L9000/M9000 MODULAR GANG PROGRAMMER

OPERATING INSTRUCTIONS – Version 5.41

I N D E X

L9000/M9000 PROGRAMMER ACCESSORY LIST

SOFTWARE

PC SOFTWARE:

- Provides remote control from a PC.
- Includes facility to store master eproms on disc, editing and device selection.
- Data can be downloaded from a PC using the RS232 or Centronics port.
- Software runs under Windows 2000/XP.

CONTRACT UPDATE SERVICE (AVAILABLE IN UK ONLY):

- Two software updates issued at six monthly intervals (only available from Lloyd Research).

LEADS:

- **CENT 1** connects L9000 to a Centronics printer
- **CENT PC** connects L9000 to a PC
- **SER AT** connects L9000 to PC AT with 9 pin plug

MODULES FOR E(E)PROM/FLASH

E(E)PROM/FLASH	***MODULE***	***PACKAGE***	***FEATURE***
8 pin serial -	PL308/4	DIL x 4	
24Cxx or 93Cxx	PL308/4S*	DIL x 4	
	PL309/4S*	SOIC x 4	
8 pin serial 93Cxx	PL306/4	In circuit x 4	
8 pin serial 24Cxx	PL310/4	In circuit x 4	
24/28/32 pin -	PL300/2	DIL x 2	
2716 to 27C080	PL300/4	DIL x 4	
	PL300/4S*	DIL x 4	
28 pin multiplexed add/data - 27C1028	PL600/4	DIL x 4	
32 pin to 27512	PL328/2*	PLCC x 2	Auto eject socket
	PL328/4*	PLCC x 4	Auto eject socket
32 pin from 27C010 to 27C080	PL232/4*	PLCC x 4	Auto eject socket 3V & 5V devices
32 pin from 27C010 to 27C080	PL332/2*	PLCC x 2	Auto eject socket
	PL332/4*	PLCC x 4	Auto eject socket
32 pin from 27C010 to 27C080	PL450/2	LCC x 2	Auto eject socket**
32 pin std. Jedec 1 – 8Mbit	PL333/4	TSOP x 4	
40 pin - 28F/29Fxxx - 8bit	PL335/4	TSOP x 4	
48 pin - 28F/29Fxxx - 8bit	PL336/4	TSOP x 4	
40 pin Jedec - 27C1024, etc.	PL400/2	DIL x 2	
	PL400/4	DIL x 4	
40 pin Intel 28F00X	PL334/4	TSOP x 4	
40/42 pin Mask compatible	PL420/4	DIL x 4	
44 pin Jedec - 27C1024, etc.	PL445/2	PLCC x 2	Clamshell socket
44 pin Jedec - 27C1024, etc.	PL444/2	PLCC x 2	Auto eject socket**
	PL444/4*	PLCC x 4	Auto eject socket
48 pin - 29F200 to 29F800	PL490/2Mk1	TSOP x 2	5V devices ++
48 pin - as Mk1 + 29LV200 to 29LV800 & 28F400	PL490/2Mk2	TSOP x 2	3V & 5V devices
48 pin - as Mk2 + 29WB/T800 + 29LV160 & 28F800B3	PL490/4Mk3*	TSOP x 4	3V & 5V devices
44 pin - 29F200 to 800, 29LV200 to 800 & 28F400	PL491/2Mk2*	SOIC x 2	3V & 5V devices
56 pin Intel 28F016SA	PL493/4	TSOP x 4	
56 pin - 84VD2108	PL501/4	TSOP x 4	3V & 5V devices
77 pin Memory boards	PL990/1		
71 pin CBGA AT49BV1611	PL993/4	CBGA x 4	
66 pin Puma eproms	PL996/1	PGA x 1	

PLEASE NOTE: PLxxx/1 = 1 socket module PLxxx/2 = 2 socket module
PLxxx/3 = 3 socket module PLxxx/4 = 4 socket module
* ZIF sockets are fitted in receptacles so they can be easily changed.
++ Available until stocks are exhausted
** Obsolete

MODULES FOR MICROCONTROLLERS

MANUFACTURER	*MODULE*	*PACKAGE*	*FAMILY*
ATMEL	PL227/4	ICP x 4	'In-circuit' AVRs
	PL229/4	DIL x 4	AVR90S2313, etc
	PL230/4	SOIC x 4	AVR90S2313, etc
	PL231/2	TQFP x 2	AVR90S8535, etc
	PL231/2Mk2	TQFP x 2	AVR90S8535 & MEGA163
	PL231/2Mk3	TQFP x 2	AVR90S8535 & MEGA163/16/8535/644P
	PL233/2	VQFN x 2	XMEGA32D4, etc.
	PL266/4	DIL x 4	AVR TINY26
	PL870/4	DIL x 4	89C1051/2051
FREESCALE/ MOTOROLA	PL700/4	PLCC x 4	68HC705C8/9(A)
	PL701/4	DIL x 4	68HC705C8/9(A)
	PL707/4	DIL x 4	8HC705JJ7/JP7
	PL708/4	SOIC x 4	68HC705JJ7/JP7
	PL715/2	SDIP x 2	68HC705B5/B16/B32/X32
	PL716/2	PLCC x 2	68HC705B5/B16/B32/X32
	PL717/2	QFP x 2	68HC705B5/B16/B32/X32
	PL732/2	QFP x 2	68HC705F32
	PL71L/2	PLCC x 2	68HC711L6
	PL720/4	SOIC x 4	68HC705P6(A)/P9 68HC805P18
	PL721/4	DIL x 4	68HC705P6(A)/P9 68HC805P18
	PL722/4	SDIP x 4	68HC705P6(A)/P9 68HC805P18
	PL723/4	SSOP x 4	68HC705P6(A)/P9 68HC805P18
	PL71D/4	PLCC x 4	68HC711D3
	PL72D/4	DIL x 4	68HC711D3
	PL71E/2Mk2	PLCC x 2	68HC711E9/EA9/E20/E32
	PL71K/2	PLCC x 2	68HC711KA4
	PL740/4	SOIC x 4	68HC705KJ1
	PL760/4	QFP x 4	68HC908AS60A/AZ60/AZ60A/AB32/AZ32A
	PL761/4	LQFP x 4	68HC908GR8/GR4
	PL762/4	SOIC x 4	68H(R)C908JK3(E)/JK8
	PL763/4	SOIC x 4	68H(R)C908JL3(E)/JL8
	PL764/4	DIL x 4	68H(R)C908JL3(E)/JL8
	PL836/2	QFP x 2	68HC708XL36
HOLTEK	PL480/4	SOIC x 4	HT48R06A
INFINEON	PL505/2	QFP x 2	C505CA
	PL506/2	QFP x 2	C505L
INTEL	PL874/3	DIL x 3	8742/8/9H
	PL875/2	DIL x 2	8751/2) including
	PL875/4	DIL x 4	8751/2) FX
	PL876/2	PLCC x 2	8751/2) parts

	PL876/4	PLCC x 4	8751/2)
	PL878/2	PLCC x 2	87C51GB
MANUFACTURER	***MODULE***	***PACKAGE***	***FAMILY***
MICROCHIP	PL650/1	DIL x 1	PICs 16C5X/71/84, 17C42/4
	PL651/4	DIL x 4	Most 40, 28 & 8 pin PICs
	PL652/4	SOIC x 4	Most 28 pin PICs
	PL653/4	SOIC x 4	8 pin PICs, e.g. 12C508/9
	PL654/2	PLCC x 2	44 pin PICs 16C64/5 16C74/5
	PL655/4	SOIC x 4	18 pin 16C6/7xx PICs
	PL656/2	PLCC x 2	68 pin PICs 17C75x
	PL657/4	DIL x 4	18 pin 16C6/7xx & 8 pin PICs
	PL658/2	PLCC x 2	44 pin PICs 17C4x
	PL659/4	SOIC x 4	14 pin PICs 16C505, 16F630/676
	PL660/4	SOIC x 4	20 pin 16C6/7xx PICs
	PL661/4	SOIC x 4	8 pin PICs SN narrow package
	PL661/4Mk2	SOIC x 4	As above + gang 12F629/675
	PL662/2	TQFP x 2	44 pin PICs 16Cxx 16Fxx
	PL663/2	TQFP x 2	64 pin PICs 16C9xx
	PL664/4	MQFP x 4	44 pin PICs 16Cxx 16Fxx
	PL665/4	ICP x 4	In-circuit PICs
	PL666/2	PLCC x 2	84 pin PICs 18C858
	PL667/4	SSOP x 4	Most 28 pin PICs
	PL668/2	SOIC x 2	18 & 28 pin PICs 16C5X
	PL671/4	DIL x 4	Most 40, 28 & 8 pin PICs including 12Fxx
	PL679/4	DIL x 4	14 pin 16F630 & 8 pin 10F2xx
NATIONAL	PL850/4	DIL x 4	COP8SAB/C720/28/40
	PL851/4	SOIC x 4	SO version of above
	PL855/4	SOIC x 4	COP87L20CJ/RJ & L84
NEC	PL014/4	QFP x 4	78P014GC
	PL018/2	QFP x 2	75P3018AGC
	PL054/2	QFP x 2	78P054GC/058GC
	PL064/2	QFP x 2	78P064GC/0308GC
	PL065/2	QFP x 2	78P064GF/0308GF
	PL083/2	QFP x 2	78P083GB
	PL116/2	QFP x 2	75P008/16
	PL117/2	SDIP x 2	75P008/16
	PL218/4	SOP x 4	17P218GT
	PL311/2	QFP x 2	75P3116GC
	PL312/2	PLCC x 2	78P312AL
	PL316/2	QFP x 2	75P316(A)GF
	PL368/2	QFP x 2	78P368GF
	PL430/4	SSOP x 4	75P4308
PHILIPS/ SIGNETICS	PL552/2	PLCC x 2	87C552
	PL592/2	PLCC x 2	87C592
	PL751/4	DIL x 4	87C750/1/2

MANUFACTURER	***MODULE***	***PACKAGE***	***FAMILY***
RENESAS/HITACHI	PL213/2	QFP x 2	H8/64F2134 Flash
	PL239/2	QFP x 2	H8/64F2398 Flash
	PL262/2	QFP x 2	H8/64F2623 & 64F2626 Flash
	PL264/2	QFP x 2	H8/64F2648 & Flash
	PL304/2	QFP x 2	H8/6473042/8 & 64F3048
	PL324/2	PLCC x 2	H8/6473256/7/8
	PL325/4	SDIP x 4	H8/6473256/7/8
	(***Note:*** *PL325 must be used with software version 2.DE or higher.*)		
	PL326/4	QFP x 4	H8/6473256/7/8
	PL327/4	QFP x 4	H8/6473294/7
	PL329/4	SDIP x 4	H8/6473294/7
	PL330/2	QFP x 2	H8/6473308 & 6473378
	PL331/2	PLCC x 2	H8/6473308 & 6473378
	PL364/4	QFP x 4	H8/6473644 OTP
	PL365/4	SDIP x 4	H8/6473644 OTP
	PL366/4	QFP x 4	H8/64F3644 Flash
	PL367/4	QFP x 4	H8/64F3664/94 Flash FP-64A
	PL369/2	QFP x 2	H8/64F3664/94 Flash FP-64E
	PL370/2	QFP x 2	H8/64F3687 Flash FP-64E
	PL380/2	FP80A x 2	H8/64738024
	PL381/2	FP80A x 2	H8/64F38124/024
	PL382/2	FP80A x 2	H8/64F38327
	PL383/2	FP100A x 2	H8/3834 & 3837
	PL384/2	FP100B x 2	H8/3834 & 3837
	PL432/2	QFP x 2	4074329
	PL532/2	PLCC x 2	H8/6475328
	PL597/2	PLCC x 2	H8/643334/7 & 64F3334/7
RENESAS/ MITSUBISHI	PL374/2	QFP x 2	37451EA
	PL375/2	QFP x 2	38223E4 38227EC
	PL376/2	QFP x 2	30624FGA/M 306NAFG & R5F3640
	PL500/4S*	TSOP x 4	M6MFB/T16S2T
ST/SGS THOMSON	PL620/4	DIL x 4	ST62T/E 10/15/20/25/28 & 30
	PL621/4	SOIC x 4	ST62T/E 10/15/20/25/28 & 30
	PL622/4	SOIC x 4	ST62T60B/60C/53C/63C
	PL623/4	SOIC x 4	ST62T55B/65B/55C/65C
	PL625/4	DIL x 4	ST62T00C/01C/55B/65B/55C/65C
	PL626/4	DIL x 4	ST62T52C/53C/60C/62C/63C
	PL630/4	SOIC x 4	ST7FLITE0x
	PL631/4	SOIC x 4	ST7FLITE1x/2x/3x
SIEMENS	PL505/2	QFP x 2	C505CA
	PL506/2	QFP x 2	C505L
SIGNETICS/	PL552/2	PLCC x 2	87C552

