ICE Emulator for 8051

Version 18-Nov-2020

```
P:000072  \\KEILS\KEILS\sieve+6F                          ..........  MIX  AI

E::w.d.l
 addr/line  code    label      mnemonic            comment
  P:00006E  351D               addc    a,1D        ; a,primz
  P:000070  F51F               mov     1F,a        ; k,a
       32                            while ( k <= SIZE )
  P:000072  C3                 clr     c
  P:000073  E520               mov     a,20
  P:000075  9413               subb    a,#13
  P:000077  E51F               mov     a,1F        ; a,k
  P:000079  9400               subb    a,#0
  P:00007B  30D202             jnb     ov,80
  P:00007E  B2E7               cpl     acc.7
```

```
E::w.v.f /l /c
          while ( TRUE )
          {
                  sieve();
-000  sieve()
      i = 0
      primz = 3
      k = 3
      anzahl = 0
```

```
E::w.r
Cy   _   R0   8   A        0    SP >00
AC   _   R1   0   B        0    -01 B2
F0   0   R2   0   IE       0    -02 00
RS   0   R3   0   DPTR     0    -03 00
Ov   _   R4   0   PSW      0    -04 03
F1   0   R5   0   PC      72    -05 00
P    _   R6   0   SP      24    -06 03
Tsk      R7   0   XSP      0    -07 00
```

For general informations about the In-Circuit Debugger refer to the **"ICE User's Guide"** (ice_user.pdf). All general commands are described in **"PowerView Command Reference"** (ide_ref.pdf) and **"General Commands and Functions"**.

WARNING

NOTE:	Do not connect or remove probe from target while target power is ON. Power up: Switch on emulator first, then target Power down: Switch off target first, then emulator

Before debugging can be started, the emulator must be configured by software:

Ready to run setup files for most standard compilers can be found on the software CD in the directory **…/Demo/I51/Compiler**. All setup files are designed to run the emulator stand alone without target hardware.

The following description should make the initial setup (to run the emulator together with the target hardware) easier. It describes a typical setup with frequently used settings. It is recommended to use the programming language PRACTICE to create a batch file, which includes all necessary setup commands. PRACTICE files (*.cmm) can be created with the PRACTICE editor pedit (Command: **PEDIT <file name>**) or with any other text editor.

A basic setup file includes the following parts:

1. Set cpu-type and -mode
2. Set system options
3. Select dualport mode (optional)
4. Set mapper (optional)
5. Select frequency (optional)
6. Activate the emulator
7. Load application file (optional)
8. Set breakpoints (optional)
9. Start application
10. Stop application (optional)

Here a typical example, how to setup the system:

1. Set **cpu-type**

   The command **sys.cpu** is used to select one derivative within a cpu-family and to set its operation mode.

```
system.down                 ; switch the system down
system.cpu I8051            ; select derivative Intel 8051
```

2. Set **system options**

   The system window controls the CPU specific setup. Please check this window very carefully and set the appropriate options. Use the ↖? button in the main tool bar and click to the option check box (Command: **HELP.PICK**) to get online help in a pop up window.

```
system.option IOSTOP on     ; switch IOSTOPE mode on
```

3. Select **dualport mode** (optional)

   Dualport allows access to emulation RAM, while emulation is running. This is necessary to display variables, set breakpoints or display the flag listings while the emulation is running. **System.Access** selects how dualport access is done.

```
system.access denied        ; denied: dualport is disabled
```

4. Set mapper (optional)

   The mapper controls the memory access of the CPU. This means the use of internal or external memory, the protection of a memory bank etc. Address ranges must be defined by using **memory classes**.

```
map.reset                   ; reset mapper (all external)
map.ram P:0x0--0x07fff      ; emulation RAM: 32KB (e.g. for
                            ; program)
map.ram X:0--0x0FFFF        ; emulation RAM: 64 KB (e.g. for data)
map.intern P:0x0--0x07fff   ; map program memory internal
map.extern X:               ; map data memory external
```

5. Select frequency (optional)

   The CPU can be clocked by an internal (emulator) or external (target) clock source. If the internal clock is used, the clock is provides by the VCO of the emulator. The setting of the internal clock is done by the “vco” command.

   The current CPU frequency can be displayed in the counter window.

```
vco.clock 20.    ; input clock to the EXTAL pin of the cpu is set
                 ; to 20 MHz (only necessary if internal clock is
                 ; used)
```

6. **Activate** the emulator

When the emulator is activated a debug-monitor program is loaded into a hidden emulator memory. Afterwards, a bondout reset-signal is inactivated and the monitor program starts. This program allows access to user memory (data.dump, data.list) and cpu-registers, and gives control to start and stop the emulation.

```
system.mode emulint          ; system works with internal target
                             ; clock
```

7. **Load** application file (optional)

Application can be loaded by various file formats. UBROF format is often used to load code and symbol information. For information about the load command for your compiler see **Compiler.**

```
data.load.ih test.hex        ; load application file
```

8. Set breakpoints (optional)

There are several ways to set breakpoints (Command: **Break.Set**). Breakpoints can be displayed using the **Break.List** command.

```
breakpoint.set main /program     ; set program break on function
                                 ; main
breakpoint.set flags /write      ; set write break on variable
                                 ; 'flags'
```

9. **Start** application

Application can be started with giving a break address. For example ***"go main"*** starts the application and stops at symbol main.

```
go                           ; run application
```

10. **Stop** application (optional)

Application can be breaked manually by using th **BREAK** command.

```
break                        ;break application manually
```

Troubleshooting

No Information available.

Target Power Supply Switch
 	
Ref: 0103
	Is there a simple way to control target power supply via the ICE to prevent problems after the ICE has been powered off?
	
	Follow the sequence below.
	
	If you own an output probe COUT8, connect it to the STROBE output connector.
	
	Type PULSE2. and press F1. You will get the pin out of the output probe COUT8. Pin 13 (OUT6) delivers +5 V after the emulator has finished its initialization and 0 V if the emulator is powered off. This can be used to drive a relay via a transistor to switch the target power on and off automatically if the Pulse Generator is not used for other purposes. The schematic of the switching unit can be found in the file TARGETC.CMM.
	
	Additionally Pin 13 (OUT6) can be controlled by ICE commands.
	
	`Target power supply off. "PULSE2.P +"`
	`Target power supply on.  "PULSE2.P -"`
	
	The following PRACTICE command file creates 3 buttons in the Toolbox for:
	
	`Target power on`
	`Target power off`
	`Target power off and QUIT.`

Wrong Location after Break Ref: 0030	**Why is the location after break wrong?** Most emulators use some bytes of user stack for the break system. Therefore it is necessary to have valid stack, if single step or breakpoints are used.
Bank Number for Bank File (*.bnk) Ref: 0114	**Which number contains R6 if the bank file is called?** The parameter value in R6 of the bank file contains the number of the requested bank. However, it depends on the used bank logic if R6 contains value 1 for the bank 1. A better description is, that R6 contains the same value as the value of the bank probe input lines for the appropriate bank number. If there is a address translation by the MMU command, R6 could contain 3 for bank 1 depending on the address translation.
8051 Banking using 8051 Ports Ref: 0049	**I have some problems using 8051 ports as a bank register. Do you know reasons for that behavior?** If port pins are used as additional address lines for banking purposes, the address lines must be synchronized to the regular addresses. In other case, nobody can predict when the port pins are valid. Refer to the manufactures 8051 manual.

8051 CPU Internal Memory Externally Ref: 0010	**Can I map the CPU internal memory externally?** No, not recommended. The CPU internal programm area must be mapped internally because this memory is an on-chip memory. If the 8051 is in microcontroller mode (EA=1), the program area can never be substituted with a memory on the target. The CPU internal data area can not be mapped externally as well, because there is no access to the internal address and data bus in any case. But what is the difference between memory which is mapped internally or externally? Only off-chip memory (program or/and data area) can be mapped internally (within the emulator provided emulation memory) or mapped externally (to the user provided memory on the target, external the emulator).
8051 Differences Bond-out vs. non Bond-out Ref: 0008	**What is the difference between a bond-out and a non bond-out emulator?** A bond-out chip provides a lot of additional signals and features which simplify the control of a CPU, like the user program stop, entry to the user program and exit from the user program. Basically however, the bond-out chip provides the addresses, data and the control lines of a CPU internal program area (EPROM, PROM, EE_PROM, FLASH_ROM). As an option, all internal peripherals and interrupt sources can be stopped while the user program has been stopped. Additional registers contain information about pending interrupts etc. Some bond-out chips are "Combi-CPUs" which can emulate more than a derivative of the 8051 family. A non bond-out emulator uses the original chip, which is readily available their local distributor. There are no additional lines and information available about the internal memory area and there is no direct way to stop internal peripherials or to prevent internal interrupt requests during an user program stop. Special workarounds (provided by the emulator) cater for nearly the same comfort as a bond-out solution. Please bear mind that the program area must be external (EA=0). Conclusion: If you use a 8051 derivative in microcontroller mode (EA=1) and have not got program memory on the target, then you must choose the bond-out solution. This solution supports both methods of operation EA=0 and EA=1. In the other case, if you use only the microprocessor mode (EA=0) with EPROM on the target, you may choose the non bond-out version.
8051 Reset while Real Time Program Execution Ref: 0064	**What can cause error messages while real time program execution, if the RESET line is activated or released?** There is a difference in behavior of the original CPU and the emulator. The emulator does not have a Schmitt-Trigger input like the CPU has. In case of problems, it is recommended to check the RESET line: Are there spikes, heavy noise or is the falling or rising slope of RESET slower than 10 us.

8051 Stop Internal Watch-dog Timer Ref: 0011	**How can I stop the internal watch-dog timer after break?** There are two different ways to stop or to service the internal watch-dog timer for the case, that the watch-dog cannot be disabled by software. It depends on the emulation technique which is used. If a bond-out chip is used, the customer may choose the IOSTOP option in the SYSTEM control window. After break, all internal peripherals including the watch-dog timer are stopped or inhibited if the option is on. In a non bond-out system, the watch-dog timer must be serviced after break to prevent a reset. The TRACE32 is able to support any software routines in the background while the emulator has stopped the user programm execution. To achieve that behavior, follow the instruction you will get if you type HELP TASK or on the appropriate pages in the user guide. This procedure can also be used to keep the emulator active for any interrupt requests after an user programm break.

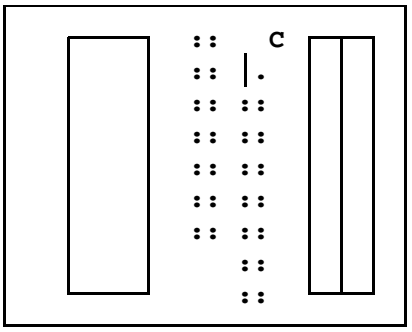
8051
 	
Trace Internal Registers
 	
Ref: 0009
	How do I trace a chip internal data transfer from one register to an other?
	
	Neither a bond-out based nor a non bond-out emulator has access to the internal busses between the registers. Also it is impossible to see any access to or from an internal auxiliary memory area, except the CPU provides special modes. During real time program execution there is no chance to trace these accesses or make decisions depending on the content. During program emulation (not a real time program execution) there are a lot of emulator instructions to verify register or internal memory. As a combination of both, so-called spot breakpoints are available.
	Nevertheless the emulator and the analyzer are able to trigger and trace on the access type (e.g. read bit direct) and on the internal addresses of byte direct and bit direct accesses.
	There are two different ways for a work-around. But bear mind, it is impossible to trace the value of the EA bit in real time.
	1. It is possible to stop the user program by setting a breakpoint to any internal bit or byte address. This causes an asynchronous break of the user program. But consider, the access must be made by an direct access (indirect accesses will not work).
	2. It is possible to trigger (also asynchronously) to an internal access using the analyzer. But consider that the internal data transfer can never be seen outside the chip. As long as the access type and the address is known the following trigger program may help (in this example SETB EA and CLR EA):
	`DATA.B0 ea_adr 0AF ;bit address of EA`
	`DATA.B0 clr_opf 0C2 ;clr EA opfetch`
	`DATA.B0 set_opf 0D2 ;set EA opfetch`
	`LL0: goto LL2 if clr_opf:a:opfetch`
	`cont if set_opf:a:opfetch`
	`LL1: s,mark.a if ea_adr`
	`goto LL0`
	`LL2: s,mark.b if ea_adr`
	`goto LL0`
	In the analyzer list window, the number and the order of access to the EA bit can be seen.
	
	These are examples, which should only show the way to work-around.

Configuration

The configuration between the derivatives of the 8051 family is done by changing the probe or connector modules. The software is configured automatically.

8051

To emulate the 8051 ROM version (no external memory) without bondout chip, a piggy-back version of the 8051 chip is used on the 8051 adapter. The OKI 85C154VS and the MHS 80C31P8/P16 are such piggy-back versions of 8051. They require an additional small adapter cable between the EPROM socket and the 26-pin connector on the module.



