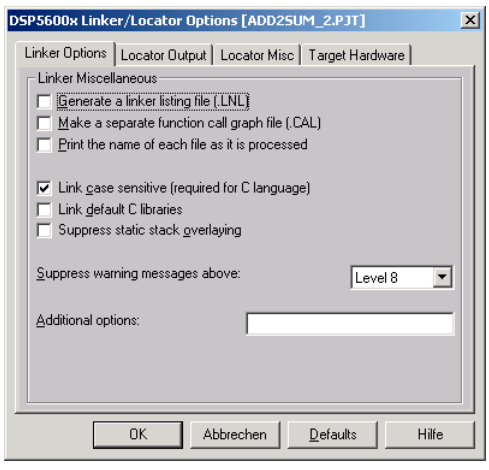
Beispiel 2

Beispiel mit initialisiertem Speicher und Modul STARTUP.ASM:

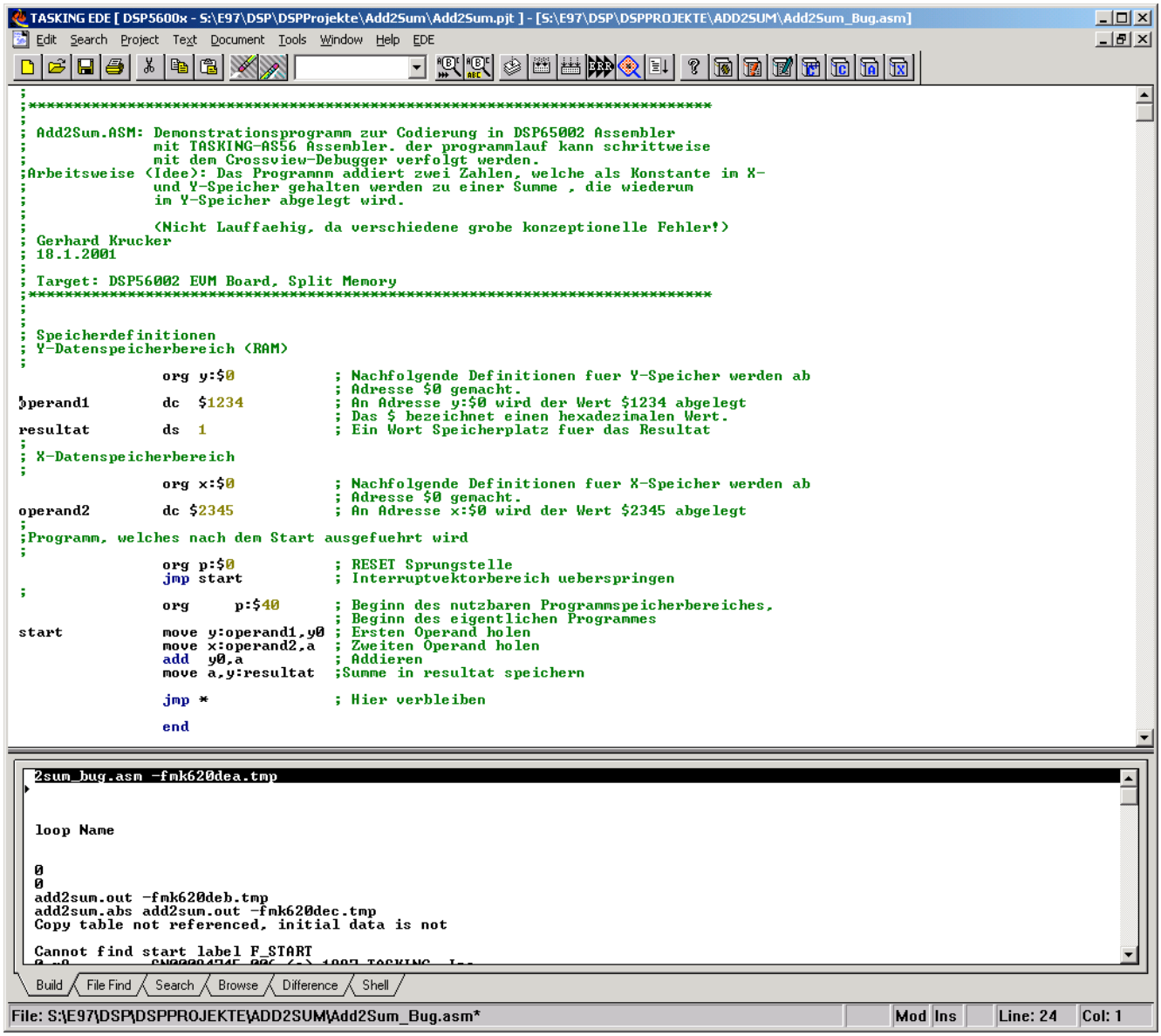


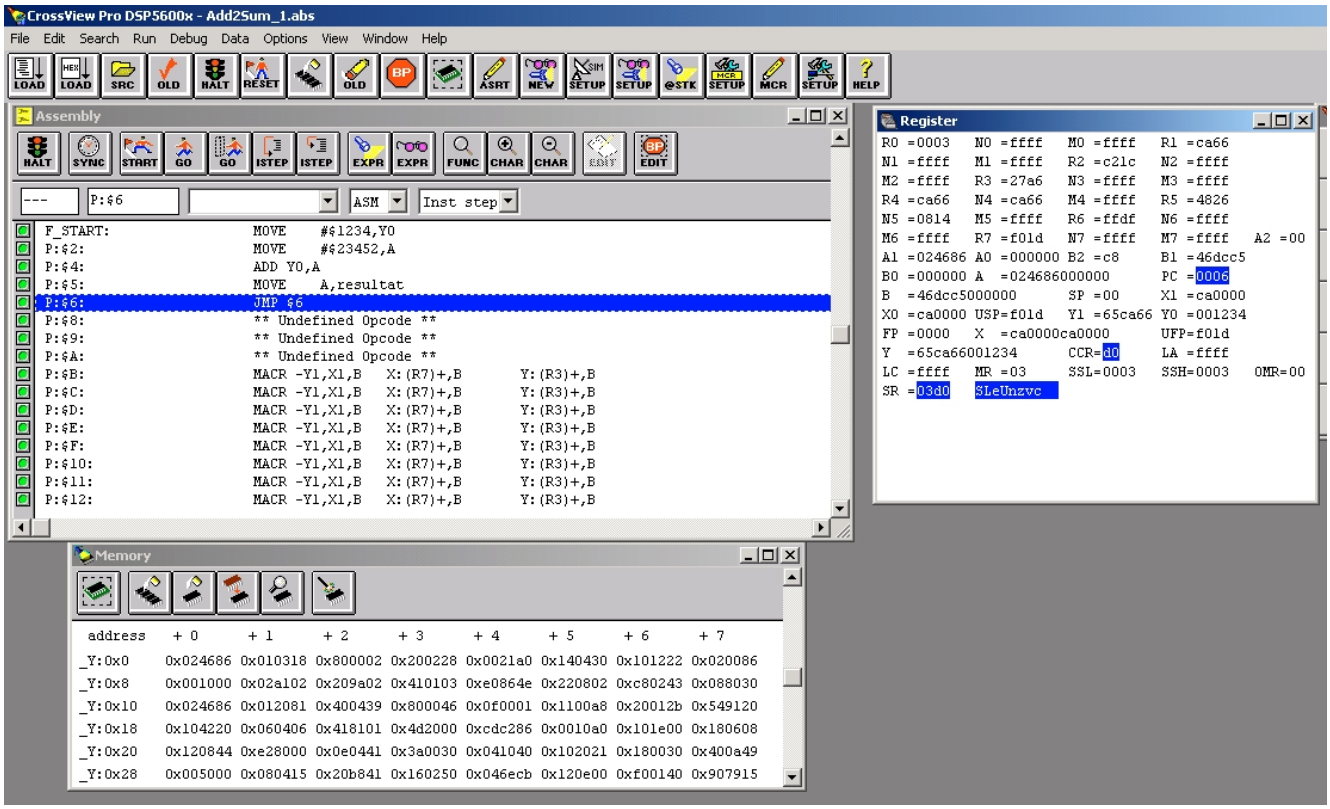
Konfigurationsdetails:





Idee eines Assemblerprogrammes, das zwei Zahlen addiert. Syntaktisch korrekt aber mit diversen konzeptionellen Fehlern:




CrossView Pro DSP5600x - Add2Sum_1.abs
File Edit Search Run Debug Data Options View Window Help
Assembly
F_START: MOVE #$1234,Y0
P:$2: MOVE #$23452,A
P:$4: ADD Y0,A
P:$5: MOVE A,resultat
P:$6: JMP $6
P:$8: ** Undefined Opcode **
P:$9: ** Undefined Opcode **
P:$A: ** Undefined Opcode **
P:$B: MACR -Y1,X1,B X:(R7)+,B Y:(R3)+,B
P:$C: MACR -Y1,X1,B X:(R7)+,B Y:(R3)+,B
P:$D: MACR -Y1,X1,B X:(R7)+,B Y:(R3)+,B
P:$E: MACR -Y1,X1,B X:(R7)+,B Y:(R3)+,B
P:$F: MACR -Y1,X1,B X:(R7)+,B Y:(R3)+,B
P:$10: MACR -Y1,X1,B X:(R7)+,B Y:(R3)+,B
P:$11: MACR -Y1,X1,B X:(R7)+,B Y:(R3)+,B
P:$12: MACR -Y1,X1,B X:(R7)+,B Y:(R3)+,B
Register
R0 =0003 N0 =ffff M0 =ffff R1 =ca66