PHILIPS	PL592/2	PLCC x 2	87C592
	PL751/4	DIL x 4	87C750/1/2
MANUFACTURER	***MODULE***	***PACKAGE***	***FAMILY***
SONY	PL992/4	SDIP x 4	CXP750010S
TEXAS INST.	PL320/4	DIL x 4	TMS320E
TOSHIBA	PL201/4	DIL x 4	47P201/2
	PL242/4	SDIP x 4	47P242VN
	PL808/4	SOIC x 4	87P808M**
	PL818/4	SOIC x 4	87P808M
	PL838/4	SDIP x 4	TMP87PS38
	PL845/2	QFP x 2	87PH47
	PL846/4	SDIP x 2	87PH46
	PL847/2	QFP x 2	47P847
	PL848/2	QFP x 2	87PS64F
	PL849/4	SDIP x 4	87PM40AN
	PL912/2	QFP x 2	91W12F
ZILOG	PL860/4	DIL x 4	Z86EO2/3/4/6/8
	PL861/4	SOIC x 4	Z86EO2/3/4/6/8
	PL863/4	PLCC x 4	Z86E30/3/4 Z86733
	PL864/4	DIL x 4	Z86E30/3/4 Z86733
	PL865/4	SOIC x 4	Z86E30/3/4 Z86733

PLEASE NOTE: PLxxx/1 = 1 socket module PLxxx/2 = 2 socket module
PLxxx/3 = 3 socket module PLxxx/4 = 4 socket module
* ZIF sockets are fitted in receptacles so they can be easily changed.
** Obsolete

MODULES FOR EPROM CARDS

EPROM CARDS	***MODULE***	***PACKAGE***
ECS4 format	PL950/4	Card x 4
MIPS 'A' format	PL900/4	Card x 4

1. INTRODUCTION

Important please note: *The M9000 uses the same electronics as the L9000 model but it is housed in a re-styled case with several ergonomic improvements such as a larger display showing counts of passed and failed devices. Unless otherwise stated all features/facilities are available on both models and, hence, only the M9000 will be referred to in this manual.*

Please also note that software updates for the L9000 cannot be used on the M9000 and vice versa.

1. General

The M9000 is a versatile Gang and Set Eprom Programmer capable of programming virtually all current single rail eproms, flash memory and microcontrollers up to 32M bits. New devices can be catered for by changing the programmer's firmware which takes about five minutes. Updates are produced by **LLOYD RESEARCH LIMITED** most months and are available from both Lloyd Research Limited and its distributors.

One or two socket modules can be fitted to cater for different requirements, e.g. a PLCC module could be fitted at the same time as a DIL module. A Device Search facility to identify the required socket module for any device in any supported package can be found on the internet at

www.lloyd-research.com

This programmer has been designed for use in R&D, Production and Product Support environments. Separate operating systems are used to prompt the user. In general, the production mode is a subset of the R&D mode.

All programming operations are performed from RAM. Master data can be loaded into RAM through one or more master devices or through the serial or parallel ports.

As the M9000 programmer features a separate data bus for each socket, it is possible to program a set of eight different memories with either identical data (gang programming) or different data (set programming).

Internal RAM size can be expanded from 2M bits to 32M bits.

Typical modules have two or four green and red LEDs. The red LEDs indicate that power has been applied to the module and it is not, therefore, advisable to fit or remove devices. The green LEDs are status lights for each socket. A permanently illuminated light indicates that a device has passed a test, i.e. program, blank check, etc. A flashing light indicates that a device has failed a test. Lights remain on or continue flashing until the next function has been started.

The controls are arranged in three separate groups on the top panel:-

1. **BASIC CONTROLS:**
   This group of 10 buttons is mostly used for simple copying from the RAM to the copy sockets. It also includes facilities to set the device type and the communications parameters.

2. **RAM CONTROLS:**
   This group of 5 buttons (above the first group) contains the main controls for reading data into memory and manipulating it. Subsequently, data can be transmitted to another computer system in a variety of formats.

3. **EDITING CONTROLS & HEX KEY PAD:**
   This group of 16 controls is used for hexadecimal editing. Some of these keys have a dual function for the more sophisticated editing functions.

As groups 2 and 3 (and part of group 1) can be locked out, there is little danger of an unskilled operator damaging components by mistake.

The M9000 can program a set of 2, 4 or 8 devices simultaneously. If the data word length is 8 bits, consecutive blocks of data can be programmed into successive devices. If the word length is 16 bits, even bytes can be programmed into one device and odd bytes into another, etc.

For the editing operations, the master data must be read into memory through a port or the master socket.

New devices and facilities can be added by updating the programmer's eprom. This procedure is described in Section 13.

A small internal battery enables the programmer to 'remember' the last device type programmed and such details as the RS232 line parameters. This saves setting up the instrument every time power is connected.

Section 3 of this manual describes the outline purpose of each control and Section 4 describes the detailed operation.

CAUTION: Whilst every precaution has been taken against accidentally damaging devices, damage may occur if the following precautions are not taken:

NEVER LEAVE DEVICES IN THE COPY SOCKETS WHEN THE INSTRUMENT IS TURNED ON OR OFF.

NEVER INSERT OR REMOVE DEVICES WHEN THE RED WARNING 'LIVE' LIGHT IS ON.

NEVER TURN THE PROGRAMMER OFF UNLESS IT IS IN THE INITIAL STATE - Such action may cause the programmer to 'forget' the parameters and passwords which are normally stored when power is disconnected. The programmer can be returned to the initial state by pressing *CANCEL*.

2. INSTALLATION

2.1. Supply Voltage Adjustment & Fuse Replacement

The programmer is supplied in two versions for use on different power supplies.

M9000 - 1 - Nominal power supply 110V or 120V, 50/60 Hz.
Fused at 2A. It is important to use anti-surge fuses.

M9000 - 2 - Nominal power supply 220V or 240V, 50/60 Hz.
Fused at 1A. It is important to use anti-surge fuses.

Voltages should not vary by more than +8%/-8%. Power consumption is about 50VA.

The supply voltage setting of each instrument is printed on the fuse section of the mains plug which is fitted at the back of the instrument. The appropriate voltage is selected by withdrawing the fuseholder and inserting it with the correct voltage opposite the arrow. ***Note that*** the fuse must be fitted opposite the chosen mains voltage, e.g. if the programmer is being used on 220V and 240V, two fuses must be fitted. The correct fuse rating must always be used for replacement.

The programmer should be connected to a 'clean' supply, free from high frequency noise and electrical transients which can, for example, be caused by motors starting and stopping.

The programmer is supplied with a mains lead which should be plugged into the rear mains IEC socket.

The mains lead wires are connected as follows:

Brown	-	Live
Blue	-	Neutral
Green/Yellow	-	Earth

NEVER CONNECT THIS INSTRUMENT TO THE WRONG SUPPLY VOLTAGE.

2.2. How To Switch On

The instrument may be turned on by the mains switch fitted to the rear IEC mains plug.

After a few seconds, the sign on message:

LLOYD RESEARCH LTD
M9000 VH.SS RM Ram

Where H.SS is the software version and RM is the RAM size expressed in M bits.

2.3. The *INITIAL* State

Frequently these instructions refer to the *INITIAL* state. This is the state into which the programmer returns at the end of each function regardless of what is shown on the display. The *INITIAL* state is automatically entered into whenever the *CANCEL* button is pressed.

2.4. Installing Or Changing Modules

One or two modules can be fitted to the M9000. Almost any module can be installed in either position with any other module. However, it is not possible to program two different types of device at the same time! Modules should not be fitted or removed when the power is on. It is important ***NOT*** to leave devices in copy sockets for one module whilst using the other module. A blank module is available.

To install a module, the module should be placed on the M9000 so that its plug fits the corresponding socket on the programmer. The module should be gently pressed into position until resistance is felt. A screwdriver should then be used to lock the module to the base unit. ***Note that*** there are two lock screws on each module. ***DO NOT OVER-TIGHTEN!***

In order to take a module out, unscrew the lock screws and gently remove the module while being careful to keep the module horizontal. Do ***NOT*** lift up the front of the module and leave the rear in position as you may break the connector.

3. THE CONTROLS

The controls are arranged in three separate groups on the top panel. Each group has a specific function as described in Section 1.

Most of the basic functions can be used regardless of the state of the password. RAM and editing functions are protected by a password. The password is graded into levels and only comes into operation when the level is exceeded (*see Section 15.1.*).

Note that the M9000 keyboard is buffered. This enables the experienced user to feed in a few extra commands while the programmer is still executing an existing command. For example, it may be required to download a file and start programming. Normally, the user would have to wait until all data has been downloaded before pressing '*Program from ram*'. However, with the buffered keyboard, the user can simply press '*Program from ram*', *0* for RAM start 0, and *ACCEPT*. After downloading, the M9000 will then pick up these commands and start programming.

3.1. Basic Controls

These ten control buttons are located on the right-hand side of the top panel.

SET TYPE
This sets the device type - 27256, 27010, 16F876, etc. (*See Section 4.1. for details.*)

PROGRAM FROM RAM
This programs copy devices from RAM. The sequence followed is blank check or illegal bit test, program and verify, and then second verify. The copy socket light will be illuminated after successful verification whereas a flashing light indicates a faulty device.

VERIFY WITH RAM
The copy devices are verified with RAM. After successful verification, the socket light is illuminated. A flashing light indicates that a device has not been successfully verified.

BLANK/ERASE
This sequence checks that all bits are set to FF or 00, if appropriate. In the case of erasable devices, the user can utilise the *STEP* keys to choose *BLANK* or *ERASE* functions.

CHECK MASTER
This function checks a single device by reading it twice at high and low Vcc. The Intelligent Identifier is read and the manufacturer and device codes are displayed. Data is ***NOT*** read into RAM.

SET COMMS
The RS232 line parameters and input/output port can be specified.

STEP >
This button advances the function currently being selected, e.g. if the current memory and address is xxxx, pressing *STEP* >, selects address xxxx + 1.

STEP <
STEP < selects xxxx - 1. For certain functions, *STEP* is also used to answer questions where > corresponds to YES and < to NO.

CANCEL
This stops current operation and may be pressed safely at any time to end any sequence. This button is continuously monitored by the programmer. The programmer is automatically put into the *INITIAL* state.

ACCEPT
This button starts a sequence or accepts data.

3.2. RAM Controls

These five controls are located above the basic controls on the top panel.

READ MASTER
This reads the master device(s) into RAM.

PROGRAM FROM PORT
This function enables devices to be programmed from one of the ports while storing data in RAM.

VERIFY WITH PORT
This function compares data being downloaded with data in RAM.

DOWNLOAD
This function is used to read data into the memory through the RS232C or Centronics interfaces. The RS232 line parameters, such as baud rate, must have been previously selected by using *SET COMMS* (*see Basic Controls*) when using the RS232 port.

UPLOAD
This function outputs a selected portion of the memory contents to the RS232C or Centronics interface. The RS232 line parameters must have been selected by using *SET COMMS* (*see Section 3.1.*) when using the RS232 port.

3.3. Editing Controls & Hex Keypad

These are a group of sixteen buttons arranged in a square at the left-hand side of the instrument. They may be used for altering memory contents using hexadecimal notation. When used for special editing functions, a particular number selects a certain function, e.g. the number 6 enables the user to compute the checksum between two RAM addresses.

0 Keyboard Lock:
If the current password is known, the keyboard can be locked. The password and/or the password level can be changed.

1 Edits the contents of the current memory address using hexadecimal notation.

2 Fills memory between any two addresses to a specific byte - typically 00 or FF.

3 Merges memory blocks of data.

4 Splits memory into two blocks of data.

5 Finds a string of up to 8 bytes long.

6 Calculates the two byte (4 characters) checksum between any two addresses. Carries generated by addition are ignored.

7 Copies data block from any specified address to another part of memory. This operation is not allowed if the new memory block is not in the memory range.

8 Changes the language used for prompts and error messages. Choose between English, French or German.

9 Prints contents of memory through the RS232C or Centronics interfaces.

A Complements (1s complement) memory between any two addresses.

B Special Functions:
This facility provides an entry to occasionally used functions and new enhancements.

C Change User:
The programmer can store parameters, such as device type and line parameters, for up to five users. The parameters last used by a particular user are recorded when a new user is selected.

D Hex Calculate:
This facility performs hexadecimal addition or subtraction and is useful for calculating hexadecimal offsets when editing.

E Change mode:
This function alternates between Production and R&D mode.

F Keyboard Unlock:
If the current password is known, the keyboard can be locked. The password and/or the password level can be changed.