	Con A	Con B	Jumper C
no Piggy-Back	not used	not used	closed
OKI-85C154VS	connected	open	closed
MHS-80C51P32	open	connected	open

80C152JA DIL

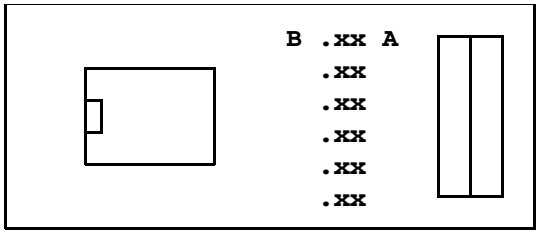
Mount adapter 80152JA-DIL and connect bridge array in position A for 83C152JA emulation or in position B for romless version and for DMA. The correct CPU type is set automatically in the system window.

80C152JA-PLCC

Mount adapter 80152JB-PLCC and select CPU-type 80C152JA in system window. Set bridge array in position A for 83C152JA emulation or position B for romless version and for DMA.

80C152JB-PLCC

Mount adapter 80152JB-PLCC and select CPU-type 80C152JB in system window. Depending on the used bus mode set the bridge array as shown in position A or B.




```
Pos A:  80C152JA,80C152JB, Opfetch via P5/P6, P5/P6 of target open
Pos B:  80C152JB, Opfetch via P0/P2, P5/P6 connected to target
```

C515C

For proper operation all switches of DIPSWITCH S101 must be closed and all switch of DIPSWITCH S100 must be open.

The C505C is a subset of the C515C with some differences.

The A/D input lines are normally connected at Port1. Due to the C515C as emulation CPU, the A/D input lines are connected at Port6. For redirection of the A/D lines to Port6 from target Port1, the DIPSWITCH S101 and S100 must be set correctly.

Only for A/D operation, the appropriate pin of S101 must be closed and the equivalent pin of S100 must be open.

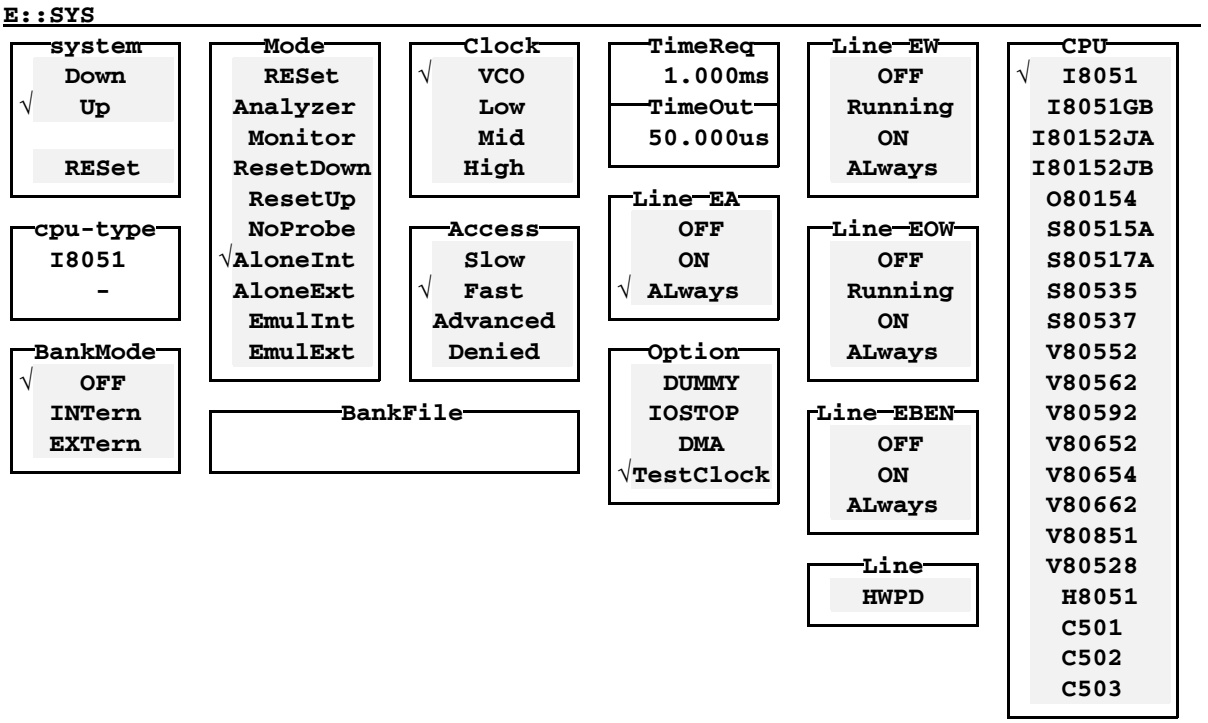
For digital functions the appropriate pin of S101 must be open and the equivalent pin of S100 must be closed.

Never close or open equivalent pins of S101 and S100 simultaneously.

S100:	pin 1:	C505C Port10
	pin 2:	C505C Port11
	pin 3:	C505C Port12
	pin 4:	C505C Port13
	pin 5:	C505C Port14
	pin 6:	C505C Port15
	pin 7:	C505C Port16
	pin 8:	C505C Port17
S101:	pin 1:	C505C A/D0 (Port60)
	pin 2:	C505C A/D1 (Port61)
	pin 3:	C505C A/D2 (Port62)
	pin 4:	C505C A/D3 (Port63)
	pin 5:	C505C A/D4 (Port64)
	pin 6:	C505C A/D5 (Port65)
	pin 7:	C505C A/D6 (Port66)
	pin 8:	C505C A/D7 (Port67)

For emulation the A/D unit of the C515C must be supplied.

Emulation Modes



The emulation head can stay in 6 modes. The modes are selected by the **SYStem.Up** or the **SYStem.Mode** command.

Format:	**SYStem.Mode** <*mode*>
<*mode*>:	**ResetDown** **ResetUp** **AloneInt** **AloneExt** **EmulInt** **EmulExt**

Reset Down	Target is down, all drivers are in tristate mode.
Reset Up	Target has power, drivers are logically in inactive state, but not tristate.
Alone Internal	Probe is running with internal clock, driver inactive. This mode is used for 'standalone' operation.
Alone External	Probe is running with external clock, driver inactive.
Emulation Internal	Probe is running with internal clock, strobes to target are generated.
Emulation External	Probe is running with external clock, strobes to target are activated.

In active mode, the power of the target is sensed and by switching down the target the emulator changes to **RESET** mode. The probe is not supplied by the target. When running without target, the target voltage is simulated by an internal pull-up resistor.

SYStem.Clock — Clock generation

Format:	**SYStem.Clock** <*option*>
<*option*>:	**VCO** **High** **Mid** **Low**

VCO	Variable frequency 1…35 MHz.
Low, Mid, High	2.5, 5.0 or 10.0 MHz.

Format:	**SYStem.CPU** <*type*>
<*mode*>:	**I8031 … V80851**

Selects the emulated processor type. This function is only required to distinguish pin compatible processors in the same emulation module.

SYStem.Access Dualport access

Format:	**SYStem.Access [Slow | Fast | Advanced | Denied]**

The mode can be changed only if the system is “down”.

Slow	Dualport access while ALE is active, for slow clock.
Fast	Dualport access while DUMMY-Cycle, for medium clock.
Advanced	Forced Dualport access while DUMMY-Cycle, for high speed emulation.
Denied	No Dualport access, when user program is running.

If DUMMY Cycle is active and the access mode FAST or ADVANCED is selected, sometimes wrong data values can appear in the trace of DUMMY cycles.

Special I/O-Register Module M582

Internal Memory

Program accesses to internal memory cannot be traced by the analyzer. Data selective breakpoints are not possible. Address read or write breakpoints on direct accessed or bit accessed memory are possible.

MOVX addressed by Ri (all bondout versions)

MOVX accesses to the emulation memory addressed by Ri causes wrong results. The upper byte of the address is wrong. The analyzer can not record and qualify these accesses. It is recommended to map such areas external for a correct program execution. But keep in mind that the analyzer doesn't work correctly.

Stack Usage

The probe for 80152 needs a valid stack at breakpoints. It uses 2 bytes of stack. All other derivatives need no stack.

Target Program Memory

It's not possible to load or modify the **target program memory area**, except the program area and the data area are not separated. The internal program memory area of a microcontroller (EA="1") should always mapped internal, because it's not possible to load a program into this area.

Power Down Mode

On boards till rev. 5 there is no support for power down mode because the CPU oscillator stops immediately and therefore several errors can appear. Newer boards support power down modes while the emulation is running. The dualport access mode must be set to **Denied** in this case.

Slow Down Mode

Slow down mode is only supported in **Slow** dualport access mode.

Idle Mode

The Emulator supports idle mode while a user program is running, but only without dualport access. Do not switch on the idle and power down bits in the peripheral window, or the system will go down immediately. Switch the dualport access mode to **Denied**. If idle mode was terminated by a reset, the analyzer records wrong INTACK cycles between last fetch before idle mode and restart from P:0000, but in reality there was no interrupt acknowledge.

PCON (only 80C517/537)

It is not possible to modify the bits PDE and PDS by an emulator command while the emulation is stopped. A modification is only possible in a user program using two special commands following immediately after each other.

Reset

The duration of target-reset and other reset signals from exception window must exceed 24 clock periods + 3 us. For Reset with high repetition rate it's recommended to switch the dualport access mode to **Denied** to avoid dualport errors.

Watchdog Reset (only 552/562)	Different from the original CPU, the Emulator generates no RESET pulse for the **external units** in a watchdog reset cycle. The internal RESET is executed.
ADC (only 552 and 562)	There is an incompatibility between the 552 and the 562 concerning the ADC resolution and the conversion time. The resolution is **always** 10 bit resp. 50 clock cycles conversion time. For use without a target, the AVREF+ and AVREF- have 10 kΩ and AVSS and AVDD 10 Ω in series.
Operation Mode (only M517E)	Don't use command **SYStem.Up** for emulation without target system. Use always **SYStem.Mode AloneInt**. Otherwise errors can appear, because the CPU will start up with external clock from a slow auxiliary oscillator of the 80517 and will not use the internal clock of the emulator.
Additional SFRs (only 80C515/80C535)	The special function registers of the 80C517 are also available when emulating the 80C515. For correct emulation of the 80C515/80C535 don't use the following SFR's: 0ECH, 0EDH, 0EEH, 0EFH, 0F6H, 0F7H, 0FAH.
XRAM Access (only 515A/517A)	When the XRAM is enabled, the XMAP1 SFR must be set, otherwise the breakpoints and analyzer trace will not work in this address range.
DMA cycles	The trigger unit can't distinguish between a DMA-READ and a DMA-WRITE cycle. The readflag and the write flag are set correctly. All DMA accesses are displayed in the trace as 'RW-DMA'. The address, the DATA and the timestamp of a DMA record is not correct when memory to memory DMA transfers are made in the internal RAM.
Emulation break during DMA transfer (80152)	If a break appears while a DMA-channel is transferring data, the DMA stops and can't be restarted automatically. Normally the last executed cycles of the DMA transfer are running in the emulation monitor program, and therefore they are not sampled by the analyzer. If a DMA cycle is in progress, the transfer will be finished (including burst mode), before the break sequence takes place.
Interrupts during Single Step	To prevent the execution of interrupts from internal sources during assembler and HLL single stepping, the commands **SETUP.IMASKASM** and **SETUP.IMASKHLL** must be used.

This additional register is only available if IOSTOP is active. The register concerns the current interrupts in progress and it is called Interrupt Status Register ISR (at location D:9E). The original CPU does not incorporate this register. The ISR is invisible while the user program is running. A RESET sets the ISR to 0FFH. When an interrupt of level 0 or level 1 occurs, the corresponding level code appears as defined below. Depending on the selected CPU, some of the interrupt sources may be inhibited.

```
  7   6   5   4      3   2   1   0

<Level 1 Code>     <Level 0 Code>
```

Internal Source	Level Code	51	851	662	652	562	552
external 0	0 0 0 0	x	x	x	x	x	x
timer 0	0 0 0 1	x	x	x	x	x	x
external 1	0 0 1 0	x	x	x	x	x	x
timer 1	0 0 1 1	x	x	x	x	x	x
SIO 0*)	0 1 0 0	x	x	x	x	x	x
E2PROM*)	0 1 0 0		x				
SIO 1	0 1 0 1				x		x
T2 capt. 0	0 1 1 0					x	x
T2 capt. 1	0 1 1 1					x	x
T2 capt. 2	1 0 0 0					x	x
T2 capt. 3	1 0 0 1					x	x
ADC complete	1 0 1 0					x	x
T2 compare 0	1 0 1 1					x	x
T2 compare 1	1 1 0 0					x	x
T2 compare 2	1 1 0 1					x	x
T2 overflow	1 1 1 0					x	x

*) Within the 83C581 mode, check flags RI, TI and IFE to decide weather a SIO 0 or E2PROM interrupt has occurred.