N1 =ffff M1 =ffff R2 =c21c N2 =ffff
M2 =ffff R3 =27a6 N3 =ffff M3 =ffff
R4 =ca66 N4 =ca66 M4 =ffff R5 =4826
N5 =0814 M5 =ffff R6 =ffdf N6 =ffff
M6 =ffff R7 =f01d N7 =ffff M7 =ffff A2 =00
A1 =024686 A0 =000000 B2 =c8 B1 =46dcc5
B0 =000000 A =024686000000 PC =0006
B =46dcc5000000 SP =00 X1 =ca0000
X0 =ca0000 USP=f01d Y1 =65ca66 Y0 =001234
FP =0000 X =ca0000ca0000 UFP=f01d
Y =65ca66001234 CCR=d0 LA =ffff
LC =ffff MR =03 SSL=0003 SSH=0003 OMR=00
SR =03d0 SLeUnzvc
Memory
address + 0 + 1 + 2 + 3 + 4 + 5 + 6 + 7
_Y:0x0 0x024686 0x010318 0x800002 0x200228 0x0021a0 0x140430 0x101222 0x020086
_Y:0x8 0x001000 0x02a102 0x209a02 0x410103 0xe0864e 0x220802 0xc80243 0x088030
_Y:0x10 0x024686 0x012081 0x400439 0x800046 0x0f0001 0x1100a8 0x20012b 0x549120
_Y:0x18 0x104220 0x060406 0x418101 0x4d2000 0xcdc286 0x0010a0 0x101e00 0x180608
_Y:0x20 0x120844 0xe28000 0x0e0441 0x3a0030 0x041040 0x102021 0x180030 0x400a49
_Y:0x28 0x005000 0x080415 0x20b841 0x160250 0x046ecb 0x120e00 0xf00140 0x907915