4. DETAILED OPERATING PROCEDURE FOR BASIC CONTROLS

4.1. Set Device Type, Set Details & Byte Order For 40 Pin Eproms

This function enables the user to specify the device type, number of devices per set and organisation - 8, 16 or 32 bit word.

After pressing *SET TYPE*, the display flashes with the message:

'*Enter device Number or <STEP> for next group*'

EITHER

Enter the required device number, e.g. 27C1001, at which time the M9000 will search for the first device and show the manufacturer. If *SET TYPE* is pressed again, the next 27C1001 (if there is one) will be displayed. ***Note that*** the full device number does not have to be entered, e.g. by entering 28F, the first device beginning 28F will be displayed. If the required device is selected but the manufacturer is wrong, use the < *STEP* > to step to the correct device. Also ***note that***, if a device number includes letters other than A to F, such as H, this letter should be left out. For example, in order to select 68HC711, simply enter 68C711.

OR

Press < *STEP* > to move to the next group of devices. If, for example, the current group is 27256, press *STEP* > to move to 27512 group or press < *STEP* to move back to the 27128 group. ***Note that,*** by pressing *STEP* >, the M9000 will display the first member of each group and, conversely, by pressing < *STEP*, the M9000 will display the last member of each group.

Having selected the required group, press *ACCEPT* to confirm the group is correct, then < *STEP* > to select the required device type followed by *ACCEPT*. The number of ICs in the set can then be specified followed by the organisation.

ICs per set	***Organisation (8 bit eproms)***	***Organisation (16 bit eproms)***
1	8 bits	16 bits
2	16 bits	32 bits
4	32 bits	32 bits
8	32 bits	32 bits

If, for example, 3 ICs per set is chosen, the setting is rounded up to 4.

The set size must be consistent with the RAM size. Therefore, an 8M bit RAM would be required for a set of four 2M bit eproms. If the set size was too large, the M9000 would not program or verify, etc., and an error message would be displayed. If, however, the eprom was a 40 pin device, a further question would be asked to find out in which order the two eight bit bytes should be programmed. For Intel systems, the byte order is usually even bytes in D0 - D7 with odd bytes in D8 - D15. Motorola systems, however, are usually the opposite way round!

Note that the device list is sequenced in three segments: memories, microcontrollers and finally eprom cards.

A device may be selected without one of the correct modules being fitted. However, the correct module ***MUST*** be fitted before reading, programming or blank checking, etc. If the module has not been fitted, the operation will be aborted with a request made to fit the appropriate module. In some cases, there may be a choice of modules, e.g. DIL or PLCC devices, etc.

The device list (*see Section 15.2.*) specifies which devices have an Intelligent Identifier code. This code can be checked by the M9000. For these devices, an extra question is asked to determine whether the user wants to check this code. ***Note that*** this choice is stored for each user on power down. Essentially, there are three choices:

1. to ignore the code,
2. to check that the device and manufacturer code are both correct, and
3. to allow the M9000 to program a mix of devices providing that the devices are compatible.

Whenever a new device has been chosen, the check is reset to the default condition. When the M9000 is supplied, the default is to check the identifier but this can be changed. In order to turn off the check, use Special Function 6, i.e. press button B followed by button 6.

The Intelligent Identifier is a two part code which specifies the device manufacturer and type. Checking it helps to avoid the possibility of damaging eproms if the eprom selected and the eprom fitted should differ. (*Please refer to the Application Note in Section 16.2. for a full explanation.*)

If a microcontroller is selected such as Intel 87C51 or a Microchip PIC16, it is possible to program one or more of the lock bits. ***Note that*** the default is ***NOT*** to program any lock bits.

With the PL332 MKII module first produced in December 1994, it is possible to protect sectors on devices such as AMD 29F040 by selecting the device type 29F040_SP. With the PL335, PL336, PL490 and PL491 modules, it is possible to protect sectors on devices such as Fujitsu 29F400TA or BA by selecting the device type 29F400TA_SP or 29F400BA_SP.

When the user is asked to specify a mask, the least significant bit refers to the first sector, etc. For example, by specifying a mask of 000003, the first two sectors would be protected. In addition, the *STEP* > key sets all the bits in the mask for the current device and the *STEP* < clears the mask. Erasing such a device removes all sector protection.

4.2. Program From RAM

1. This mode may only be selected from the *INITIAL* state. If necessary, change device type and set mode as described in Section 4.1. In R&D mode, the user is asked the RAM start address. In Production mode, the address is assumed to be 0. The M9000 then works out whether there is enough RAM considering the set size. If there is not, the function will be aborted with a message:

   '*Insufficient ram for specified set size*'

2. The copy devices ***MUST*** be blank in Production mode. However, in R&D mode, the device need not be blank but ***MUST*** be programmable. A test for programmability – '*Illegal bit test*'- is performed if the device is not blank unless the device uses flash technology. In this case, 'non blank' devices are automatically erased in R&D mode but ***NOT*** in Production mode. After completion of the pre-program checks which are performed at just over 4.75V for 5V parts, all devices are programmed simultaneously. If a device fails to program, the M9000 continues with the remaining devices.

3. The program sequence follows the pre-program checks and the device(s) is(are) then verified with Vcc set to just under 5.25V. In Production mode, a second verify at low Vcc is performed and, as most eprom vendors only recommend a single verify, this procedure is extremely thorough. (*See the next section for details of the verify procedure*.) Lock bits of microcontrollers such as the 87C51 and PICs are programmed after ***ALL*** devices have verified successfully. (*Refer to the Application Note in Section 16.4. for more details on programming lock bits*.) ***Note that*** some devices which can be 'in circuit' programmed are verified once at nominal Vcc. Flash Microchip PICs are an example of this.

4. During these procedures, the % done is indicated on the display.

5. As a precaution against a user turning off the programmer and forgetting that the RAM data will be lost, the M9000 will not allow any device(s) to be programmed from RAM unless one of the following actions has been taken after turning on the programmer:

   Data must have been read into RAM from master device(s),
   OR
   Data must have been downloaded through one of the ports,
   OR
   Data must have been edited using the edit function.

REMEMBER THAT RAM DATA WILL BE LOST IF THE PROGRAMMER IS TURNED OFF.

As a further precaution against programming incorrect data into devices, the M9000 will not allow any device(s) to be programmed from RAM if any of the following *RAM Edit* functions are cancelled before the function has been completed:-

Copy, Complement, Fill, Merge or Split.

4.3. Verify with RAM

1. This mode may only be selected from the *INITIAL* state. If necessary, change device type and set mode as described in Section 4.1. In R&D mode, the user is asked the RAM start address. In Production mode, the address is assumed to be 0. The M9000 then works out whether there is enough RAM considering the set size. If not, the function will be aborted with a message:

   '*Insufficient ram for specified set size*'

2. The M9000 stops verifying if it finds a discrepancy between RAM and copy data in R&D mode. In such cases, the address, master data and copy data will be displayed and the green light will flash above the relevant sockets. If *ACCEPT* is then pressed, the M9000 will continue to verify the other devices. If *STEP* > is pressed, it will also verify the failed devices. (In Production mode and for most microcontrollers, the details of failed devices are passed over.)

3. In Production mode, a two pass verify at both high and low Vcc is performed for most devices. Some parts, such as flash Microchip PICs, are verified once at nominal Vcc. If the device manufacturer specifies that the device works at 5V + or - 10%, the device is checked at these limits. The device list in Section 15.2. shows which devices have a 10% limit. It is possible to turn off the second verify using a Special Function (*see Section 6.11*).

4. In R&D mode, a single verify is performed at the high Vcc limit unless the device manufacturer specifies a two pass verify in which case a two pass verify is performed.

5. After verifying a set of devices, the total checksum is displayed. However, individual eprom checksums can be checked by using the *STEP* keys. The relevant eprom(s) is (are) indicated by the appropriate copy socket light(s). ***Note that*** if, for example, two sets of devices are programmed, both copy sockets lights will be lit for identical eproms.

6. If the user has requested the programming of lock bits for microcontrollers, the program run ends with the message '*Locked @ xxxx*' where xxxx is the checksum. (*For more details on programming security bits, please refer to the Application Note in Section 16.4*.)

4.4. Set Communications Parameters

1. This process may only be started from the *INITIAL* state. It enables the following parameters to be seen and/or changed. These parameters are automatically retained until they are next changed even if the power supply is disconnected. Confirmation that the parameters have been retained is given by the typical sign on message:

If, for some reason, the parameters have not been retained, the second line of the sign on message would be:

'*W)Default parameters*'

The following parameters may be set:

Data input through?	RS232 or Centronics parallel ports
Data output to?	RS232 or Centronics ports
Baud rate?	19K2, 9600, 4800, 2400, 1200, 600
Data bits?	7 or 8
Parity?	Odd, even or none
Stop bits?	1 or 2
RS232 handshake?	Soft - Xon/Xoff or Hard - DTR

CAUTION 1: Do ***NOT*** select the Centronics port for input unless the remote computer has been turned on because the M9000 will continually analyse the signals on the Centronics port and the keyboard may become inoperative.

CAUTION 2: If the Centronics port has been selected for input and output, it will not be possible to turn the M9000 into remote control from a remote computer.

4.5. Blank Check & Erase Flash Eproms

1. This mode may only be selected from the *INITIAL* state. If necessary, change device type as described in Section 4.1. ***Note that*** this function cannot be used for Texas 27C292.

2. If any device has been programmed, the address and data will be displayed and the appropriate copy socket light will flash. In order to carry on blank checking the remaining blank devices, press the *ACCEPT* button. Alternatively, press *STEP* > to carry on blank checking all devices. In Production mode, the programmer **ONLY** reports the presence of programmed devices.

3. At the end of the procedure, the copy socket lights will be illuminated for blank devices whereas the lights will flash for failed devices.

   Note that, in the case of 16 bit devices, the second line of the LCD can only show four separate devices because each device has two bytes of data. The actual faulty device is shown by the flashing light.

4. In the case of flash memory, the user has the choice of either performing the standard blank check or erasing the device by using the *STEP* keys. In the latter case, it is also blank checked. If it is required to program the device immediately after the blank

check, the user can simply press the *PROGRAM FROM RAM* button when the programmer is in R&D mode. The M9000 will then automatically erase the device (if necessary) before programming.

4.6. Check Master

This facility provides a simple method of checking a single master without having to read the data into RAM. The Intelligent Identifier is first checked if the device is specified in the device list as having an Identifier and if the user has requested this check during the *SET TYPE* function. If the device is read-compatible with the selected device, it is read twice at high and low Vcc. If the checksum is the same both times, it will be displayed. If the checksum is different, a message – '*Faulty device*'- will be displayed. If the device is blank, the word - '*blank*'- will be displayed. If the checks are successful, the copy socket light will be permanently illuminated. If an error is detected, the light will flash, etc. If the Intelligent Identifier code is inconsistent with the eprom being checked, an error message – '*Wrong type*'- will be displayed instead of the checksum. If the eprom contains no Intelligent Identifier code and a check has been requested, a message – '*No code*' - will be displayed. (***Note that*** *Section 16.2. contains an Application Note on the Intelligent Identifier.*)

4.7. Program From Master

On the M9000 programmer, there is no master socket. Data from a master device is read into RAM and then blank devices are programmed from RAM.

1. Select the correct device type as described in Section 4.1. ***Note that*** for gang programming, '*Devices per set*' is 1. (*Please refer to Section 7 for programming sets of devices.*)

2. Fit the required socket module. In most cases, it does not matter if a different module is fitted in the other socket module position. If it does matter, the LCD will display – '*remove PLXXX module*'.

3. Set the operating mode to either '*Production*' or '*R&D*'. Production mode is the mode to use for most production applications. R&D mode is quicker and more flexible. (*Please refer to Section 4.3. for more details.*)

4. Place a master device in any socket and press button '*READ*' to read the data from the master and store it in RAM. ***Note that***, in R&D mode, the programmer will ask for the RAM Start Address which should normally be 0. After the device has been read, the checksum is displayed. ***Note that*** it is possible to read a device in a PLCC package for example and then to program devices in, for example, a DIL package.

5. The required number of blank devices should now be fitted. In order to start programming, press the *PROGRAM* button. In Production mode, the programmer checks that all parts are blank, programs the parts and then verifies them at both high and low Vcc. In R&D mode, the programmer performs an '*Illegal bit check*' in order to ensure that each bit can be programmed before programming and then verifies only at high Vcc.

   In Production mode, the programmer indicates failed parts after all parts have been verified whereas, in R&D mode, the programmer stops at all verification failures and shows the RAM, device data and the address.

6. After programming, a green LED is illuminated above each device if it has verified successfully. A flashing LED indicates a failure.

   Counts of devices programmed and failures are shown on the LCD. Counts are reset at power on. (*Please refer to Section 6.11., Special Function12*.)