This additional register is only available if IOSTOP is active. The register concerns the current interrupts in progress and it is called Interrupt Status Register ISR (at location D:9E). The original CPU does not incorporate this register. The ISR is invisible while the user program is running. A RESET sets the ISR to 0FFH. When an interrupt occurs, the corresponding level code appears as defined below. Depending on the selected CPU, some of the interrupt sources may be inhibited.

```
  7   6   5   4       3   2   1   0

<Level 1 Code>      <Level 0 Code>
```

Internal Source	Level Code
external 0	0 0 0 0
timer 0	0 0 0 1
external 1	0 0 1 0
timer 1	0 0 1 1
SIO 0	0 1 0 0
I2C	0 1 0 1
Derivative Int1	0 1 1 0
Derivative Int2	0 1 1 1
Derivative Int3	1 0 0 0
Derivative Int4	1 0 0 1
Derivative Int5	1 0 1 0
Derivative Int6	1 0 1 1
Derivative Int7	1 1 0 0
Derivative Int8	1 1 0 1
Derivative Int9	1 1 1 0

There are two additional Register available if IOSTOP is active. The registers concern the current interrupts in progress and they are called Interrupt Status Register IS0 (D:0FD) and IS1 (D:0FE). The original CPU does not have this register. The IS0 and IS1 are only readable and invisible while user program is running. A RESET set the IS0/1 to 0FFH. When an interrupt of level 0, 1, 2 or level 3 occurs, the corresponding level code appears as defined below.

```
IS0, IS1                    7   6   5   4          3   2   1   0
IS0                         <  Level 1  >          <  Level 0  >
IS1                         <  Level 2  >          <  Level 3  >
```

Internal Source	Level Code
external 0	0 0 0 0
timer 0	0 0 0 1
external 1	0 0 1 0
timer 1	0 0 1 1
SIO 0	0 1 0 0
timer 2	0 1 0 1
SIO 1	0 1 1 0
comp. timer	0 1 1 1
A/D converter	1 0 0 0
external int 2	1 0 0 1
external int 3	1 0 1 0
external int 4	1 0 1 1
external int 5	1 1 0 0
external int 6	1 1 0 1

Setting read or write breakpoints to internal direct or bit addressed memory is possible. The emulator hardware tracks the executed code and triggers on instructions that access the specified location. Indirect addressed accesses to the breakpoint location will not trigger the breakpoint:

```
b.s d:0x40 /w                ; set direct addressed breakpoint
...

mov a,0x40                   ; the breakpoint is triggered

mov r0,#40                   ; the breakpoint is not triggered
mov a,@r0

b.s b:0x0 /r                 ; set bit addressed breakpoint
...

movb 0x0,c                   ; the breakpoint is triggered

mov r0,#20                   ; the breakpoint is not triggered
mov @r0,a
```

The breakpoint list commands list only one memory class:

```
b.l                          ; list program breakpoints

b.l x:                       ; list breakpoints in external data memory

b.l d:                       ; list breakpoints in direct addressed mem.
```

Format:	**SYStem.Line** <*option*>
<*option*>:	**EA [ALways | ON | OFF]** **EBEN [ALways | ON | OFF]**

The options can be changed only if the system is 'down'. The **EBEN** line is only available on the 80152.

EA ALways EA-line is always active. This option must be chosen if a chip without bond-out capabilities is emulated (e.g 8031).

EA ON EA-line is connected to the target system.

EA OFF EA-line is always off.

EBEN ALways EBEN-line is always active. The opfetch is made via P5/P6.

EBEN ON EBEN-line is connected to the target system.

EA OFF EBEN-line is always off.

Only the following combinations of LINE EA and LINE EBEN are allowed:

LINE EA	LINE EBEN	Opfetch via
always	off	P0,P2
off	always	P5,P6,P0,P2
always	always	P5,P6
on	on	target bustype

Format: **SYStem.Line** <*option*>

<*option*>: **EW [OFF | Running | ON | Always]**
EOW [OFF | Running | ON | Always]
HWPD [OFF | ON]

The options can be changed only if the system is 'down'. The **EW** and **EOW** lines are only available on some specific controllers (80582,80517).

EW OFF	Watchdog always disabled, independent of target EW.
EW Running	Switched to target EW, while user program is running, otherwise OFF.
EW ON	Always connected to the target EW.
EW Always	Always enabled.
EOW OFF	Oscillator watchdog always disabled, independent of target EW.
EOW Running	Switched to target EW, while user program is running, otherwise OFF.
EOW ON	Always connected to the target EW.
EOW Always	Always enabled.
HWPD	When activated the hardware power down feature is enabled.

SYStem.Option DUMMY DUMMY cycles

Format: **SYStem.Option DUMMY [OFF | ON]**

The option can be changed only if the system is 'down'. DUMMY-Cycles are CPU-cycles for internal operation, all accesses to memory are discard. Normally this cycle gives no practicable informations about program flow. If DUMMY is off, the analyzer can't record the dummy-cycle and the trigger-unit can't recognize dummy-cycles too. In prestore mode the analyzer records the last opfetch cycle which was executed before the data memory access as a prestore address. If DUMMY is on, a wrong prestore address can occur. Either the prestore address seems to be the next opfetch behind the data access (but that is true, because the CPU makes a dummy cycle before the data access with the address of the next opcode), or the prestore address is the address of a dualport access while access mode Fast or Advanced. Switch DUMMY off decreases the really number of cycles counting in the counter window.

SYStem.Option IOSTOP — Stop peripherals

Format: **SYStem.Option IOSTOP [OFF | ON]**

The options can be changed only if the system is “down”. The IOSTOP option has no effect while the user program is running. If user program is not running and IOSTOP is switched on, all internal timers stop, all interrupts are inhibit, UART stops after sending or receiving actual data and inhibits capture registers and the TR2 input. IOSTOP has no effect on the AD converter, the PWM circuitry, the I2C logic and the EEPROM. When IOSTOP is switched off, all internal IO devices keep running while emulation stops. If IOSTOP is active and EA="1" and the program memory is mapped external, a Data window **shows not the correct memory contents**, but the program is still running correct.

Format: **SYStem.Option DMA [OFF | ON]**

The options can be changed only if the system is 'down'. This option must be activated, when the DMA's of the 80152 are used. The dualport access mode must be **Slow** or **Denied** in this case. **LINE EA** should be set to **ALways** and **LINE EBEN** to **OFF** for the 80C152JB. Depending on the emulated chip some restrictions exist with the 80152:

```
80C152JA DIL  : SFR     -       -       -
83C152JA DIL  : SFR  DMA-C      -       -
80C152JA PLCC : SFR     -       -      PIN
83C152JA PLCC : SFR  DMA-C      -      PIN

80C152JC DIL  : SFR     -       -       -
80C152JC PLCC : SFR     -       -      PIN

80C152JB PLCC :  -   DMA-P5  FETCH-P5   -
80C152JD PLCC : SFR  DMA-P5  FETCH-P5   -
```

SFR Use only the SFR of the current emulated CPU. All other SFR's of the emulation CPU (80C152J) are available, but not relevant for a correct emulation.

DMA-C In the microcontroller mode, the fetch of the program is performed via the port P5/P6 of the 80C152JB. In this bus mode the emulator can't support DMA-Cycles.

DMA-P5 If the alternative bus modes (program access not via P0/P2) are selected, the emulator can't support DMA-accesses.

Fetch-P5 If the alternative bus modes (Opfetch not via P0/P2) was selected, the memory must be mapped internal and the external EPROM must be removed from the target. It’s not possible to run a program from the external program memory.

PIN Do not connect the NC pins with any signal.

Format: **SYStem.Option TestClock [ON | OFF]**

ON The clock test circuit is active. Clock fails will be detected by the emulator system. The emulator changes to reset state.

OFF No clock check. The external clock may be switched off, but no trace of program and data is possible.

eXception.Activate — Force exception

Format:	**eXception.Activate RES [ON | OFF]**
Format:	**eXception.Activate OFF**

RES Activates the RES line.

OFF No activation of any exception line.

eXception.Enable — Enable exception

Format:	**eXception.Enable RES [ON | OFF]**
Format:	**eXception.Enable OFF**
Format:	**eXception.Enable ON**

RES Enables the RES line.

ON Enables all exception line.

OFF Disables all exception lines.

Format:	**eXception.Trigger INT0 [ON | OFF]**
Format:	**eXception.Trigger INT1 [ON | OFF]**
Format:	**eXception.Trigger Pulse** [**ON**|**OFF**]
Format:	**eXception.Trigger RES [ON | OFF]**
Format:	**eXception.Trigger T0 [ON | OFF]**
Format:	**eXception.Trigger T1 [ON | OFF]**
Format:	**eXception.Trigger OFF**
Format:	**eXception.Trigger ON**

INT0	Trigger on INT0 line.
INT1	Trigger on INT1 line.
P	Trigger on P line.
RES	Trigger on RES line.
T0	Trigger on T0 line.
T1	Trigger on T1 line.
ON	Trigger on all exception lines.
OFF	No trigger on any exception lines.

Format:	**eXception.Pulse INT0** <*width*> <*period*>
Format:	**eXception.Pulse INT1** <*width*> <*period*>
Format:	**eXception.Pulse RES** <*width*> <*period*>
Format:	**eXception.Pulse T0** <*width*> <*period*>
Format:	**eXception.Pulse T1** <*width*> <*period*>
Format:	**eXception.Pulse OFF**

INT0	Stimulate INT0 line.
INT1	Stimulate INT1 line.
RES	Stimulate RES line.
T0	Stimulate T0 line.
T1	Stimulate T1 line.
OFF	No stimulation on any exception line.

Banked Target Systems

In banked systems the upper address lines are either supplied internally or by the external bank probe. 8 additional lines offer 256 different memory banks. Accessing the different pages is done by extending all memory and pc addresses to 24 bit. The address bits A16 to A23 select the memory bank. Every command which makes a memory access first calls a special bank driver subroutine to select the temporary memory bank. On realtime emulation the bank number is traced on the upper 8 bits of the address bus. The breakpoints function stores the bank address back to the MSB of the program counter.

Format:	**SYStem.BankFile** <*file*>

This command loads the bank driver. The bank driver is a special subroutine to select the actual bank. It is loaded to a reserved area in the emulation monitor program. Loading a special bank driver gives a maximum of flexibility to the user. A bank address delivered by the emulator may be used to set microcontroller ports or external MMUs in the target system. The bank file consists of a code number defining the bank operation mode and a code area which consists of a subroutine to set the correct bank state. Writes to internal CPU ports may be executed directly, while ports in target systems must be accessed by a special system call to address 700H and 715H. Translation of logical bank numbers to physical bank numbers can be done by the **MMU** command. The **BNK** register holds the physical bank number. The **PP** (Program Pointer) register hold the logical 24-bit PC address.

Format:	**SYStem.Bank** <*option*>
<*option*>:	**OFF** **INTernal** **EXTernal**

Internal

Internal banking to support paged EPROMs. The internal bank register is set by writing to an address range selected by the command **MAP.Bank**.

This example uses a common program area on 0--3fffh a banked area from 4000--7fff with 4 banks

```
map.res                                 ; reset mapper
map.mirror  p:0x0--0x03fff p:0x10000    ; mirror for common area
map.mirror  p:0x0--0x03fff p:0x20000
map.mirror  p:0x0--0x03fff p:0x30000
map.bank    0x4000--0x7fff              ; set area of banked eprom
system.bankfile banksel.bnk             ; load bank file
system.up
```

Bank drivers are special subroutines (max. length 256 bytes) to set the bank or an external MMU:

```
                org 5ffh

                db   1              ; select internal mode

bank:                               ; destination area in system memory
                org 600h

                push dpl
                push dph
                mov a,r6            ; physical bank
                mov dptr,4000h      ; set DPTR to banked area
                lcall 700h          ; subroutine to write byte to target
                pop dph             ; system setting the page register
                pop dpl             ; in the EPROM
                ret                 ; return
```

External

External banked systems use a register or output pins of the CPU to generate the upper memory addresses. These lines must be feedback to the emulator with the bank probe. Unused inputs of the bank probe must be grounded (or jumpered to ground pin).

This example uses a common program area on 0--3fffh a banked area from 4000--7fff with 4 banks

```
map.res                                 ; reset mapper
map.mirror  p:0x0--0x3fff p:0x10000     ; mirror for common area
map.mirror  p:0x0--0x3fff p:0x20000
map.mirror  p:0x0--0x3fff p:0x30000
system.bankfile banksel.bnk             ; load bank file
system.up
```

This example selects the bank by **internal** port 3 bit 2 and 4:

```
; bank switching program

input:          r0,r1               ; address
                r6                  ; bank
                r4                  ; read/write
                r5                  ; program/data

usable:         r7,a,b,psw
```

```
                stack & direct
                addressable
                memory 8--1f
                Hex

; start address of program is 600h

byte 0:         0                   ; no banking
                1                   ; internal banking (banked eprom)
                2                   ; external banking (from pod)
```

```
                org 5ffh

                db   2              ; using external banking logic

                org 600h

banking:        mov  a,r6           ; bank to acc
                mov  c,acc.0
                mov  p3.2,c         ; address bit 16 to port 3, bit 2
                mov  c,acc.1
                mov  p3.4,c         ; address bit 17 to port 3, bit 4
                ret
```

Now the bank select is done by an **external** register selected at A0h:

```
org 5ffh

                db   2              ;select external mode

bank:                               ;destination area in system memory
                org 600h

                push dpl
                push dph
                mov a,r6            ;physical bank
                mov dptr,0a0h       ;set DPTR to banked area
                lcall 700h          ;subroutine to write byte to target
                pop dph             ;system setting the bank register
                pop dpl             ;in the target
                ret                 ;return
```

The next examples shows the map and load commands for translated bank numbers:

This example uses a common program area on 0--7fffh and a banked area from 8000--ffff with 4 banks. The translation from logical banks to physical banks is as follows:

logical 0 -> physical 7

logical 1 -> physical 4

logical 2 -> physical 5

logical 3 -> physical 2

```
system.bankfile banksel.bnk                ; load bank file (uses
                                           ; physical banks)
system.mode ai

map.res                                    ; reset mapper
map.mirror  p:0x0--0x7fff p:0x70000        ; mirror for common area
map.mirror  p:0x0--0x7fff p:0x40000
map.mirror  p:0x0--0x7fff p:0x50000
map.mirror  p:0x0--0x7fff p:0x20000

map.ram p:0--0ffff                         ; map memory in (physical)
                                           ; banks
map.ram p:78000--7ffff
map.ram p:48000--4ffff
map.ram p:58000--5ffff
map.ram p:28000--2ffff
map.intern

symbol.reset
mmu.reset
mmu.create p:00000--07fff p:00000--07fff
mmu.create p:08000--0ffff p:78000--7ffff
mmu.create p:00000--07fff p:70000--77fff
mmu.create p:18000--1ffff p:48000--4ffff
mmu.create p:00000--07fff p:40000--47fff
mmu.create p:28000--2ffff p:58000--5ffff
mmu.create p:00000--07fff p:50000--57fff
mmu.create p:38000--3ffff p:28000--2ffff
mmu.create p:00000--07fff p:20000--27fff
mmu.on

d.load.o applic.omf /ext /p /nc            ; load file from BL51 (KEIL)
```

Addr	Function	Address	Data	Result
700H	MemWrite	DPTR	A	-
715H	MemRead	DPTR	-	A

Memory Classes

Access Class	Description
P	Program
X	External Data
XP	External or Program
D	Internal direct access
I	Internal indirect access
B	Internal bit addressing
EP	Program emulation memory access
EX	External Data emulation memory access
C	CPU-access
E	Emulation memory access

Keywords for the Trigger Unit

General 8051 Keywords for the Trigger Unit

Input Event	Meaning	Analyzer Hardware			
		ECC8	HAC	HA120	SA120
BRANCH	Opfetch after jump on condition			X	X
CDATA, READCODE	MOVC cycle	X	X	X	X
DATA	**CDATA** or **XDATA**	X	X	X	X
DIRECT	**DIRectReaD** or **DIRectWRite**	X	X	X	X
DIRectBIT	**DIRRDBIT** or **DIRWRBIT** or **DIRRD-WRBIT**	X	X	X	X
DIRectBYTE	**DIRRDBYTE** or **DIRWRBYTE** or **DIRRDWRBYTE**	X	X	X	X
DIRectReaD	**DIRRDBIT** or **DIRRDBYTE** or **DIRRDWRBIT** or **DIRRDWRBYTE**	X	X	X	X
DIRectWRite	**DIRWRBIT** or **DIRWRBYTE** or **DIRRDWRBIT** or **DIRRDWRBYTE**	X	X	X	X
DIRRDBIT	Address constant for a internal bit read cycle read	X	X	X	X
DIRRDBYTE	Address constant for a internal direct read cycle read	X	X	X	X
DIRRDWRBIT	Address constant for a internal bit modify cycle read	X	X	X	X
DIRRDWRBYTE	Address constant for a internal byte modify cycle read	X	X	X	X
DIRWRBIT	Address constant for a internal bit write cycle read	X	X	X	X
DIRWRBYTE	Address constant for a internal byte write cycle read			X	X
DUMMY	Discard fetch	X	X	X	X
eXternal, **XDATA**	**EXTREAD** or **EXTWRITE**	X	X	X	X
EXTernalREAD	External data read cycle	X	X	X	X
EXTernalWRITE	External data write cycle	X	X	X	X

FETCH	**OPFETCH** or **FETCH1** or **FETCH2** or **DIRECT**	X	X	X	X
FETCH1	Fetch first operand	X	X	X	X
FETCH2	Fetch second operand	X	X	X	X
IACK	interrupt acknowledge cycle	X	X	X	X
INT	**INT0** or **INT1**			X	X
INT0, P32	Interrupt 0			X	X
INT1, P33	Interrupt 1			X	X
MULDIVCYC	Multiply or division cycle	X	X	X	X
OPFetch	Program memory read cycle	X	X	X	X
PORT	Signal from port analyzer			X	X
Program	**FETCH** or **DUMMY** or **READCODE** or **MULDIVCYC**	X	X	X	X
P30 .. P37	Port 3x			X	X
Read	**DIRectReaD** or **EXTREAD**	X	X	X	X
READCODE, CDATA	MOVC cycle	X	X	X	X
RXD, **P30**	ReceiverData cycle		X	X	X
TIMER	**T0** or **T1**			X	X
Timer0, **P34**				X	X
Timer1, **P35**				X	X
TXD, **P31**	TransmitterData cycle			X	X
Write	**DIRectWRite** or **EXTWRITE**	X	X	X	X
XDATA, eXternal	**EXTREAD** or **EXTWRITE**	X	X	X	X

80152 Keywords for the Trigger Unit

Input Event	**Meaning**	**Analyzer hardware**			
		ECC8	**HAC**	**HA120**	**SA120**
DMACycle	DMA cycle	X	X	X	X

For not CPU-specific keywords, see **non-declarable input variables** in **"ICE/FIRE Analyzer Trigger Unit Programming Guide"** (analyzer_prog.pdf).

P10 .. **P17**	Port 1
B0 .. **B7**	Bank probe inputs

Dequeueing

The disassembled lines in the analyzer are displayed after the last record of the opfetch.

If an interrupt acknowledge cycle was sampled by the analyzer, the last opfetch before the cycle "intack" was not executed. If DUMMY is off two "intack" cycles appear, otherwise three cycles are shown in the analyzer list window.

While multiplication and division, a pseudo cycle "MULDIV" appears in the analyzer list window.

Port Analyzer

The port analyzer of the 80517 is connected to ports 0,1,2,3,4,5,6 and 7, port 8 cannot be traced. On 80152 probes the port analyzer is always connected with P5/P6 of the target.

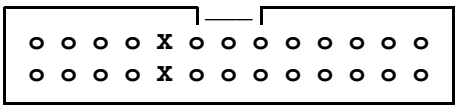
Keywords for the Port Analyzer

00 .. **07**	Port 0
10 .. **17**	Port 1
20 .. **27**	Port 2
30 .. **37**	Port 3
40 .. **47**	Port 4
50 .. **57**	Port 5
X0 .. **X7**	Port 6 or free channels
Y0 .. **Y7**	Port 7 or free channels

Additional Trace Channels

Module 8051

Front view:

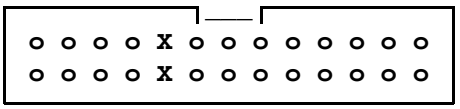


G G G G 7 6 5 4 3 2 1 0

G = GND, X= NC, P40--P47, P50--P57

Module M582

Front view:

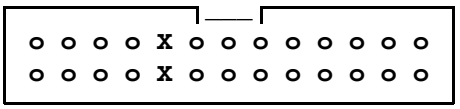


G G G G 7 6 5 4 3 2 1 0

G = GND, X= NC, PX0--PX7, PY0--PY7

Adapter M582-C562

Front view:

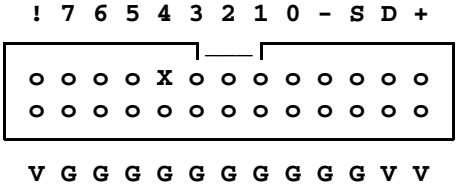


G G G G 7 6 5 4 3 2 1 0

G = GND, X= NC, P40--P47, P50--P57

Adapter M582-C552

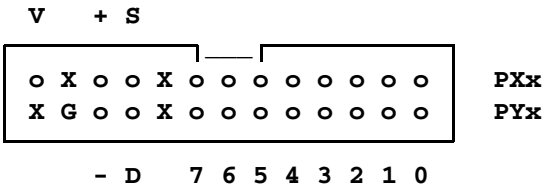
Front view:



G = GND, V = +5V, P50--P57, ! = internal use
- = AVREF-, S = AVSS, D = AVDD , + = AVREF+
X = NC

Module M592

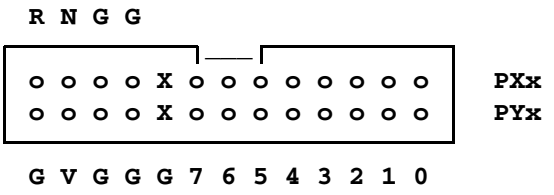
Front view:



G = GND, X= NC, PX0--PX7, PY0--PY7
- = AVREF-, + = AVREF+, S = AVSS, D=AVDD

Module S517-C535

Front view: (backwards)

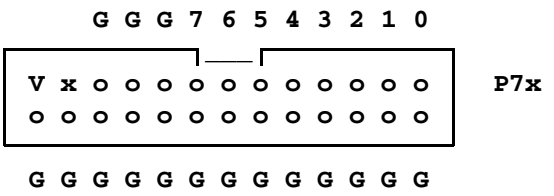


G = GND, X= NC, PX0--PX7, PY0--PY7
V = +5V, R = VAREF, N = VAGND

For use without target, the AVREF have 10 kΩ
in series.

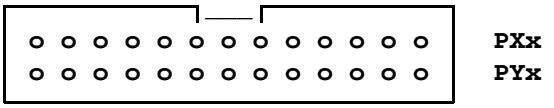
Pin 4 (VPD) of the socket of the 80C515/535 is not connected.
Pin 37 (VBB) of the socket of the 80C515/535 is not connected.

Module 80152



G = GND, X= NC, V = +5 V (max. 20 mA)

Front view:



G G G G G 7 6 5 4 3 2 1 0

G = GND, PX0--PX7, PY0--PY7

Compilers

Language	Compiler	Company	Option	Comment
ASM	A8051	Ashling Microsystems Ltd.	SYM	with converter
ASM	A8051	IAR Systems AB	UBROF	Source level debugging
C	SDCC	GNU Compiler Collection	CDB	
C	ICC8051	IAR Systems AB	UBROF	Banking support
C	C51	ARM Germany GmbH	EOMF-51	
C	CC	Small Device C Compiler	COFF	
PASCAL	SYSTEM51-PASCAL	KSC Software Systems	OMF-51	No type information
PLM	PL/M-51	Intel Corporation	OMF-51	
PLM	PLM51	TASKING	IEEE	OMF also possible

3rd-Party Tool integrations

CPU	Tool	Company	Host
	WINDOWS CE PLATF. BUILDER	-	Windows
	CODE::BLOCKS	-	-
	C++TEST	-	Windows
	ADENEO	-	
	CODEWRIGHT	Borland Software Corporation	Windows
	CODE CONFIDENCE TOOLS	Code Confidence Ltd	Windows
	CODE CONFIDENCE TOOLS	Code Confidence Ltd	Linux
	EASYCODE	EASYCODE GmbH	Windows
	ECLIPSE	Eclipse Foundation, Inc	Windows
	RHAPSODY IN MICROC	IBM Deutschland GmbH	Windows
	RHAPSODY IN C++	IBM Deutschland GmbH	Windows
	CHRONVIEW	Inchron GmbH	Windows

CPU	Tool	Company	Host
	LDRA TOOL SUITE	LDRA Technology, Inc.	Windows
	UML DEBUGGER	LieberLieber Software GmbH	Windows
	TRACEANALYZER	LUXOFT	Windows
	SIMULINK	The MathWorks Inc.	Windows
	ATTOL TOOLS	MicroMax Inc.	Windows
	VISUAL BASIC INTERFACE	Microsoft Corporation	Windows
	LABVIEW	NATIONAL INSTRUMENTS Corporation	Windows
	TPT	PikeTec GmbH	Windows
	X-TOOLS / X32	PTC	Windows
	CANTATA	QA Systems Ltd	Windows
	RAPITIME	Rapita Systems Ltd.	Windows
	TESSY	Razorcat Development GmbH	Windows
	DA-C	RistanCASE	Windows
	ECU-TEST	TraceTronic GmbH	Windows
	UNDODB	Undo Software	Linux
	TA INSPECTOR	Vector	Windows
	VECTORCAST UNIT TESTING	Vector Software	Windows
	VECTORCAST CODE COVERAGE	Vector Software	Windows