NOTES:

The following facilities are also available:-

a) The minimum number of devices to be programmed can be specified. This is very useful when programming DIL parts as it protects against the operator forgetting to close the socket latch. (*Please refer Section 6.11., Special Function 14*.)

b) The ability to verify at low VCC can be turned off for flash devices. (*Please refer to Section 6.11., Special Function 15*.)

c) The blank check can be turned off for flash devices. (*Please refer to Section 6.11., Special Function15*.)

The red LEDs on each module indicate that power is being applied. Do ***NOT*** fit devices, remove devices or stick on labels when red LEDs are illuminated. Power is automatically removed after programming/verification. To remove power in an emergency, press the *CANCEL* button.

5. DETAILED OPERATING PROCEDURE FOR RAM CONTROLS

5.1. Program From Port

1. This mode may only be selected from the *INITIAL* state. If necessary, change device type and set mode as described in Section 4.1.

2. This function enables the user to program one or more devices at the same time as downloading data to RAM. This procedure is ***NOT*** recommended for most applications.

3. There are a few restrictions which apply to this procedure which do not apply to '*Program from ram*', e.g.:

    - Devices must be blank.
    - Certain algorithms (some AMD, Atmel and Texas) are not suitable for this method.
    - FLASH algorithms using the Status Register are not suitable for this method.
    - This method assumes that the RAM Start Address is zero for the program and the verify sequence. It can, however, be specified as non-zero for the download.

4. The procedure starts in a similar way to the *DOWNLOAD* function and continues in a similar manner to the *Program from RAM* function. As there is no need for data to be in sequence, the M9000 cannot show the programming % completed. Instead, however, the download address is shown as for the *DOWNLOAD* function. After downloading and programming, the data is verified with RAM.

5. The checksum is shown after programming but ***note that*** the programming checksum will only be the same as the download checksum if all data bytes have been downloaded. Bytes not downloaded are set to the blank condition (FF for eproms).

6. As part of the procedure, the RAM is filled with FF and devices are blank checked before programming. However, the sending computer can start to send data immediately after the *ACCEPT* button has been pressed for the RAM Start Address. Any data received during the RAM fill and blank check is stored. If the M9000 data buffer fills up, the handshake (Xon/Xoff or DTR Lo for serial data) will be invoked.

CAUTION: If for any reason the data transmission is halted, the procedure should be stopped by pressing *CANCEL*. Under no circumstances should the devices be removed when the red 'live' light is on. Removing devices when programming voltages are present can lead to device destruction.

5.2. Read Master Device(s) Into RAM

1. This mode may only be selected from the *INITIAL* state. If necessary, change device type and set mode as described in Section 4.1. In R&D mode, the user is asked the RAM start address whereas, in Production mode, the address is assumed to be 0. The M9000 then works out whether there is sufficient RAM considering the set size. If there is not, the function will be aborted with the message -

   '*Insufficient ram for specified set size*'

   In this case, the set size must be reduced or more RAM must be fitted.

2. Data is then read into RAM starting at the RAM Start Address. If the M9000 is in set programming mode (2 or more ICs per set) and if one or more devices are missing from the set, this part of RAM will unchanged. ***Note that*** eproms must only be fitted in the copy sockets corresponding to a logical set. (S*ee Section 7.2. for a full explanation*). Data is stored in RAM according to the data structure of bits per word. In 8 bit mode, data will be stored in consecutive RAM addresses. In 16 bit mode, data will be stored in alternate locations. In 32 bit mode, data will be stored in every fourth byte.

3. Progress is indicated in % during reading.

4. After devices have been read, the checksum will be displayed and the relevant copy socket lights will be illuminated.

5. After reading a set of devices into RAM, it can be useful to check the individual checksums of each eprom. This can be done by using the *STEP* keys. The display shows the checksum for each device. The relevant eprom is indicated by the copy socket light.

5.3. Verify With Port

1. This mode can only be selected from the *INITIAL* state. If necessary, change device type and set mode as described in Section 4.1.

2. This procedure enables the user to compare a file of data on a computer to RAM data which may have been read into RAM from one or more devices or from another computer file.

3. While data is being read into RAM, any differences are displayed on the LCD for a few seconds.

4. An appropriate error message indicating whether or not the data blocks are identical is displayed at the end.

5.4. Download Data To The M9000

1. This facility allows data to be downloaded from another computer system such as a PC or mainframe. Windows compatible PC software is available for this purpose.

2. Set up comms. parameters and port to be used as described in Section 4.4., using the *SET COMMS* function. For speed, the parallel port is recommended. This is about five to six times faster than when using the serial port of a typical PC running at 9,600 baud. ***Note that*** downloading Intel Hex or Motorola files using our PC software is even faster as data compression is used.

3. Press *DOWNLOAD*. The programmer then displays the current data format for a possible change by using one of the *STEP* keys. Alternatives are Intel, Motorola S, Tektronix, Ascii Hex Space or Binary formats. Extended address records are catered for.

4. The Load Data from Address is the next item to be requested. This is the address of the first byte to be read from the data being downloaded.

5. The Load Data to Address is then requested. This is the address of the last byte to be read from the data being downloaded. It must be higher than the Load Data from Address. The difference between these addresses must be less than or equal to the RAM size. If the address is not valid, the questions will be repeated. ***Note that*** this address can be calculated automatically according to the current set size and the Load Data from Address by pressing *STEP* >.

6. The '*RAM start*' is prompted. This is the address where the first byte will be stored in the programmer RAM. It is checked as follows:

   '*Ram start* < *or* = *'Load data to'* – *'Load data from*'

   This condition ensures that data will not overflow because the RAM is too small.

7. The M9000 now displays a message to show which port is expecting data.

8. Once data transmission has started, the address currently being downloaded will be displayed.

9. After downloading all data, the checksum of the downloaded data will be displayed. However, the RAM checksum could be different.

10. Defaults for downloading are the same as those last used for uploading or downloading.

5.5. Upload Data From M9000

1. This facility allows data to be uploaded to another computer system.

2. Set up comms. parameters and port to be used as described in Section 4.5., using the *SET COMMS* function.

3. Press *UP LOAD*. The programmer displays the current data format for a possible change by using one of the *STEP* keys. The main formats are Intel, Motorola S, Tektronix, Ascii Hex Space or Binary formats. Extended address records are catered for.

   The M9000 then requests a terminator byte to transmit. This byte is transmitted after all data has been uploaded. Typically, this byte is set to some convenient value such as 1A (Control Z) so that the software of the receiving device can recognize the end of file. If no terminator is required, the recommended setting is 00, corresponding to a null byte. This question is omitted for binary formats.

4. The Load Data from Address is next requested. This is the file address of the first byte to be transmitted from the M9000.

5. The Load Data to Address is then requested. This is the address of the last byte to be transmitted from the M9000. It must be higher than the Load Data from Address. The difference between these addresses must be less than or equal to the RAM size. If the address is not valid, the questions will be repeated. ***Note that*** this address can be calculated automatically (according to the current set size and the Load Data from Address) by pressing *STEP* >.

6. The '*RAM start*' is requested. This is the address where the first byte will be taken from in the programmer RAM. It is checked as follows:

   '*Ram start' < or = 'Load data to' - 'Load data from*'

   This condition ensures that data will not overflow because the RAM is too small.

7. The M9000 now displays a message to show which port is being used.

8. Once data transmission has started, the address currently being uploaded will be displayed.

9. After uploading all data, the checksum will be displayed.

10. Defaults for uploading are the same as those last used for uploading or downloading.

6. DETAILED OPERATING PROCEDURE FOR HEXADECIMAL/EDITING KEYS

6.1. Edit RAM

1. This mode may only be selected from the *INITIAL* state. It is used to change individual bytes of RAM.

2. After pressing Hex Key 1, the Edit Start Address is requested. The address must be a valid RAM address.

3. The M9000 then displays the Hex data and the Ascii symbol for that value. Data can be changed by entering a new Hex value. ***Note that***, if it is required to change a byte from possibly 'F6' to '03', then either '03' or '3' can be entered. The leading '0' is automatically inserted. The *STEP* keys will step to the next address.

4. The function can be terminated by pressing *ACCEPT*.

6.2. Fill RAM To Predetermined Value

1. This mode may only be selected from the *INITIAL* state. It is used to set a selected range of RAM to a specified value.

2. After pressing Hex Key 2, the RAM Fill Start Address is requested. The address must be a valid RAM address.

3. Next, the RAM Fill End Address is requested. The end address must be higher than the start address and must also be a valid RAM address.

4. The RAM fill byte is then requested.

5. The address being filled is shown every few seconds.

6. At the end of the procedure, the limits filled are displayed together with the RAM fill byte.

7. If valid addresses have not been entered, the questions will be repeated.

6.3. Merge Data Blocks

1. This mode can only be selected from the *INITIAL* state. It is used to merge together two blocks of memory which have previously been split for a 16 bit system.

2. After pressing Hex Key 3, the programmer merges two blocks of 8 bit data into one block of 16 bit data. The even bytes should be in the lower half of RAM starting at 00000 and the odd bytes in the upper half of RAM. The upper half of RAM depends on the RAM size as follows:

L/M9000 32M RAM 00200000

L/M9000 08M RAM	00080000
L/M9000 04M RAM	00040000
L/M9000 02M RAM	00020000

3. Approximate progress of the merge is indicated by a reducing count.

4. The merged data block starts at RAM Address 0.

   Consider an example of merging two blocks of RAM:

1st Block	***2nd Block***
Byte 0 = 00	Byte 80000 = 40
Byte 1 = 01	Byte 80001 = 41
Byte 2 = 02	Byte 80002 = 42
Byte 3 = 03	Byte 80003 = 43

   The new block starts at hex address 0: 00 40 01 41 02 42 03 43

   NOTE: Data from two master eproms can be merged automatically by specifying 16 bit data when using the *SET TYPE* facility. These devices can be read into ram at RAM Start 0.

6.4. Split Data Blocks

1. This mode can only be selected from the *INITIAL* state. It is typically used to split 16 bit data into two blocks of 8 bit data. After the split, the block containing the even numbered bytes starts at 0 and the block containing the odd numbered bytes starts at the first address of the upper half of RAM.

   The upper half of RAM depends on the RAM size as follows:

L/M9000 32M RAM	00200000
L/M9000 08M RAM	00080000
L/M9000 04M RAM	00040000
L/M9000 02M RAM	00020000

2. Approximate progress of the split is indicated by a reducing count.

   Consider an example of splitting a RAM block:

Byte 0 = 00
Byte 1 = 01
Byte 2 = 02
Byte 3 = 03
Byte 4 = 04
Byte 5 = 05

After the split, there would be two blocks as follows (the example assuming a RAM size of 8M bits):

1st Block	*2nd Block*
Byte 0 = 00	Byte 80000 = 01
Byte 1 = 02	Byte 80001 = 03
Byte 2 = 04	Byte 80002 = 05

NOTE: This function is performed automatically when using the *Set Programming* facility by specifying 16 bit data. Two eproms can, for example, be programmed with 16 bit or 32 bit data by specifying the number of data bits when setting the device type.

6.5. Find Character String & Replace

1. This mode can only be entered from the *INITIAL* state. It is used to find a string of up to 8 hex bytes between specified limits in RAM. 'Don't care' bytes are allowed.

2. Press Hex Key 5. The required start address in RAM must be entered, then press *ACCEPT*. The default is the last used address.

3. The last RAM address must be entered and then press the *ACCEPT* button. The default is the last address used. The last address must be higher than the first address.

4. The first byte must now be entered. The Ascii character is shown, e.g. if Hex Byte 32 is entered, the Ascii character 2 will be displayed. If the first byte is a 'don't care' term, simply press the *STEP* > key. If this is the end of the string, press the *ACCEPT* button.

5. In order to enter a second or subsequent byte, press *STEP* > and repeat the procedure.

6. If the character string is found, the *EDIT RAM* procedure will be entered as described in Section 6.1. A search can be initiated for another occurrence of the string by pressing the *ACCEPT* button. If the string is not found, an end of function message will be displayed and the programmer will return to the *INITIAL* state.

6.6. Checksum Between Specified RAM Addresses

1. This mode can only be selected from the *INITIAL* state. It is used to compute the 2 byte (4 Hex character checksum) between two user specified addresses.

2. Press Hex Key 6. The M9000 requests the start address. Enter a valid RAM address and press the *ACCEPT* button.

3. The M9000 then asks for the Checksum End Address. This must also be a valid RAM address which is higher than the start address.

4. Press the *ACCEPT* button after entering the upper limit. The M9000 adds each byte to an accumulator and displays the resultant checksum which corresponds to the 16 bit addition of all RAM bytes with carries ignored.

5. The address currently being 'checksumed' is displayed every few seconds.

6.7. Copy Block Of RAM Data

1. This mode can only be selected from the *INITIAL* state. It may be used to re-arrange data in RAM. The data block to be moved is called the Source block. The Source block is moved to the Destination block. There are no restrictions regarding where data is moved to and the M9000 allows the Source block to overlap the Destination block.