Realtime Operation Systems

Company	Product	Comment
CMX Systems Inc.	CMX-RTX	
ARM Germany GmbH	RTX51/-tiny	

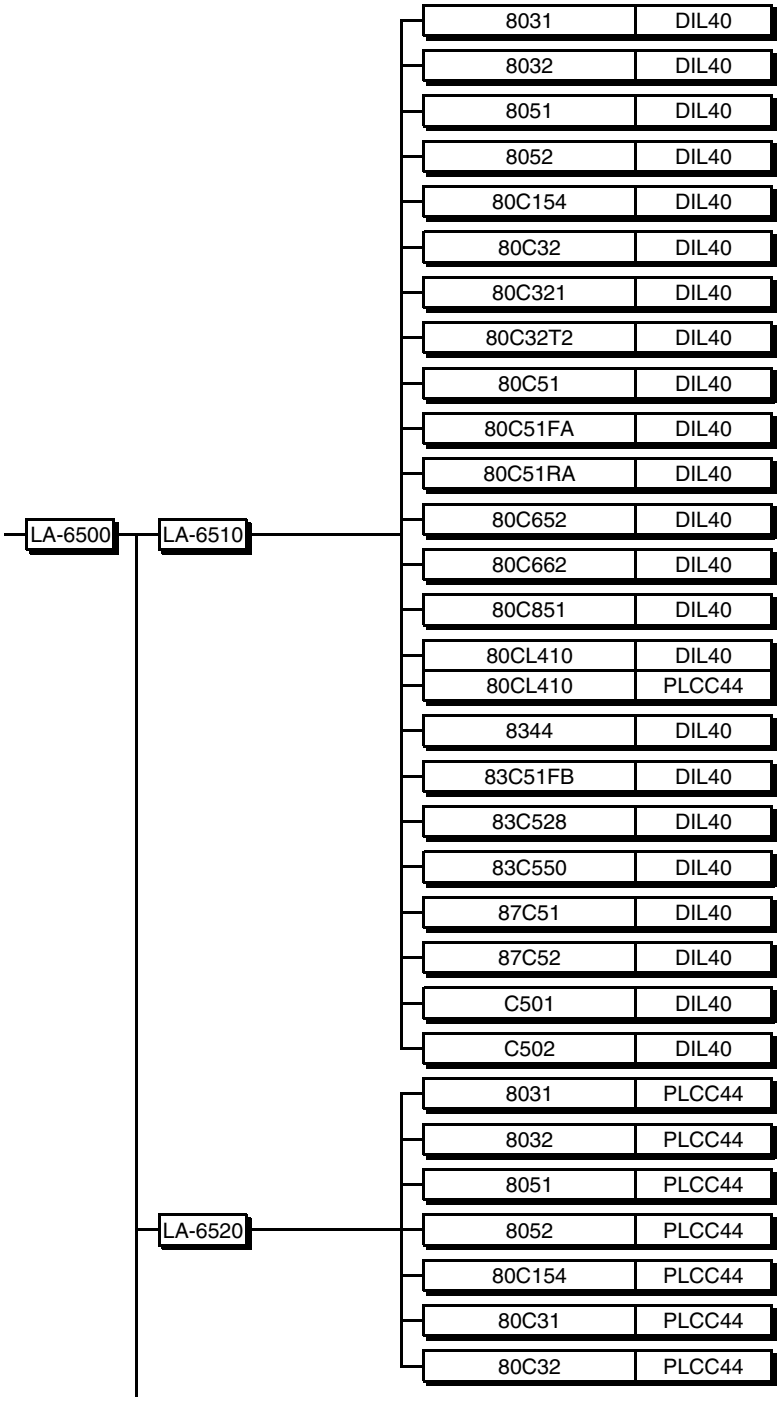
Emulation Frequency

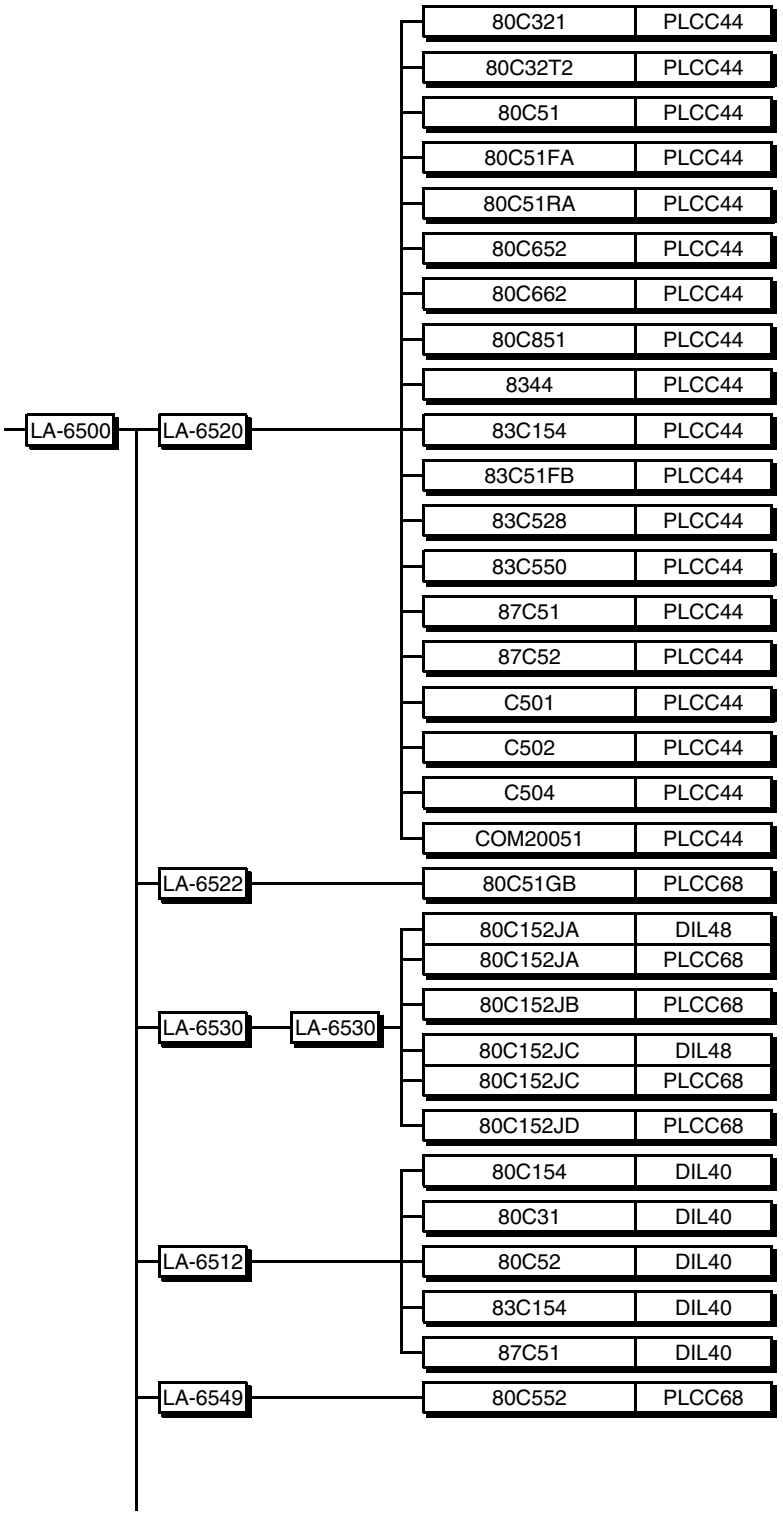
The emulation probe is designed for running with CPUs up to 30 MHz. The max. speed is limited by the used processor type.

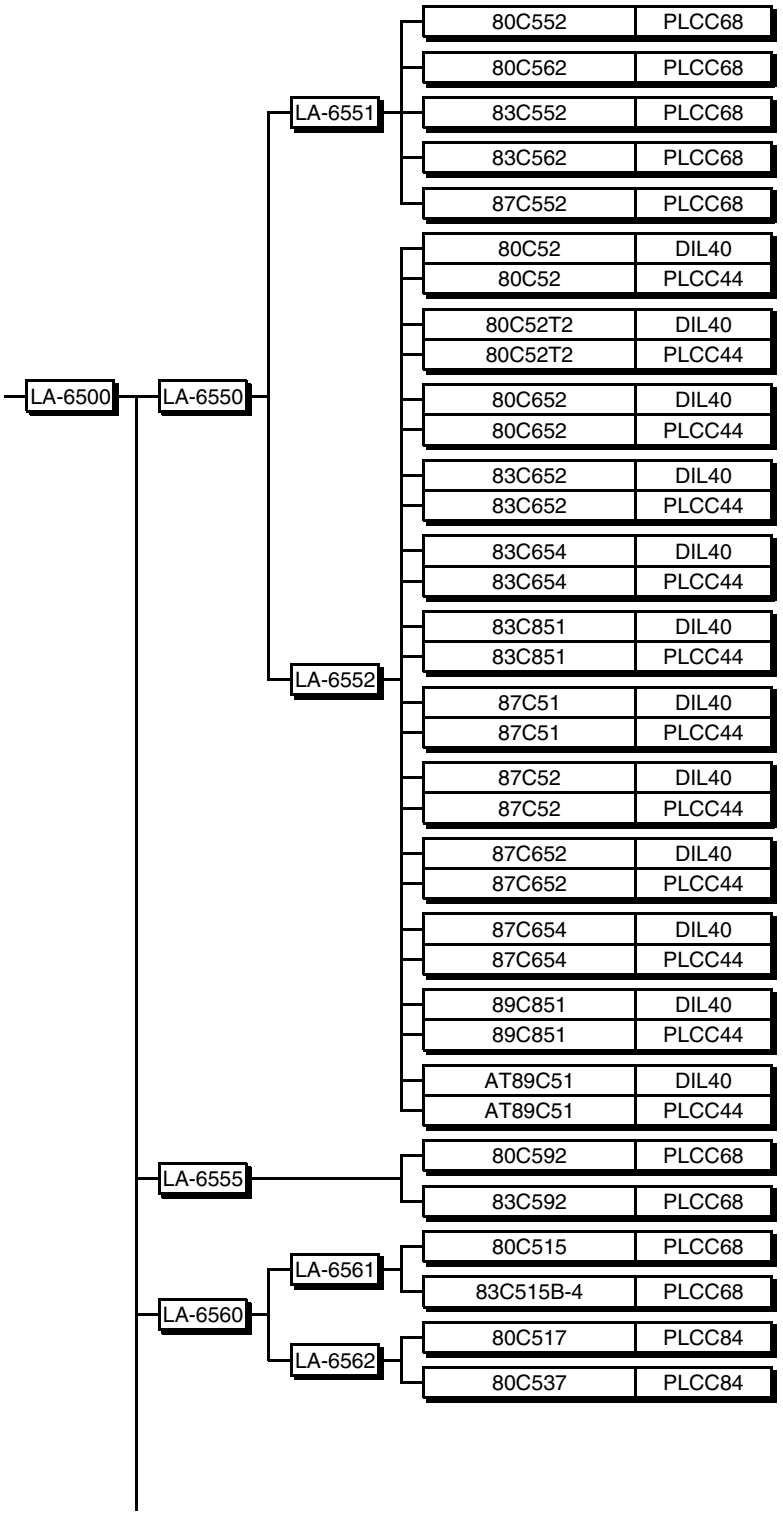
Module	CPU	F-W0-15	F-W0-35	S-W0-15	S-W0-35	S-W1-15	S-W1-35	DRAM
LA-6510	8031	30.0+	30.0+	30.0+	28.6	30.0+	30.0+	
LA-6510	8032	30.0+	30.0+	30.0+	28.6	30.0+	30.0+	
LA-6510	8051	30.0+	30.0+	30.0+	28.6	30.0+	30.0+	
LA-6510	8052	30.0+	30.0+	30.0+	28.6	30.0+	30.0+	
LA-6530	80C152JA	16.0+	16.0+	16.0+	16.0+	16.0+	16.0+	
LA-6530	80C152JB	16.0+	16.0+	16.0+	16.0+	16.0+	16.0+	
LA-6530	80C152JC	16.0+	16.0+	16.0+	16.0+	16.0+	16.0+	
LA-6530	80C152JD	16.0+	16.0+	16.0+	16.0+	16.0+	16.0+	
LA-6510	80C154	22.0+	22.0+	22.0+	22.0+	22.0+	22.0+	
LA-6512	80C31	30.0+	30.0+	30.0+	28.6	30.0+	30.0+	
LA-6510	80C32	30.0+	30.0+	30.0+	28.6	30.0+	30.0+	
LA-6510	80C321	12.0+	12.0+	12.0+	12.0+	12.0+	12.0+	
LA-6510	80C32T2	16.0+	16.0+	16.0+	16.0+	16.0+	16.0+	
LA-6510	80C51	30.0+	30.0+	30.0+	28.6	30.0+	30.0+	
LA-6560	80C515	12.0+	12.0+	12.0+	12.0+	12.0+	12.0+	
LA-6568	80C515A	18.0+	18.0+	18.0+	18.0+	18.0+	18.0+	
LA-6560	80C517	12.0+	12.0+	12.0+	12.0+	12.0+	12.0+	
LA-6567	80C517A	18.0+	18.0+	18.0+	18.0+	18.0+	18.0+	
LA-6510	80C51FA	16.0+	16.0+	16.0+	16.0+	16.0+	16.0+	
LA-6522	80C51GB	16.0+	16.0+	16.0+	16.0+	16.0+	16.0+	
LA-6510	80C51RA	24.0+	24.0+	24.0+	24.0+	24.0+	24.0+	
LA-6550	80C52	16.0+	16.0+	16.0+	16.0+	16.0+	16.0+	
LA-6550	80C52T2	16.0+	16.0+	16.0+	16.0+	16.0+	16.0+	
LA-6570	80C535	12.0+	12.0+	12.0+	12.0+	12.0+	12.0+	
LA-6560	80C537	12.0+	12.0+	12.0+	12.0+	12.0+	12.0+	
LA-6550	80C552	16.0+	16.0+	16.0+	16.0+	16.0+	16.0+	
LA-6550	80C562	30.0+	30.0+	30.0+	28.6	30.0+	30.0+	
LA-6555	80C592	16.0+	16.0+	16.0+	16.0+	16.0+	16.0+	
LA-6510	80C652	12.0+	12.0+	12.0+	12.0+	12.0+	12.0+	
LA-6510	80C662	16.0+	16.0+	16.0+	16.0+	16.0+	16.0+	
LA-6510	80C851	12.0+	12.0+	12.0+	12.0+	12.0+	12.0+	
LA-6510	80CL410	20.0+	20.0+	20.0+	20.0+	20.0+	20.0+	
LA-6510	8344	12.0+	12.0+	12.0+	12.0+	12.0+	12.0+	
LA-6512	83C154	22.0+	22.0+	22.0+	22.0+	22.0+	22.0+	
LA-6570	83C515A	18.0+	18.0+	18.0+	18.0+	18.0+	18.0+	
LA-6568	83C515B-4	12.0+	12.0+	12.0+	12.0+	12.0+	12.0+	
LA-6570	83C517A	18.0+	18.0+	18.0+	18.0+	18.0+	18.0+	
LA-6510	83C51FB	16.0+	16.0+	16.0+	16.0+	16.0+	16.0+	

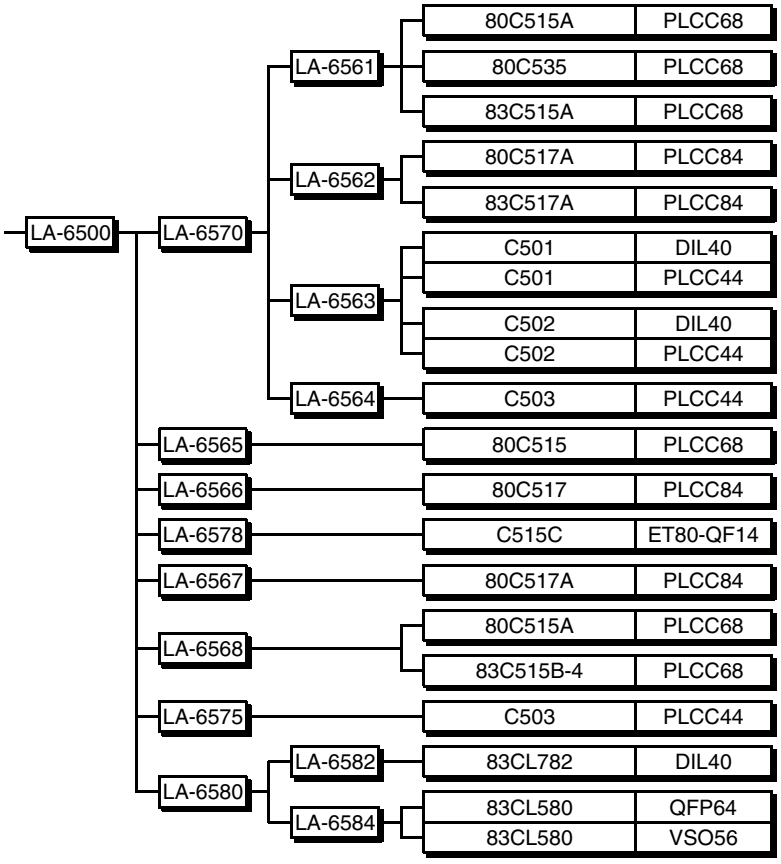
Module	CPU	F-W0-15	F-W0-35	S-W0-15	S-W0-35	S-W1-15	S-W1-35	DRAM
LA-6510	83C528	16.0+	16.0+	16.0+	16.0+	16.0+	16.0+	
LA-6510	83C550	16.0+	16.0+	16.0+	16.0+	16.0+	16.0+	
LA-6550	83C552	16.0+	16.0+	16.0+	16.0+	16.0+	16.0+	
LA-6550	83C562	16.0+	16.0+	16.0+	16.0+	16.0+	16.0+	
LA-6555	83C592	16.0+	16.0+	16.0+	16.0+	16.0+	16.0+	
LA-6550	83C652	16.0+	16.0+	16.0+	16.0+	16.0+	16.0+	
LA-6550	83C654	16.0+	16.0+	16.0+	16.0+	16.0+	16.0+	
LA-6550	83C851	16.0+	16.0+	16.0+	16.0+	16.0+	16.0+	
LA-6580	83CL580	12.0+	12.0+	12.0+	12.0+	12.0+	12.0+	
LA-6580	83CL782	12.0+	12.0+	12.0+	12.0+	12.0+	12.0+	
LA-6512	87C51	12.0+	12.0+	12.0+	12.0+	12.0+	12.0+	
LA-6550	87C52	16.0+	16.0+	16.0+	16.0+	16.0+	16.0+	
LA-6550	87C552	16.0+	16.0+	16.0+	16.0+	16.0+	16.0+	
LA-6550	87C652	16.0+	16.0+	16.0+	16.0+	16.0+	16.0+	
LA-6550	87C654	16.0+	16.0+	16.0+	16.0+	16.0+	16.0+	
LA-6550	89C851	16.0+	16.0+	16.0+	16.0+	16.0+	16.0+	
LA-6550	AT89C51	16.0+	16.0+	16.0+	16.0+	16.0+	16.0+	
LA-6570	C501	18.0+	18.0+	18.0+	18.0+	18.0+	18.0+	
LA-6510	C502	18.0+	18.0+	18.0+	18.0+	18.0+	18.0+	
LA-6570	C503	18.0+	18.0+	18.0+	18.0+	18.0+	18.0+	
LA-6520	C504	30.0+	30.0+	30.0+	28.6	30.0+	30.0+	
LA-6578	C515C	10.0	9.5	9.6	9.1	10.0+	10.0+	
LA-6520	COM20051	16.0+	16.0+	16.0+	16.0+	16.0+	16.0+	

Module Overview



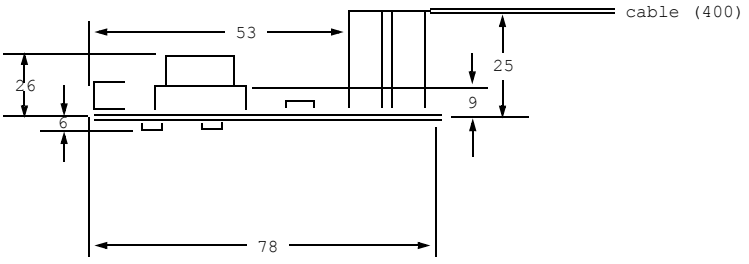




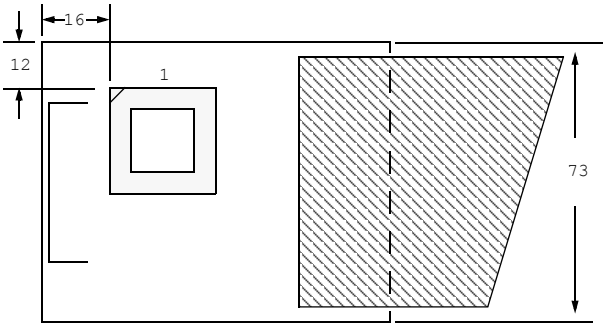


Dimension

LA-6510 M-8051-DIL40



SIDE VIEW

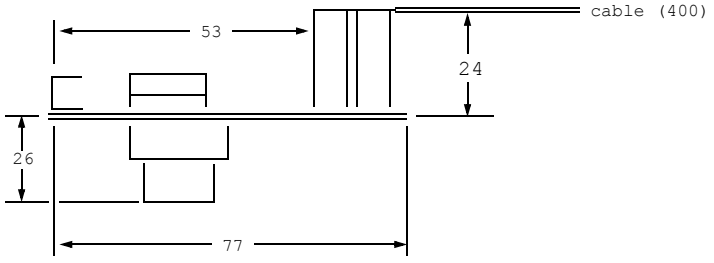


TOP VIEW

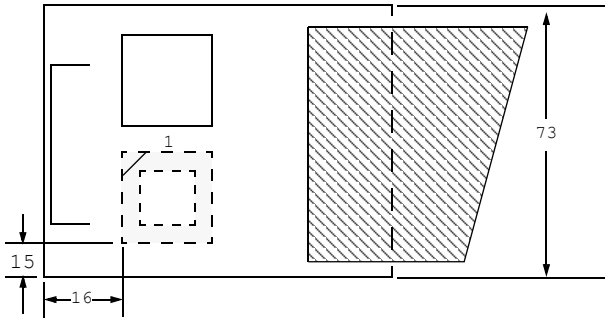
(dimensions define position of socket to target)

Dimension

LA-6520 M-8051-PLCC



SIDE VIEW

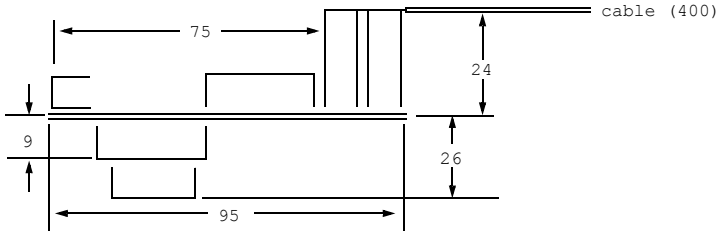


(dimensions define position of socket to target)

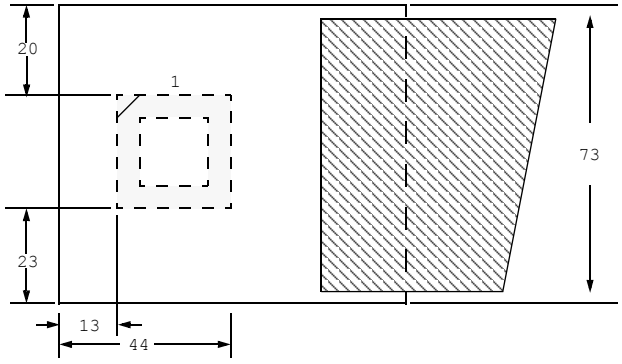
TOP VIEW

Dimension

LA-6522 M-8051GB-PLCC



SIDE VIEW

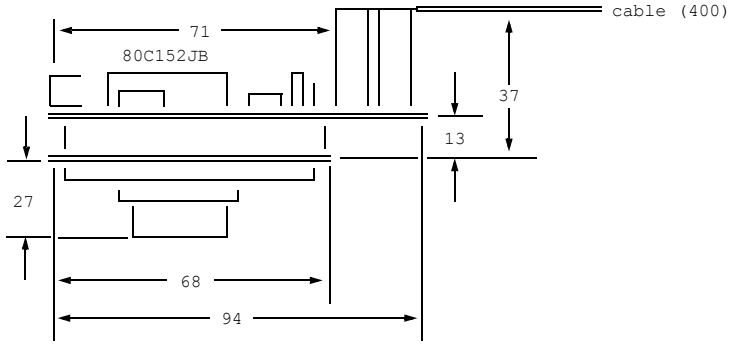


(dimensions define position of socket to target)

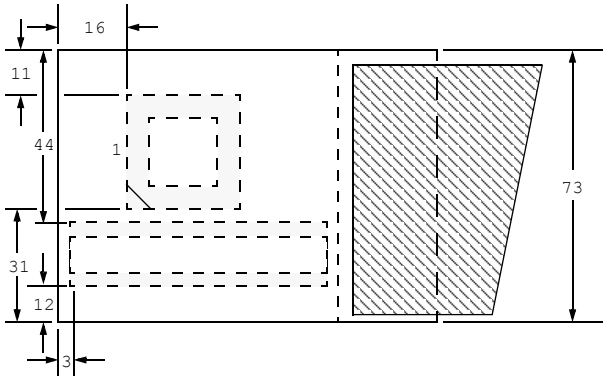
TOP VIEW

Dimension

LA-6530 M-80152-JA/JB



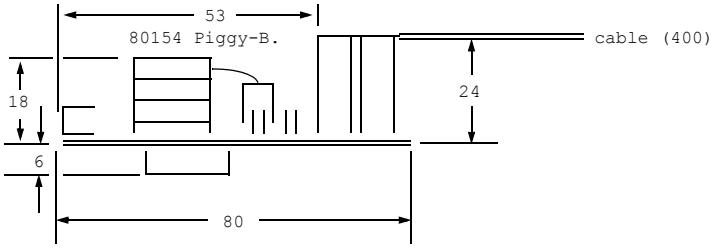
SIDE VIEW



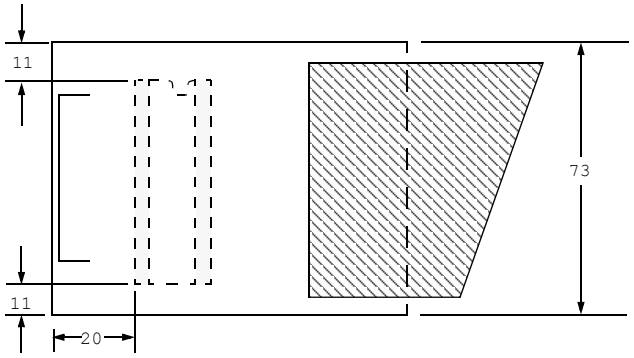
TOP VIEW

Dimension

LA-6512 M-80154-DIL40



SIDE VIEW

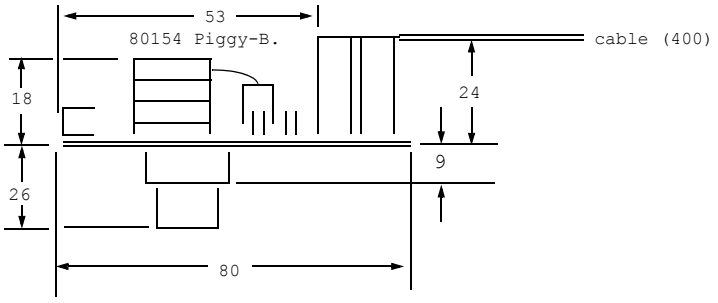


(dimensions define position of socket to target)