2. Press Hex Key 7. The Source Block Start Address is requested. This must be a valid RAM address. Press the *ACCEPT* button.

3. The Source Block End Address is then requested. This must be higher than the start address and must also be a valid RAM address. Press the *ACCEPT* button.

4. The Destination Address is next requested. This must also be a valid RAM address. The M9000 works out the Destination Block End Address and checks that there is sufficient RAM. Press the *ACCEPT* button.

5. If invalid addresses have been entered, the questions will be repeated.

6. While moving data, the byte which is currently being moved is displayed every few seconds. ***Note that***, for this function, data blocks may overlap despite the address being displayed which is for information only.

7. This function ends with a message showing the first address of the Source block and Destination blocks.

6.8. Change Language

1. This facility changes the text/prompt language between English, French or German.

2. Press Hex Key 8. The programmer requests the required language. *Step* > or *Step* < to change language.

3. Press the *ACCEPT* button when the required language has been selected. All future text messages and prompts will appear in the required language.

6.9. Print RAM

1. This mode can only be selected from the *INITIAL* state. The contents of a specified part of RAM will be output to the RS232C or the Centronics port. The port and line parameters must previously have been defined using *SET COMMS*.

2. Press Hex Key 9. The M9000 requests the first address from which to print. Enter the required address which must be a valid RAM address and press *ACCEPT*.

3. The M9000 requests the last address to which to print. This must also be a valid RAM address which is a higher address than the start address.

4. The printout takes the form of a Hex dump with the equivalent Ascii character.

6.10. Complement RAM Between Specified RAM Addresses

1. This mode can only be selected from the INITIAL state. The contents of a specified part of RAM are complemented (1s complement).

2. Press Hex Key A. The programmer requests the first RAM address to complement. Enter the required address (which must be a valid RAM address) and press the *ACCEPT* button.

3. The M9000 then requests the last address to be complemented. This address must also be a valid RAM address which is higher than the start address. Press the *ACCEPT* button.

4. The M9000 displays the address which is being complemented every few seconds.

6.11. Special Functions

This mode can only be selected from the *INITIA*L state. Press Hex Key B to select *SPECIAL FUNCTIONS*. The *STEP* keys can then be used to select the required function or the relevant hex key can be pressed.

0 Press Key 0 to display the software release year and month. ***Note that*** there could be more than one release in any one month.

1 Press Hex Key 1 to display the model number (L9000 or M9000), software revision number and RAM size. This message can be read remotely.

2 Press Hex Key 2 to check hardware calibration. A warning message to remove proms is displayed. After removing any devices, press *ACCEPT*. The programmer will halt and put calibration voltages across three preset potentiometers. In order to check these voltages, the bottom of the instrument must be removed. ***Note that*** some screws also retain the feet. There are three preset controls which are identified on the track-side of the board. The voltage across the ends of the presets must be measured on a 4.5 digit DVM or similar instrument after a warming up period of ten minutes.

The voltages are:	RV1 =	25.0V +/- 0.1V
	RV2 =	6.00V +/- 0.05V
	RV3 =	6.00V +/- 0.05V

It is also recommended that data should be downloaded to the programmer from a computer whose clock frequency is known to be accurate to within 1%. If there is no hardware error for 10 bytes of data for example, it may safely be assumed that the clock frequency is correct for the M9000. This ensures that the program pulse widths will also be correct. Alternatively, the clock frequency can be checked on pins 15 and 18 of U12 counter timer as 1.843 MHz.

IN ORDER TO RETURN THE PROGRAMMER TO NORMAL OPERATION, TURN IT OFF FOR AT LEAST A MINUTE.

4 Press Hex Key 4 to display the method of control. Either '*Local*' or '*Remote*' will be displayed. <*STEP* switches to local while *STEP*> switches to remote. When remote is selected, the M9000 assumes that all remote control commands should be echoed back to the controller and that a prompt character should be sent to the controller after the completion of the current command string. In remote mode, the local keyboard will also be operative unless it has been remotely disabled. If '*Remote*' is selected, two further questions will be asked - whether the prompt feature and the echo feature are required. The default is '*yes*', press <*STEP* to cancel.

6 Press Hex Key 6 to display the Intelligent Identifier default. If this is '*yes*', the Intelligent Identifier check will be turned on whenever a new device has been selected. If this is '*no*', the check will be turned off whenever a new device has been selected. The operation of the Intelligent Identifier is explained in full in the Application Note in Section 16.2.

8 Press Hex Key 8 to display the modules fitted. If the M9000 software fitted does not recognise the module, a number will be displayed. In such cases, there is either a fault or a later version of software is required.

9 Press Hex Key 9 to enter the Hex code to action the remote control command string. Default = 0DH (carriage return)

10 Press Hex Key A to enter this function. This function enables the user to increment a number in RAM which is typically a serial number. The parameters set up by this function are: number base (Binary or Decimal), length of number and which end of the byte string has the low or high byte. When these parameters have been set up, it is possible to increment or decrement a byte string by remote control.

11 Press Hex Key B to print the M9000 device list to the currently selected output port. The output port can either be the RS232 or the Centronics port as selected using the *SET COMMS* function. The function can print the complete device list or a device list for one manufacturer or a range of manufacturers. Between one and ninety nine copies can be printed. The listing also shows the version number and the date on which the software was released. Devices are listed by manufacturer with five devices per line giving a maximum of 69 characters per line. The number of characters per line has been restricted to 69 so that laser printers can be used to print directly onto A4 paper. In order to print a full list, press *ACCEPT* when the M9000 requests the manufacturer at which to start printing and press *ACCEPT* again when the M9000 requests the manufacturer at which to stop printing. For a partial listing of, for example, Texas Instruments' devices, use the *STEP* > to scroll to this manufacturer for the start and press *ACCEPT* to set this manufacturer as the last manufacturer as well. The M9000

will then just print Texas only devices. In order to print, for example, both Texas and Toshiba devices, simply follow the above steps but set the last manufacturer to Toshiba instead.

When selecting manufacturers, the M9000 displays a code as well as the name of the manufacturer. This is the *Manufacturer Code* used as part of the Intelligent Identifier check. Some manufacturers, such as Intel, have two codes so they appear twice. Also, ***note that*** GI and Microchip have the same code because they are the same company.

The latest device list (which is the same for both the L9000 and the M9000) is available from the internet at:-

www.lloydres.co.uk/l9000/l9device.txt

12 Press Hex Key C to display the number of devices which have programmed correctly, which have failed to program and the grand total. Press <*STEP* to reset the count to zero. The totals will be reset to zero every time the programmer is turned on. The totals will only include the number of devices which have completed the programming sequence. They will not include any devices if the programming sequence has been terminated with the *CANCEL* key. (***Note that*** the M9000 displays this information as the LCD is larger.)

13 Press Hex Key D to select the slave address of serial eeproms such as the 24Cxx. This setting is used to set both hardware and software addresses when programming 24Cxx devices on the PL308 module. ***Note that***, if devices are programmed using the PL308 module, the setting is irrelevant. However, by selecting the appropriate address, it is possible to connect the M9000 to an external I²C bus and program a selected device. As it is possible to have up to eight 2K devices on an I²C bus, the possible values of this address are 0 to 7. However, ***note that***, as device size increases, the number of devices on the bus decreases, e.g. there can only be 4 x 4K (24C04) devices. The possible addresses for these devices would be 0, 2, 4 or 6.

14 Press Hex Key E to set the minimum number of devices to program at a time. This function is useful as it can detect incorrectly inserted or missing devices before the program function has commenced. It is recommended for high volumes where boredom can become a factor. If there are too few devices, programming will be aborted and the green LEDs above unused sockets will flash. The default setting is 1. Press the *ACCEPT* button to step to next question – '*Sockets in use*'. Enter the hex bit map of the sockets in use. This function is useful in conjunction with the function above to mask out faulty sockets. The default setting is FF.

15 Select function control sequence set up - *i) Blank check Flash IC's before Program?* Press Hex Key F to display the current setting. Press <*STEP* to set '*no*' or *STEP*> to set '*yes*'. If '*no*' has been selected, the program sequence will skip the blank check before program. Press *ACCEPT* in order to step to next question - *ii) Verify 2nd pass with low Vcc in Production mode?* Press Hex Key F to display the current setting. Press <*STEP* to set '*no*' or *STEP*> to set '*yes*'. If '*no*' has been selected, the production verify and program sequences will skip the second verify pass. Press the *ACCEPT* button to step to next question - *iii) Check CE & OE work:* Press Hex Key F to display the current setting. Press <*STEP* to set '*no*' or *STEP*> to set '*yes*'. The test will check the device CE and OE pins every time a device function is carried out.

When using socket adaptors for some microcontrollers, the test should be turned off as they fail the test. Simply press the *ACCEPT* button to terminate the sequence.

6.12. Change User & Parameter Storage

1. This mode can only be selected from the *INITIAL* state. The facility enables one user to store his parameters whilst another user is using the programmer. In fact, parameters may be stored for up to five separate users.

   User parameters include RS232 parameters, choice of port, device type, access speed, number of ICs per set, bits per word, data format for up/downloading, terminator for uploading, load at and load to addresses for downloading, password and password level.

2. Press Hex Key C. The current user will be displayed. A new user number - 1 to 5 - can be entered. Press the *ACCEPT* button to end.

6.13. Hexadecimal Calculations

1. This function can only be selected from the INITIAL state. This facility performs 2 byte hex addition or subtraction. The 3 byte result is displayed. The first byte pair default is the value previously used so that it can be used as a constant.

2. Press Hex Key D. The last value entered will be displayed. A new value can be entered. The arithmetic operator + or - can be changed by using one of the *STEP* keys. Press the ACCEPT button.

3. The second value may now be entered. The arithmetic operator + or - can be changed by using one of the *STEP* keys. Press the *ACCEPT* button.

4. The M0000 will display the answer.

6.14. Keyboard Password

1. This mode can only be selected from the *INITIAL* state. This facility provides a simple means of ensuring that the more complex functions can only be used by people who know the password. The password 'level' can be set by the user so that the point at which the password must be known is appropriate. In a Production environment, for example, when the programmer is being used to copy one particular device, it may be desirable to stop the device type being changed.

2. The programmer requires the user to enter a 3 character password to lock or unlock the keyboard. The user can then change the password. Whenever the password is changed, the user can reset the password to one of four levels.

3. Each function has a password level. Level 4 is the highest level and is specified for the most complex functions. If the password has been set at level 3 for example, all functions can be performed without using the password except those at level 4. The password level for each function is shown in Section 15.1.

4. The procedures for locking and unlocking the keyboard are identical. Only locking is, therefore, described. If at any time the programmer 'loses' its parameter information, the password will be reset to the default value of 000. If the password has been set and forgotten, contact the manufacturer or your distributor.

5. Press Hex Key 0 (F to unlock). Enter a 3 character password using any of the 16 hexadecimal keys.

6. Press the *ACCEPT* button after entering the password. If the entry was incorrect, the question will be repeated whereas if it was correct, the new password will be requested. The default is the existing password. Enter a new password if required.

7. Press the *ACCEPT* button after entering the new or retaining the old password. If a new password is given, a new level can be set.

8. Enter a new password level which must be in the range 1 to 4.

9. The programmer will return to the *INITIAL* state after a second.

7. SET FACILITIES

7.1. General

The term – '*set programming*' - is used in this manual to describe the ability to program more than one device simultaneously with different data. Set programming is used whenever the user specifies more than 1 IC per set. The programmer can program up to eight devices simultaneously providing that the data can be contained within the memory. The way data is programmed into the devices is dependent on the data word length. If the data has eight bits, then the first data block will be programmed into the first device and the second data block will be programmed into the second device, etc. If, however, the data has sixteen bits, the odd RAM bytes will be programmed into one device and the even into the other.

The set parameters must be entered using the *SET TYPE* facility as described in Section 4.1. If, therefore, there are four devices in a set, simply answer 4 to the question '*ICs per set? 1*' Note that entries are always rounded up to the nearest set size above the entry. If a 3 is entered, a set size of four will be used.

7.2. Which Sockets To Use

For eproms with an 8 bit data bus:

> The right-hand socket is always programmed with the least significant data byte. For 8 bit data, therefore, the right-hand device will contain the first data block and the device next to the right-hand socket will contain the next data block, etc. For 16 bit data, the right-hand device will contain the first, third and fifth bytes of data and the device next to the right-hand device will contain the second, fourth and sixth bytes. If there were say 4 ICs per set with sixteen bit data, the right-hand pair of devices will contain the lowest block of data and the left-hand pair will contain the highest block of data.

For eproms with a 16 bit data bus:

> In the case of 16 bit data and four socket modules, the two right-hand sockets will always be programmed with identical data. Similarly, the two left-hand sockets will also always contain the same data. To program two devices with 32 bit data, therefore, use one socket on the right-hand side of the module and one on the left-hand side. If two sets of eproms are required, load all four sockets. ***Note that*** the two left-hand devices will contain identical data and, similarly, the two devices on the right-hand side will also contain identical data.
>
> In order to read a pair of devices into RAM, do ***NOT*** place them on the same side of a four socket module. Instead, the device containing the lowest two bytes should be placed on the right-hand side and the device containing the highest two bytes should be placed in one of the sockets on the left-hand side.
>
> ***Note that*** all 16 bit socket modules with four sockets work in the same way.

7.3. Making Multiple Sets Of Copies

For eproms with an 8 bit data bus:

Multiple sets of devices can automatically be made by simply putting devices into spare sockets. Therefore, in order to make two sets of two devices, simply put four suitable devices into the right-hand (or left-hand) group of copy sockets. The left-hand pair of devices will contain the same data as the right-hand pair. To make four pairs, simply put another four devices into the left-hand group of sockets. ***Note that*** it is possible to program parts of sets by simply omitting devices.

Example of set programming four sets of two 32 pin devices using two PL300/4 modules:

<-PL300-> <-PL300->

2 1 2 1 2 1 2 1

Eproms with a 16 bit data bus:

Multiple sets of devices can automatically be made by putting devices into spare sockets. Therefore, in order to make two sets of two devices, simply put four devices into a PL400/4 or use two PL400/2 modules. However, ***note that***, unlike the PL300 module, the two devices on the right-hand side of the PL400/4 module will be identical and, similarly, the two devices on the left-hand side will also be identical. Programming will be found to be slightly quicker if only one socket of the left-hand or right-hand pair on the PL400/4 is used.

Example of set programming four sets of two 40 pin devices using two PL400/4 modules:

<-PL400-> <-PL400->

2 2 1 1 2 2 11

7.4. Important Points Regarding Set Programming

1. Set programming is ignored when blank checking.

2. When reading 16 bit data into RAM from a master device, data is stored in every alternate ram byte. With 32 bit data, data is stored in every fourth byte.

3. When loading RAM from a port, data is always stored according to the record format specified. The set programming details are irrelevant.

7.5. Valid Start Addresses

When programming sets of devices, it is important to appreciate that increasing the number of ICs per set increases the RAM required. Therefore, if there were four 1M bit eproms in a set, the programmer would require 4M bits of RAM. However, the organisation of an 8, 16 or 32 bit word does not alter the RAM requirement.

When starting to program, verify or read master devices, the M9000 checks that:-

RAM start + (No. of ICs per set X Device size) < or = RAM size

If there is insufficient RAM, an error message is displayed:

'*Insufficient ram for specified set size*'

The remedy is either to fit more RAM or reduce the set size.

7.6. Programming Sets Of Four Devices With Two Socket Modules

Sometimes there may be a requirement to program a set of four eproms when only two socket modules are available.

In order to copy a set of four eproms, fit two 2 socket modules such as a PL300/2 or PL450/2 and then select 8 devices per set with, for example, 8 bits per word. Read the four MASTERS into RAM. Replace the MASTERS with blank devices and program from RAM. Using this method, the user RAM must be at least twice as large as the set size.

It is also possible to program a set of four devices with 32 bit data after data has been downloaded to RAM. In order to do this, select 4 ICs per set with a 32 bit word and then program two devices from the required RAM Start Address, normally 0. The right-hand device will contain bits 0 to 7 and the left-hand device will contain bits 8 to 15. Program the second pair of devices with the RAM Start Address + 2. The right-hand device will contain bits 16 to 23 and the left-hand device will contain bits 24 to 32.

8. REMOTE CONTROL

8.1. General

For most applications, our PC software is recommended. However, some users may wish to write their own application. This section offers guidance for remote control.

Any function, except Keyboard Lock/Unlock, can be controlled remotely by using the RS232 or Centronics ports once remote control has been initiated.

A single Ascii character is allocated to each key. Receipt of this character followed by a carriage return initiates the same action as local operation. After each successful operation, program, verify, etc., the M9000 responds with a prompt '+'. The M9000 responds with a negative prompt '-'if the command cannot be understood or if it has not been successful. There are options to suppress the prompt and the local echo. Optionally, the local keyboard can be totally disabled. The carriage return character can be changed using Special Function 9. Valid characters are in the range - 01h to 29h - except for NULL, ESC, XOFF, XON, SPACE and QUOTE.

The RS232 port is recommended for remote control if the user wants to control the programmer in great detail and to monitor operations. The Centronics port is recommended for fast data transfer. It is feasible to use a mixture of both.

8.2. Entering Remote Control

The M9000 always powers up in local mode. Remote control can be entered the following ways:

1. By using the Special Function facility on the programmer (*see Section 6.11.4.*). The user is also asked whether the prompt and/or echo options are required. The default is '*yes*'.

2. By sending a '^' character to whichever port is defined as input by the *Set Comms* function on the keyboard. This character and all future commands are echoed back to the port selected for output by the *Set Comms* function. This option should only be chosen if the M9000 is connected to a controller with a bi-directional port such as RS232.

3. By sending a '%' character to whichever port is defined as being the input by the *Set Comms* function on the keyboard. The programmer is immediately switched to remote control but both the prompt and the echo are suppressed.

8.3. Disabling The Keyboard

Once the programmer is in remote operation, a special command - '*H*' - disables the local keyboard until the M9000 is powered up again or until a fresh '^' has been received. The programmer should be in the *INITIAL* state when this command is sent. ***Note that***, with software versions prior to 2.37, an '*H*' will put the M9000 into remote control. However, with versions 2.37 and beyond, the '*H*' command will ***NOT*** put the M9000 into remote mode.

8.4. Entering Remote Control Commands

Once the remote mode has been entered, the input port is continuously scanned. Any character received is monitored. Assuming that there has been no hardware error, such as a parity failure or framing error, the programmer assumes that the character is a remote control instruction and tries to obey it as though the relevant button has been pressed on the keyboard. Certain additional characters have been introduced to remove the need for a programming language in some cases. The device type can, for example, be set directly as can the upload and download formats.

8.5. Error Detection & Correction

Errors can only be reported to the user if the prompt is enabled. As already explained, remote commands can be grouped together on a line and terminated by a carriage return. When the M9000 has finished these commands, it sends a prompt character back to the controller. If no errors have occurred, the prompt will be a '+'. If an error has been found or if a command has not been understood, a '-' prompt will be sent. The controller should then send a '-' back to the M9000 whereupon the M9000 will send a two line error message back to the controller. The error buffer will then be cleared. ***Note that*** the M9000 always records the first error if there is more than one.

If an invalid command has been received, a '-' prompt will be returned to the controller unless the local echo has been turned off. The first line of the display will show '*Invalid command*' and the second will show the character. The M9000 will be returned to the I*NITIAL* state and the copy sockets will be powered down. This is very similar to pressing *CANCEL* on the keyboard. The function must, therefore, be started again.

Note that the M9000 will remain in remote mode until it has been returned to local.

8.6. Return To Local Only Operation

The M9000 can be returned to local only operation using the Special Function command which can be performed remotely.

B4[G <RETURN>

Select Special Function B. Then press Hex Key 4 and step back to '*local*'. As happens with all remote control commands, no action will be taken until a carriage return has been received.

8.7. Reading The Display From The Controller

The 'read-back' command '*/*' causes the M9000 to transmit the current message as two separate lines to the controller. Each line starts with a '*?*'.

8.8. Software Considerations

The ability to download a command file means that simple Ascii files can be used to control repetitive command sequences. These are typically found in R&D applications where it is required to download a file after compiling a program and, in Production applications, where certain pre-defined jobs need to be done.

Avoid data files which have any text after the end of file record. This text will be interpreted as remote control instructions.

Typically, remote control commands can be created as a one line file which must be terminated by a carriage return. For documentation purposes, notes can be added at the end of each line. All characters between a '*;*' character and the next carriage return are ignored. Some of these instructions have notes prefixed by ' 'as examples.

8.9. Hardware Considerations

The remote controller must be compatible with the M9000. In the case of serial devices, the RS232 hardware parameters must be the same. Under most circumstances, the highest possible baud rate should be used for fast data transfer.

Note that, if a hardware error occurs, such as a framing error, it is necessary to press *CANCEL* on the programmer.

8.10. Remote Control Commands

0 - F Hexadecimal keys

G Accept or GO

H Disables keyboard of L/M9000

I Download or Input data
After this instruction, the data transfer format can be set:

A	Ascii Hex Space
B	Pure Binary
D	Dec. Binary
E	Extended Tektronix
H	Binary with header = FF.
I	Intel
M	Motorola
N	Intel 32 bit
T	Tektronix

K Blank check

L Shows which lights are on or flashing.
Eight bytes are returned to the controller with the following meaning:-

. = Light off
F = Light flashing - test failed
P = Light on - test passed

The first character received is for the left-hand socket of the left-hand module whether or not it has been fitted. ***Note that*** this function can only be initiated from the *INITIAL* state.

O Upload or Output data.
After this instruction, the data transfer format can be set as for download.

P Program devices from RAM

R Reads master device into RAM

S Set comms
The port is chosen by the next instruction:

[RS232 port
] Centronics

T Set type
This instruction can be followed by the *STEP* command or a special method of setting the device type. *T"27C256 Tex"* sets the device type to 27C256 Texas. The quote signs around the device type and make are essential. ***Note that*** this instruction sets the set size to 1.

V Verifies the copy sockets with RAM

Y Verifies the port with RAM.

Z Programs devices from port

* Cancel
Subsequent commands will be obeyed.

[Step <

] Step >

> Increment serial number

< Decrement serial number

/ Reads back both lines of current message in display

- Reads back first error message and then clears error buffer

ESC The current command is terminated and all other commands (1Bhex) are abandoned

Space Command ignored

8.11. Incrementing/Decrementing Serial Numbers In RAM

Providing that the serial number parameters have been set up using the Special Function A, it is possible to increment a decimal or binary serial number. It is important to note that the serial number parameters must be set up as described in Section 6.11.10., each time the programmer is powered up. The serial number is edited using the *EDIT* function and by sending a '<' or a '>' character to the M9000. ***Note that***, if the serial number is defined as being a decimal number, it ***must*** be a decimal number otherwise an error message will be generated.

9. THE SERIAL RS232C PORT

9.1. General

The programmer is fitted with an RS232C serial port. The port socket is a 25 way D connector located at the rear of the instrument. The socket is wired as a DTE interface which means that it can be connected directly to another computer with a DCE interface. If it is required to connect the programmer to another piece of terminal equipment such as a printer, it will almost certainly be necessary to cross over connections - 2 and 3, 4 and 5, 20 and 6 - which is shown below.

9.2. Connection Details

PIN NO.	*PURPOSE*	*TITLE*	
2	Output	Transmit data	This line carries the data sent from the programmer to the external computer.
3	Input	Receive data	This line carries the data sent to the programmer to the external computer.
4	Output	Request to - Send	This line is always hi because in virtually all instances data can be received at 19000 Baud. In most cases, it can be left dis-connected unless the host requires this line to be driven hi.
5	Input	Clear to Send	This line must be hi to enable the programmer to transmit.
6	Input	Data Set Ready	This line must be hi to enable the programmer to transmit.
7	Common ground		
20	Output	Data Terminal - Ready	This line is sent hi except when downloading data with a hard handshake. It is then used for flow control.

9.3. Connection To Another Computer (DCE Interface)

PIN	*PROGRAMMER FUNCTION*	*DIRECTION*	*COMPUTER PIN*	*'AT' PIN*	*'XT' PIN*	*COMPUTER FUNCTION*
2	Transmit data	>	2	2	3	Receive data
3	Receive data	<	3	3	2	Transmit data
5	Clear to send	<	5			Request to send
6	Data set ready	<	6	4	20	Data terminal ready
7	GROUND		7	5	7	GROUND
20	Data terminal ready	>	20	6&8	6&5	Data set ready

If a hardware handshake is used, the following programmer pins ***MUST*** be wired - 2, 3, 6 and 7.

If a software handshake is used, the following pins ***MUST*** be wired - 2, 3 and 7.

Note that the computer may, however, require all handshake lines to be connected.

9.4. Connection To A Printer, Etc. (DTE Interface)

PROGRAMMER			*PRINTER, etc.*	
PIN	*FUNCTION*	*DIRECTION*	*PIN*	*FUNCTION*
2	Transmit data	>	3	Receive data
5	Clear to send	<	4	Request to send
6	Data set ready	<	20	Data terminal ready
7	GROUND		7	GROUND
20	Data terminal ready	>	6	Data set ready

If a hardware handshake is used, the following programmer pins of the programmer ***MUST*** be wired - 2, 3, 6 and 7.