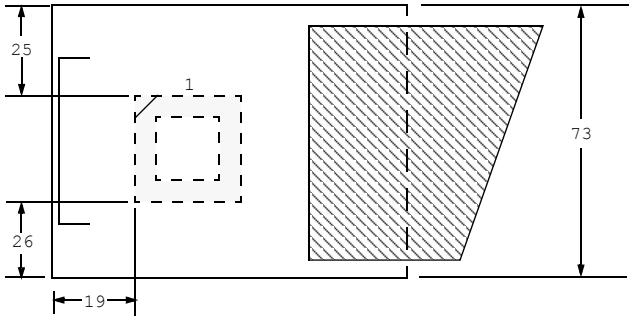
TOP VIEW

Dimension

LA-6514 M-80154-PLCC



SIDE VIEW

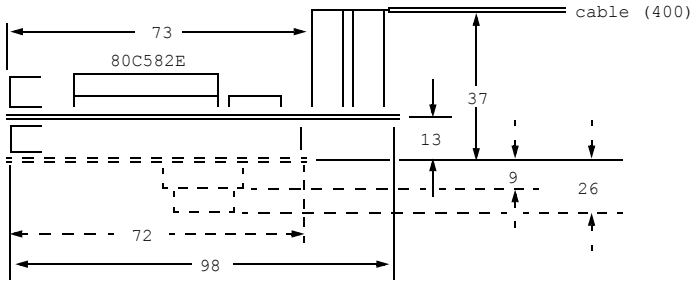


(dimensions define position of socket to target)

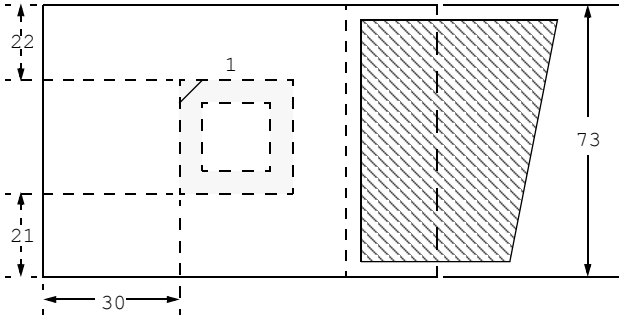
TOP VIEW

Dimension

LA-6550 M-80582-B



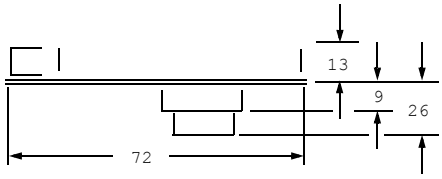
SIDE VIEW



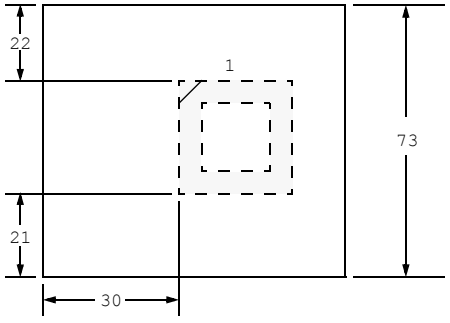
TOP VIEW

Dimension

LA-6551 A-80582-C552



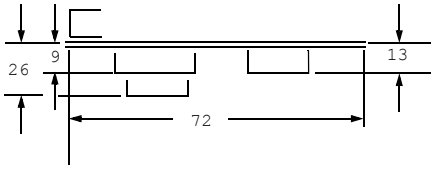
SIDE VIEW



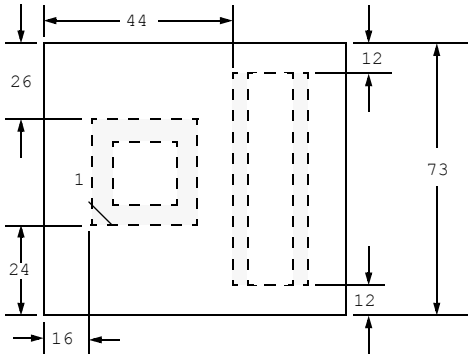
TOP VIEW (all dimension in mm)

Dimension

LA-6552 A-80582-C652



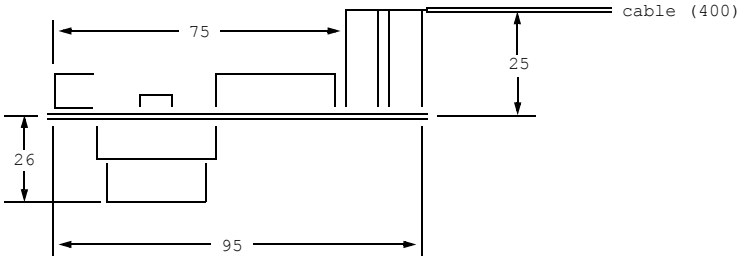
SIDE VIEW



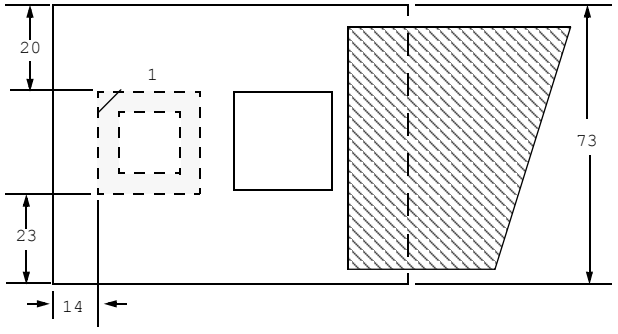
TOP VIEW (all dimension in mm)

Dimension

LA-6555 M-80592



SIDE VIEW

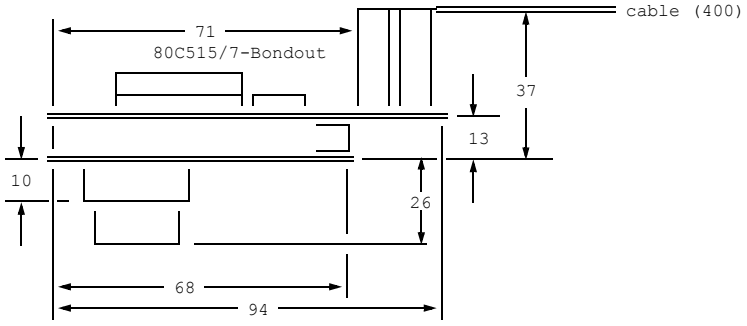


(dimensions define position of socket to target)

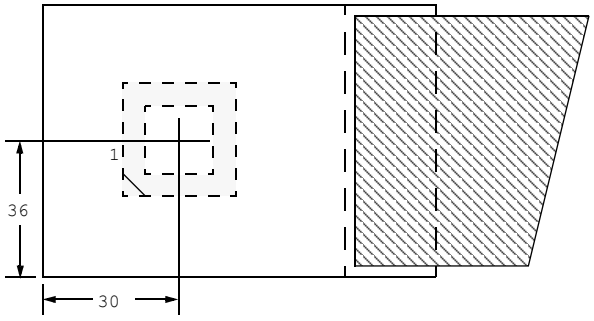
TOP VIEW

Dimension

LA-6560 M-80517



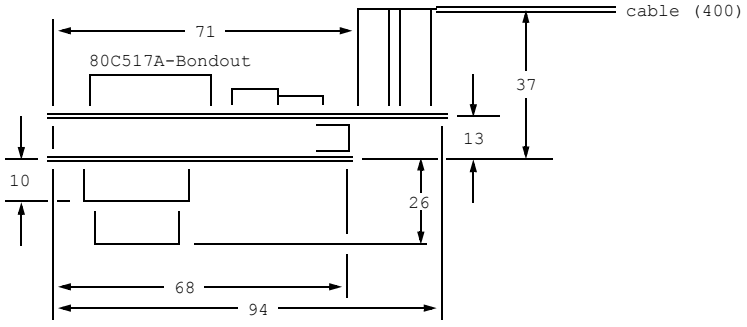
SIDE VIEW



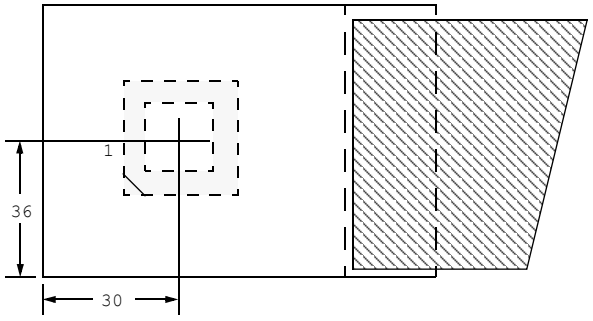
TOP VIEW

Dimension

LA-6570 M-80517A



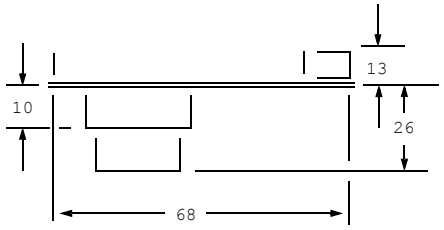
SIDE VIEW



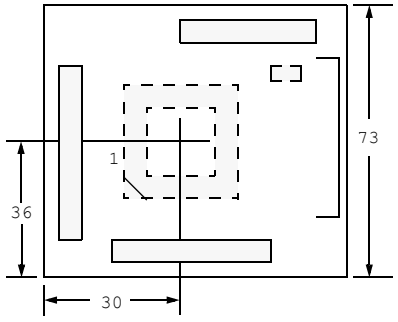
TOP VIEW

Dimension

LA-6561 A-80517-515



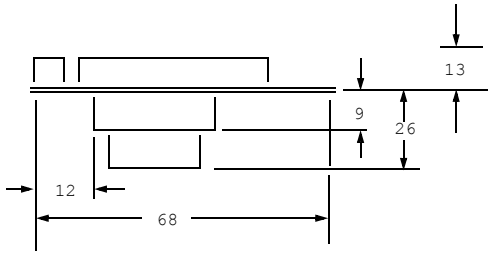
SIDE VIEW



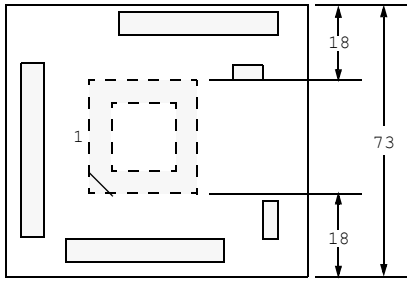
TOP VIEW (all dimension in mm)

Dimension

LA-6562 A-80517-517



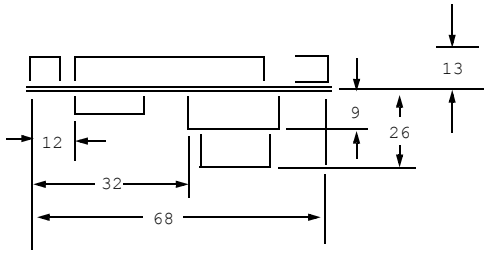
SIDE VIEW



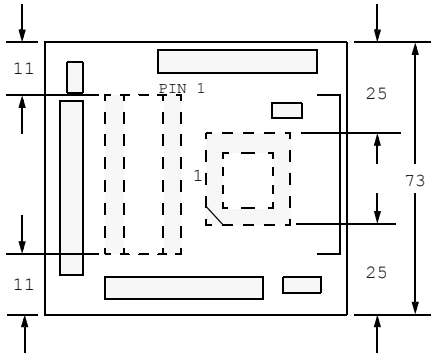
TOP VIEW (all dimension in mm)

Dimension

LA-6563 A-80517-C502



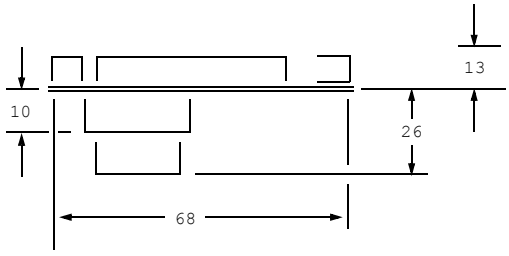
SIDE VIEW



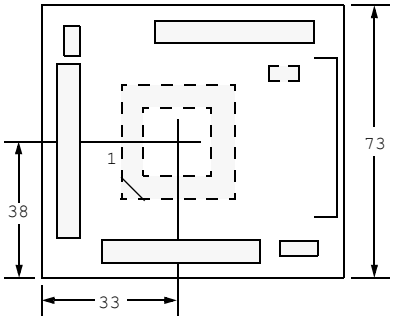
TOP VIEW (all dimension in mm)