If a software handshake is used, the following pins of the programmer ***MUST*** be wired - 2, 3 and 7. Leave other pins open circuit.

10. DATA FORMATS

10.1. General

Data can be transferred to and from the programmer in several data formats, namely:

Intel
Motorola S
Tektronix
Ascii Hex Space
Binary

Since there are slight variants in the use of these formats, this section defines the implementation of this product.

A useful feature of this programmer is the ability to transmit a terminator byte after uploading data. This enables general purpose software to be used by the remote computer. If the terminator is not required, the terminator should be set to '*00*' corresponding to a null byte. The default value of the terminator is the value last used for the format in question. The terminator is omitted for Binary formats.

With effect from version 2.15, the M9000 will accept lower case Ascii characters - '*a*', '*b*', '*c*', '*d*', '*e*' or '*f*' - when downloading data. The M9000 always transmits data with upper case letters.

Another useful feature of this product is the ability to upload or download part of a data block. The questions asked prior to data transfer - '*Load data from?*' and '*Load data to?*' - refer to the target address of the data and ***NOT*** to the location in the programmer. The first location from which to take data ***MUST*** be entered in response to the prompt - '*Ram start?*'

10.2. Intel Format

There are two Intel formats supported by the M9000. The first is '*Intel Hex'* which supports addresses up to 20 bits long and the second is '*Intel 32 bit*' which supports addresses up to 32 bits long. However, for simplicity, either format will download an Intel file with record types 00, 01, 02, 03 or 04. However when uploading using *Intel 32 bit*, a type 04 record is generated when the address is above FFFF. With *Intel Hex* format, a type 02 record is generated.

There are four types of record recognised by this programmer:-

Type 00	Data record
Type 01	End of file record
Type 02	Extended segment address record (for 20 bit addresses)
Type 04	Extended linear address record (for 32 bit addresses)

Each data record contains the record type, length, data load address and checksum as well as up to 255 bytes of data. The format is specified as follows:

:NNAAAATTDD.....DDCC where

:	is the first byte of the record.
NN	is a two Ascii hexadecimal representation of the number of data bytes. The maximum value is 255 bytes (0FFH). ***Note that*** 00 counts as zero bytes of data.
AAAA	is a four Ascii hexadecimal representation of the data load address of the first byte of data. Successive bytes are stored in successive locations. During loading, this address is added to the segment base address to calculate the load address which is then displayed.
TT	is a two Ascii hexadecimal representation of the record type: 00 = Data 01 = End of file 02 = Address extension 04 = Address extension for address bits A16 to A32
DD..DD	For type 00 data records, this is a two Ascii hexadecimal representation of each byte of data. This field is not present in end of file records - type 01. For 04, Extended Linear Address record DDDD represents address bits - A16 to A32.
CC	is a two Ascii hexadecimal representation of the negative checksum of the record. Starting with the record length and ending with the checksum, the hexadecimal sum (taken two at a time) modulo 256 is 0 (2s complement).

Data loading stops after reading a type 01 record.

Each record is normally separated by a carriage return/linefeed.

A '*Checksum error*' is generated if a checksum error is found in any record.

10.3. Motorola Format

There are six types of record recognised by this programmer:

Type S1 data record with up to 32 data bytes	Address field 2 bytes
Type S2 data record with up to 32 data bytes	Address field 3 bytes
Type S3 data record with up to 32 data bytes	Address field 4 bytes
Type S7 end of file record	Address field is 4 bytes
Type S8 end of file record	Address field is 3 bytes
Type S9 end of file record	Address field is 2 bytes

Each data record contains the record type, length, data load address and checksum. The format is specified as follows:

STNNAA..AADD..DDCC where

S	is the first Ascii character in the record.
T	is the second Ascii character in the record representing the record type: 1, 2 or 3 = Data record 7, 8 or 9 = End of file record
NN	is a two Ascii hexadecimal representation of the record length. The record length includes the start address, data and checksum fields. The maximum number of data bytes is 250 bytes (0FFH). The maximum length is, therefore, 255 bytes inclusive of the start address and the checksum.
AA..AA	is a four, six or eight Ascii hexadecimal representation of the data load address of the first byte of data. Successive bytes are stored in successive locations. During loading, this address is displayed.
DD..DD	is a two Ascii hexadecimal representation of each byte of data. This field is not present in end of file (types 7, 8 and 9 records).
CC	is a two Ascii hexadecimal representation of the 1s complement of the eight lower order digits obtained after summing the record length, address and data bytes. Hence, the least significant byte of the sum of the record length, address, data bytes ***AND*** checksum will be FFH.

Data loading stops after reading a type 7, type 8 or type 9 record.

Record types not described above are ignored.

Each record may be separated by a carriage return/linefeed.

A 'Checksum error' is generated if a checksum error is found in any record.

10.4. Binary Formats

There are three types of binary format recognised by this programmer:-

1. Binary (header = FF)
2. Binary (no header)
3. Dec. Binary

1. Binary (header = FF)

The format is specified as follows:-

HBBBB.....BB where

H	is the start of data header with binary value 11111111

B is the Binary data

2. Binary (no header)

The format is specified as follows:-

BBBB.....BB where

B is the Binary data

3. Dec. Binary

The format is specified as follows:

HH..HHLBBB.....BB where

H is the start of data header with a string of binary values 11111111

L is the end of header with binary value 00000000

B is the Binary data

As the Binary formats carry no address information, the first byte is assumed to be for address zero. The next byte is assumed to be byte 1, etc. Binary formats can, therefore, only be used to transmit continuous blocks of data. There are also no checksums so it is recommended that parity is enabled (odd or even) when using the serial port.

The M9000 assumes that data transmission has been completed when the last byte of data has been read or when data transmission has stopped for at least five seconds. In the latter case, the address field in the display will remain unchanged for at least 5 seconds. The checksum will then be displayed and the buzzer will sound as a warning that an error might have occurred.

10.5. Ascii Hex Space Format

There are three types of record recognised by this programmer.

Start of data marker is byte 01H or 02H or both.

Data records consist of a two Ascii hexadecimal representation of the byte. Each pair of Ascii characters is separated by a space (20H).

End of data marker is indicated by byte 03H.

All bytes prior to the start of the data marker character are ignored ***AND*** all bytes after the end of the data marker are also ignored.

Carriage returns (0DH) and/or linefeeds (0AH) are ignored, if present. If either is present, a further space separator is not required but it may be present.

The upload record format is:

*
11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF where
%

*	is a non printable character corresponding to byte 01H or 02H
11 22 33	Each data byte is represented by a two Ascii hexadecimal byte pair. Each byte pair must be separated by a space (20H). For example, hexadecimal byte 2A is coded as 32H 41H.
%	is a non printable character corresponding to byte 03H

This format has no provision for transmitting address information. Uploading or downloading, therefore, starts at the first address and ends at the last address.

When uploading data, the format is as shown in the above example.

10.6. Tektronix Hex Format

This format can only transmit data within the address range - 0 to FFFF. For larger address ranges, use Extended Tektronix Hexadecimal format. The M9000 checks that the '*Load data from*' and '*Load data to*' addresses are not greater than FFFF. Unlike Intel format, Extended and normal Tektronix Hex formats are ***NOT*** compatible.

Tektronix Hex has only one type of record with the following format:

/AAAANNCCDD..DDCS where

/	Indicates start of record
AAAA	Address of first byte of data
NN	Record length
CC	A two digit hexadecimal number which is the sum, modulo 256, of AAAA and NN
DD	Data - each data byte is represented by two Ascii characters
CS	Data character checksum, modulo 256

End of file is indicated by records with a record length of 0.

Because of the limited address range, valid '*Load data from*' and '*Load data to*' addresses must be in the range - 0 to FFFF.

10.7. Extended Tektronix Hex Format

Unlike the normal Tektronix Hex format, this format can load data over an enormous address range from 0 to FFFFFFFFFFFFFFFF. However, as with most programmers, the M9000 only examines data over the address range - 0 to FFFFFFFF.

There are three types of record:

Data	= 6
End of file	= 8
Symbol block	= 3 (Ignored)

The data record format is:

%NNTCSLA.....ADD where

%	Extended Tektronix Hex record
NN	The number of Ascii characters in the record excluding the %
T	Record type = 6
CS	Two digit checksum of all characters excluding the % and the checksum, modulo 256
L	Address field length - 0 indicates 16 Ascii characters
A..A	Address field with variable length from 1 to 16 characters
DD..DD	Data - each data byte is represented by two Ascii characters

The end of file record is similar but the record type is 8.

The address field is the program starting address.

There is no data.

11. PARALLEL INPUT/OUTPUT (CENTRONICS) PORT

Data can be output from or input through the Centronics port at the rear of the M9000. The programmer has a 36 way standard Centronics connector. As there are no supply voltages present, this port can be connected to any 'standard' Centronics printer with a ribbon cable. Pin 1 at one end of the cable is connected to pin 1 at the other end, etc. It is, however, advisable to check that there is no conflict between the programmer and your printer ***BEFORE*** connecting the two products.

Pin numbers carry the signals shown below. ***Note that*** the connector shows the pin numbers in very small numbers.

The signal direction below is shown when the M9000 is transmitting data. Data direction is reversed when the M9000 is receiving data.

PIN	*FUNCTION*	*DIRECTION wrt L9000*	*COMMENT*
1	Data Strobe	Output	Implemented
2	Data bit 1	Output	Implemented
3	Data bit 2	Output	Implemented
4	Data bit 3	Output	Implemented
5	Data bit 4	Output	Implemented
6	Data bit 5	Output	Implemented
7	Data bit 6	Output	Implemented
8	Data bit 7	Output	Implemented
10	Acknowledge	Input	Implemented
11	Busy	Input	Implemented
12	Paper empty	Input	Not used in transmit mode. Low in receive mode.
13	On-line	Input	Pulled hi through 1K resistor in transmit mode; not monitored otherwise.
14	Not used		No connection
15	Not used		No connection
16	Ground		Implemented
17	Ground		Implemented
18	+5V		No connection
19-30	Ground		Implemented
31	Initialise		No connection
32	Fault	Output	Pulled hi through 1K resistor to +5V; otherwise not used.
33	Ground		Implemented
34	Not used		No connection
35	Not used		No connection
36	Demand		No connection

12. SYSTEM MESSAGES

12.1. Error Messages

A few error messages lock up the programmer to avoid confusion. These are typically used when there is a danger that another message may follow so quickly that the original message may have been missed.

Such messages are prefixed with '*E)*'. These cause the current function to be aborted. The only way to return to normal operation is by pressing *CANCEL*. Messages without this prefix do not lock up the M9000.

E)Buffer o'flow

The programmer has not been able to process data being downloaded fast enough and a buffer has overflowed. Use a lower baud rate. This error is most unlikely to happen at any baud rate, even 19,200.

Data pin o/c

The device indicated by the flashing light has a faulty data bit which appears to be disconnected. Occasionally, this fault can be caused by a device which does not meet Ioh = -400 microamps. This fault can also be caused by fitting a 24 pin device in the top of a copy socket instead of a 28 pin device.

E)Non-Ascii char

While reading a record, the programmer has found a byte which is not in the range - 0-9 or A-F. As this may have been a transmission error, it is worth trying again. Most MDS formats encode Hex data as two Ascii bytes. If a non-Ascii byte is therefore found, the programmer knows that an error has occurred.

E)RS232-Framing

While receiving data through the RS232 port, a framing error has occurred because a valid stop bit has not been detected. Check that the baud rate, number of data and stop bits, and parity are the same for both the programmer and the remote computer.

E)RS232-Overrun

While receiving data through the RS232C port, an extra character has been received before the programmer has been able to process the last character sent. A character has, therefore, been lost and, hence, the function has been aborted. This should never happen unless data has been sent before the programmer is ready to receive it. Check that the baud rate, number of data and stop bits, and parity are the same for both the programmer and the remote computer.

E)RS232-Parity

Whilst receiving data through the RS232C port, a byte has been read with an incorrect parity bit. Check that the baud rate, number of data and stop bits, and parity are the same for both the programmer and the remote computer.

12.2. Warning Messages

W)Default paras

This message may be displayed when the power is turned on. It indicates that the battery backed up memory used to retain the parameters has been corrupted. The programmer will, therefore, assume default parameters.

12.3. System Failure Messages

S)Sys.EPROM fail

This message may be displayed when the power is first turned on. It indicates that there is a fault in the system EPROM. ***DO NOT USE THE INSTRUMENT.***

S)User RAM fail

This message may be displayed when the power is first turned on. It indicates that there is a fault in the user RAM. ***DO NOT USE THE INSTRUMENT.*** After firmware version 2.32, the user RAM is also checked when the programmer is in the *INITIAL* state. If this message is displayed, the user will not be able to program further devices until more data has been loaded into RAM. However the user would be unwise to use the programmer until the reason for the failure is found.