Dimension

LA-6564 A-80C517-C503



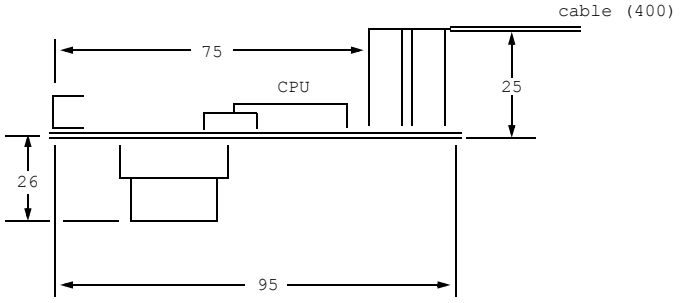
SIDE VIEW



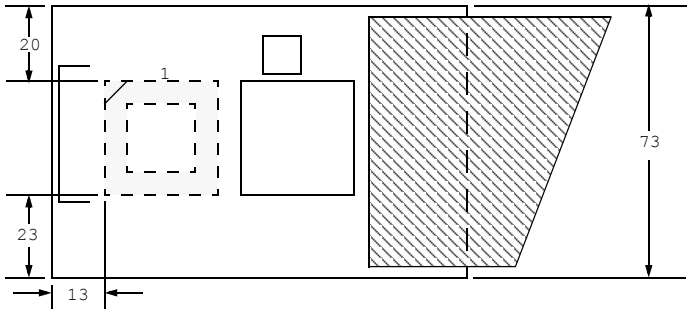
TOP VIEW (all dimension in mm)

Dimension

LA-6565 M-80535-PLCC



SIDE VIEW

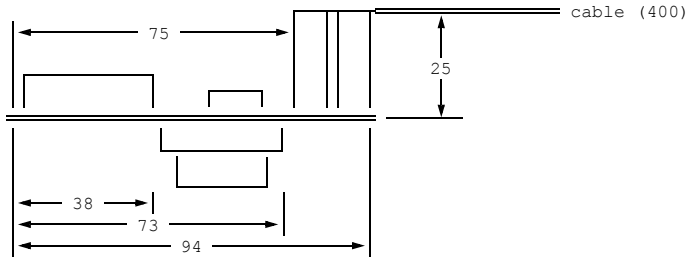


TOP VIEW

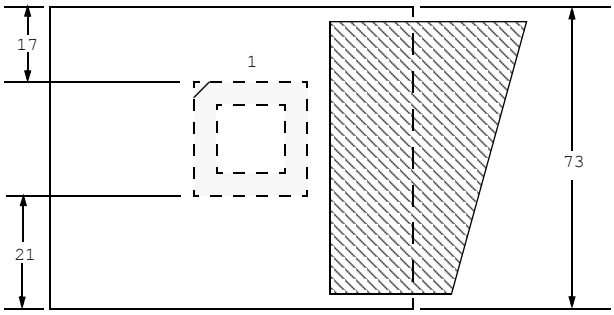
(dimensions define position of socket to target)

Dimension

LA-6566 M-80537-PLCC



SIDE VIEW

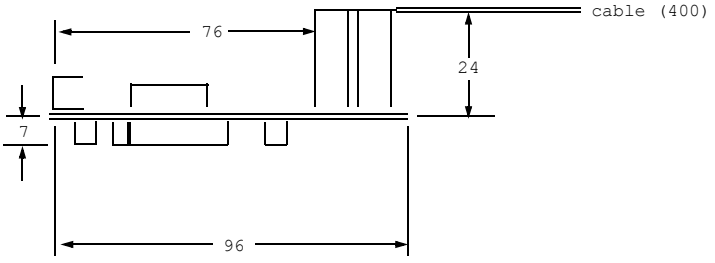


TOP VIEW

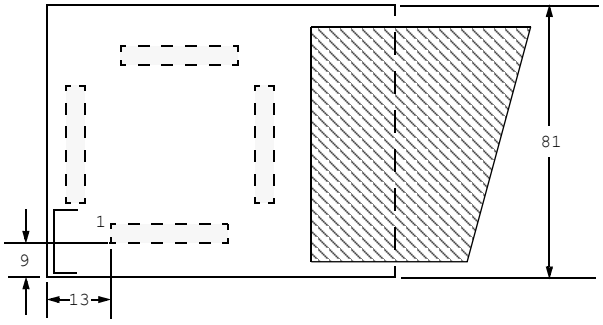
(dimensions define position of socket to target)

Dimension

LA-6578 M-C515C



SIDE VIEW

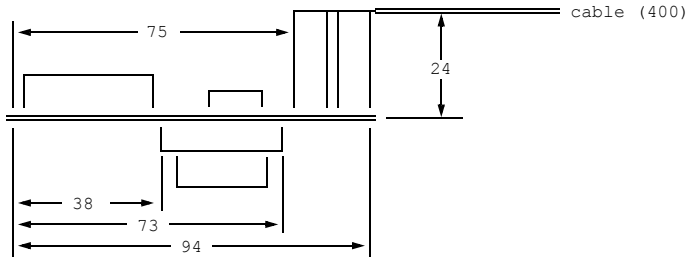


(dimensions define position of socket to target)

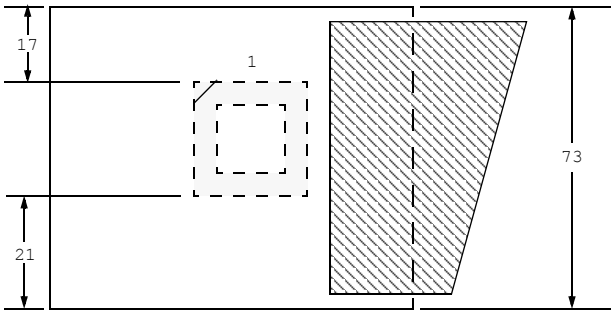
TOP VIEW

Dimension

LA-6567 M-83517A-PLCC



SIDE VIEW

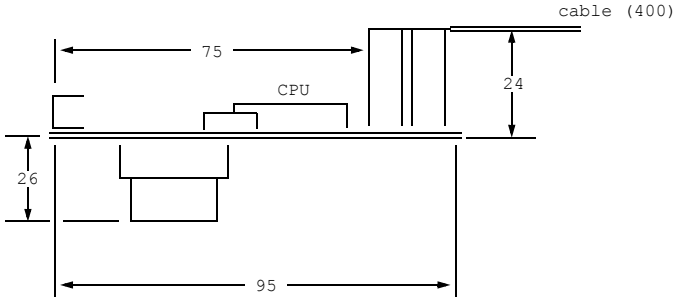


TOP VIEW

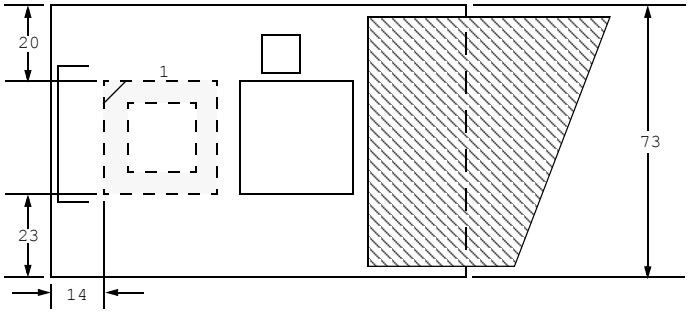
(dimensions define position of socket to target)

Dimension

LA-6568 M-83515A-PLCC



SIDE VIEW

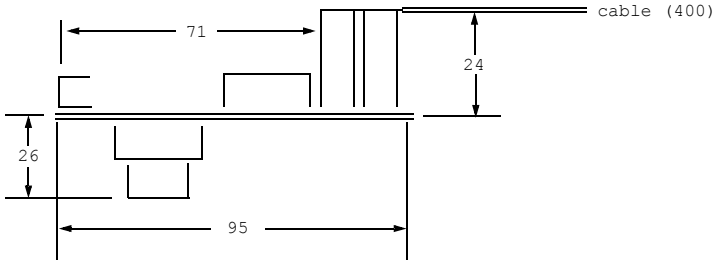


TOP VIEW

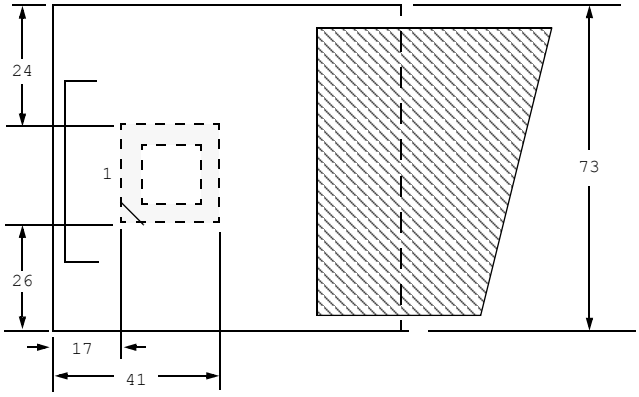
(dimensions define position of socket to target)

Dimension

LA-6575 M-C503-PLCC



SIDE VIEW

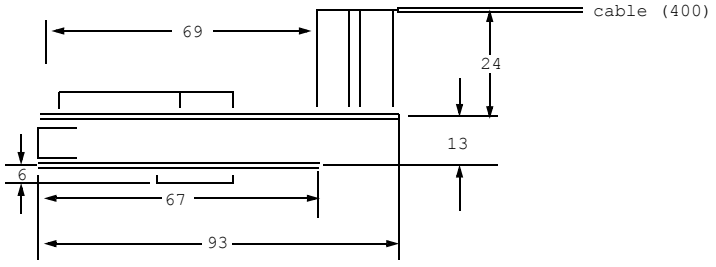


TOP VIEW

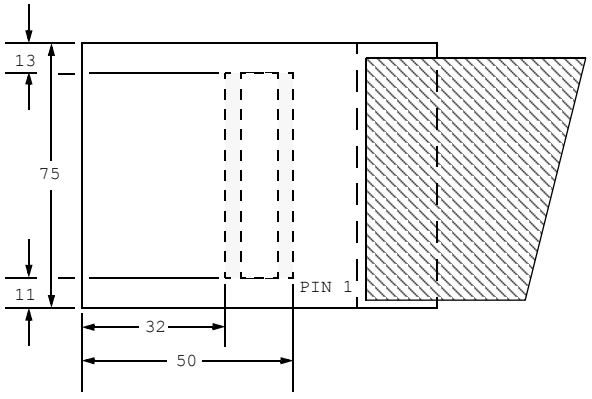
(dimensions define position of socket to target)

Dimension

LA-6580 M-85CL000



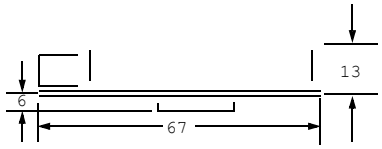
SIDE VIEW



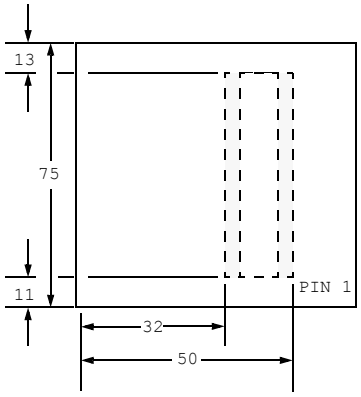
TOP VIEW

Dimension

LA-6582 A-85CL782



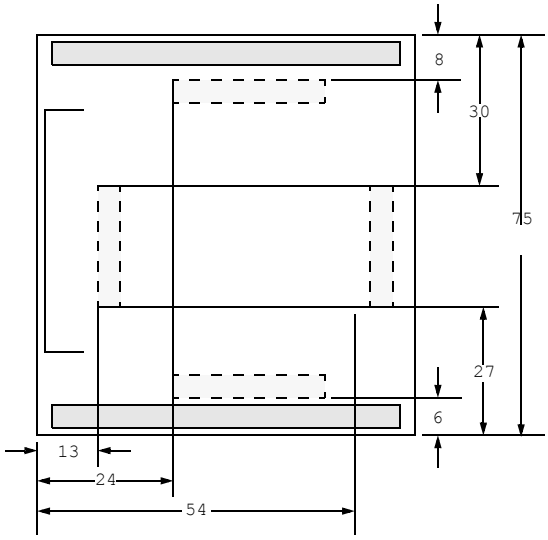
SIDE VIEW



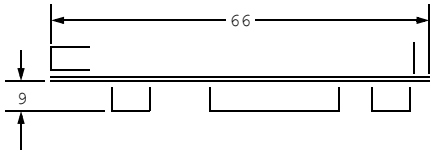
TOP VIEW (all dimension in mm)

Dimension

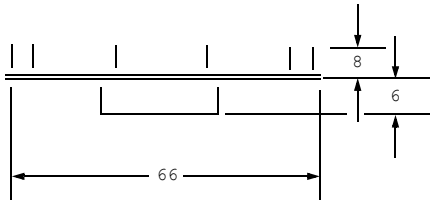
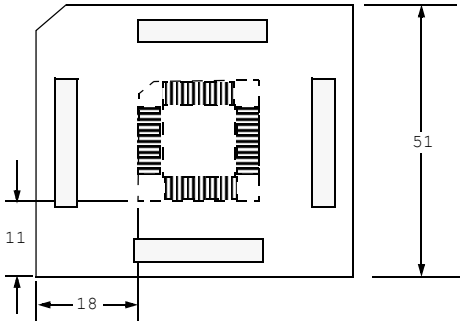
LA-6584 A-85CL580



TOP VIEW (all dimension in mm)



SIDE VIEW

Socket CPU	Adapter
ET80-QF14 C515C	**YA-1131 ET80-EYA-QF14** **Emul. Adapter for YAMAICHI socket ET080-QF14**  SIDE VIEW  TOP VIEW (all dimensions in mm)