S)Prom Vcc fail

This message means that the Vcc voltage for the devices being programmed has fallen out of limits. As the likely cause is a faulty eprom, try another. If this fault happens consistently, check the hardware calibration as described in Section 6.11.2. If it still fails, there is probably a hardware fault in the programmer.

S)Prom Vpp fail

This message means that the Vpp voltage for the devices being programmed has fallen out of limits. As the likely cause is a faulty eprom, try another. If this fault happens consistently, check the hardware calibration as described in Section 6.11.2. If it still fails, there is probably a hardware fault in the programmer.

12.4. Information Messages

Various messages are displayed when this programmer is operated. Most messages are self-explanatory and are not mentioned in this section. This section only includes messages which may require further explanation.

Reading port - Data out of range

This message can be displayed when downloading data. It is displayed because the address of the data being downloaded is not between the '*Load data from'* and the '*Load data to*' addresses specified before the download. (*Please refer to Section 10 for an explanation of data formats*.)

13. UPDATING PROGRAMMER SOFTWARE

The range of devices which can be programmed by this programmer can be increased by updating the system software which is held in either a plug-in eprom module or an eprom. Parameters, such as programmer Vcc and Vpp, are also controlled by software. ***Note that*** very old programmers may be fitted with eproms instead of eprom modules.

In order to update the programmer, proceed as described below.

1. Switch off and remove mains lead.

2. Turn programmer upside down with the front of the programmer facing you and remove six screws which hold the base on. Remove the base.

3. Locate the system eprom module or eprom which is on the track-side of the printed circuit board and remove it. Be careful not to damage the pins as eprom modules are re-used.

IF FITTING A NEW EPROM MODULE:

4. Make sure that the pins on the new update module are straight and then fit the replacement module ensuring that the edge marked - *FRONT OF THE PROGRAMMER* - is facing the front of the programmer and that the arrow on the right-hand side of the module is aligned with the screw on the right-hand side of the pcb. The module should then easily push into the place from where the original system eprom (module) was removed. Make sure that there are no bent or damaged pins.

TO COMPLETE ALL UPDATES:

5. Fit base and re-connect supply.

6. When the programmer is turned on, the sign on message will be:

W)Default parameters

7. It may be necessary to change the text/prompt language using Hex Key 8.

14. SPECIFICATIONS/FEATURES UNIQUE TO PARTICULAR MODULES

14.1. General

The M9000 is supplied as a base unit with power supplies, keyboard, LCD and a minimum of 2M bit RAM. It also has a serial port and high speed Centronics port as well as facilities for two plug-in modules for different types of eproms or microcontrollers. The specific features of certain modules are described in this section.

Features of the M9000 are detailed below:-

FUNCTION KEYS:
- All keys are mechanical
- 15 single function keys
- 16 hexadecimal keys for editing, etc.

DISPLAY:
- 48 characters
- Full upper and lower case Ascii set

ZIF SOCKETS:
- None on base unit but one to eight may be fitted depending on modules selected.

DEVICE TESTS:
- Illegal bit test
- Program
- Verify
- Data logic levels
- Programming & Erase margins
- Marginal Vcc
- Checksum

PROGRAMMING ALGORITHMS:
- High speed algorithms for AMD, Intel, Fujitsu, Renesas/Hitachi, Signetics, Toshiba, etc. (Typical algorithms such as Quick pulse, Snap, etc.)

SERIAL INTERFACE:
- RS232C with choice of hard or soft handshake on input and output
- Three wire connection to VDUs and printers simplifies connection
- Line parameters may be specified through the keyboard and are retained.
- Baud rates are:
  600, 1200, 1800, 2400, 3600, 4800, 7200, 9600, 19200 (displayed as 19K2)
- Choice of 7 or 8 data bits with optional parity
- Choice of 1 or 2 stop bits

PARALLEL INTERFACE:
- Centronics bi-directional interface for downloading data at high speed (equivalent to 50/60k baud) and for printing RAM contents, etc.

AUDIBLE RESPONSE:

- Error buzz - end of programming/verify indicated by intermittent buzz

EDIT FUNCTIONS AVAILABLE:

- Amend byte
- Set to Hex value between limits
- Merge blocks
- Split blocks
- Find character string and replace
- Checksum
- Copy block
- Complement
- Print RAM using RS232C or Centronics port

INPUT AND OUTPUT FORMATS:

- Data can be up and downloaded in one of the following formats:
  Intel, Motorola Exorciser, Binary, Tektronix or Ascii Hex space
- Extended addresses are catered for
- General purpose software can be used to upload programs because the last record can be followed by a user specified terminator.

RAM SIZE:

- M9000: choice of 2M, 8M or 32M

POWER:

- M9000-1 Voltage selector for 110V or 120V 50/60 Hz
- M9000-2 Voltage selector for 220V or 240V 50/60 Hz
- Supplies should always be within + or - 8%

14.2. PL450 Module For LCC Eproms

- The PL450 module is only available with two sockets.

- It should be noted that LCC eproms are available in a wide variety of sizes. The standard sockets provided on this module accommodate devices sized as follows:

  Nominal device size: 13.8 - 14.2 x 11.3 - 11.7mm
  Nominal thickness: 2.5 - 3.1mm

- As only two eproms can be fitted on each module, it is not possible to program a 32 bit word in one operation. The first two eproms can be programmed by starting programming at '*Ram start*' 0 and the second pair by starting at '*Ram start*' 2, etc.

- This socket module is now obsolete.

14.3. PL490 & PL491 Modules For 29FX00 In TSOP & SOP AND PL335 & PL336 Module For 29F016 In TSOP

- Some devices are programmed using a technique known as '*temporary sector unprotect*'. In this mode, even sectors which are protected can be programmed and erased.

- After erasing, programming and verification, the device is left with exactly the same sectors protected as before programming. (*Please refer to Section 15.3. for more details.*)

- Other devices such as the Sharp 28F400SUL are unprotected in the erase cycle.

- It is possible to protect sectors on devices such as Fujitsu 29F400TA or BA by selecting the device type - 29F400TA_SP or 29F400BA_SP.

  The user is asked to specify a mask. The least significant bit refers to the first sector, etc. For example, by specifying a mask of 000003, the first two sectors would be protected. In addition the *STEP* > key sets all the bits in the mask for the current device and the *STEP* < clears the mask. Erasing such a device removes all sector protection.

- **PL490Mk3 modification**:
  The following modification is required to program Intel 28F800B3. ***Note that*** this modification has already been made to all modules shipped after August 1999.

  1. Remove R5.
  2. Add a wire link between R5 (end near text R5) and right-hand end of capacitor to the right of text CS4.
  3. Link RN1 pins 5 and 12.

14.4. PL650 & PL668 Modules For PICS

- **PL650**
  This is not a gang module. There are four DIL sockets instead to cover the majority of PICs which program very quickly. The 16C55, for example, programs in about 1.5 seconds in Group 3.

- **PL668**
  This is a gang module. The outer two SOIC sockets are for 28 pin devices such as 16C57 whereas the inner two SOIC sockets are for 18 pin devices such as 16C56.

- Be careful to select the correct device name according to the specific device being used. Parts with suffix HS, XT or LP have been 'factory configured' to the relevant oscillator setting. Parts without these suffixes have not been 'factory configured'. It is important to select the exact part name before programming a device. If the wrong device type is

selected, the device may fail the *Blank Check* function. For example, if a PIC16C57 without a suffix is being programmed as a PIC16C57 XT, the device type - PIC16C57 without a suffix - must be selected.

- The M9000 can secure a device by programming a bit in the Configuration Register. Secured devices can still be read but the data is scrambled. When an attempt is made to read a secured device, the scrambled data is read into RAM, the socket LED flashes and the display indicates that the device is locked in order to warn the user that the data is not valid.

- The security/lock bit can be automatically programmed in the device. This option must be selected when using the *SET TYPE* function.

- The Checksum is calculated by simply adding up all the memory locations from 000H to the maximum user address (e.g. device address 1FF for the PIC16C54) and the unprotected state of the configuration fuses. The addition takes place 'byte-wide' which means that the low byte of a 12 bit wide memory location is added to the high byte (with the upper four bits always '*0*'). Any carry bits exceeding 16 bits are neglected.

- For devices with a word size of less than 16 bits, you must preset the unused RAM bits to zero. This will be set automatically if a master device is read into RAM. If the master is downloaded from a file, either read a blank device into RAM or fill RAM with zero before downloading the file or make sure that the file contains data for the whole device. If non-zero data is found in these bits, the M9000 may stop the device from being programmed with an error message – '*Illegal bit test fail*'.

- The RAM location for the configuration word must contain the correct data for your device. (*See the table below for the address of the configuration word.*) ***Note that*** some compilers do not automatically add the configuration byte into the object file. In such cases, the user is advised to add it to the file.

- Some devices such as the PIC16C715 contain PARITY bits. The parity bits are generated by the programmer and must not be contained in the programmer RAM or data file. The parity bits will be validated during the verify function and the error message - '*Parity verify fail*' - will be displayed if an error should occur.

Memory Map Type	***PL650 Group***	***Word Bits***	***Addresses Code***	***ID***	***Config. Data***
16C52 Device	2	12	0-17F	None	FFF
RAM		16	0-2FF	None	1FFE-F
16C54 Device	2	12	0-1FF	200-3	FFF
RAM		16	0-3FF	400-7	1FFE-F
16C55 Device	3	12	0-1FF	200-3	FFF
RAM		16	0-3FF	400-7	1FFE-F
16C55A Device	3	12	0-1FF	200-3	FFF
RAM		16	0-3FF	400-7	1FFE-F
16C554 Device	1	14	0-1FF	2000-3	2007
RAM		16	0-3FF	4000-7	400E-F
16C556 Device	1	14	0-3FF	2000-3	2007
RAM		16	0-7FF	4000-7	400E-F
16C558 Device	1	14	0-7FF	2000-3	2007

	RAM		16	0-FFF	4000-7	400E-F
16C56	Device	2	12	0-3FF	400-3	FFF
	RAM		16	0-7FF	800-7	1FFE-F
Memory Map Type		***PL650 Group***	***Word Bits***	***Addresses Code***	***ID***	***Config. Data***
16C57	Device	3	12	0-7FF	800-3	FFF
	RAM		16	0-FFF	1000-7	1FFE-F
16C57C	Device	3	12	0-7FF	800-3	FFF
	RAM		16	0-FFF	1000-7	1FFE-F
16C58A	Device	3	12	0-7FF	800-3	FFF
	RAM		16	0-FFF	1000-7	1FFE-F
16C61	Device	1	14	0-3FF	2000-3	2007
	RAM		16	0-7FF	4000-7	400E-F
16C620	Device	1	14	0-1FF	2000-3	2007
	RAM		16	0-3FF	4000-7	00E-F
16C621	Device	1	14	0-3FF	2000-3	2007
	RAM		16	0-7FF	4000-7	400E-F
16C622	Device	1	14	0-7FF	2000-3	2007
	RAM		16	0-FFF	4000-7	400E-F
16CE625	Device	1	14	0-7FF	2000-3	2007
	RAM		16	0-FFF	4000-7	400E-F
16C71	Device	1	14	0-3FF	2000-3	2007
	RAM		16	0-7FF	4000-7	400E-F
16C710	Device	1	14	0-1FF	2000-3	2007
	RAM		16	0-3FF	4000-7	400E-F
16C711	Device	1	14	0-3FF	2000-3	2007
	RAM		16	0-7FF	4000-7	400E-F
16C715	Device	1	14	0-7FF	2000-3	2007
	RAM		16	0-FFF	4000-7	400E-F
16F83	Device	1	14	0-1FF	2000-3	2007 2100-213F
	RAM		16	0-3FF	4000-7	400E-F 4200-427F
16C84	Device	1	14	0-3FF	2000-3	2007 2100-213F
	RAM		16	0-7FF	4000-7	400E-F 4200-427F
16F84	Device	1	14	0-3FF	2000-3	2007 2100-213F
	RAM		16	0-7FF	4000-7	400E-F 4200-427F
16F84A	Device	1	14	0-3FF	2000-3	2007 2100-213F
	RAM		16	0-7FF	4000-7	400E-F 4200-427F
17C42	Device	4	16	0-7FF	FE00	
	RAM		16	0-FFF	1FC00-1	
17C43	Device	4	16	0-0FFF	FE00	
	RAM		16	0-1FFF	1FC00-1	
17C44	Device	4	16	0-1FFF	FE00	
	RAM		16	0-3FFF	1FC00-1	

14.5. PL700 & PL701 Modules For Motorola 68HC705C8/9

- These devices can only be programmed in gang mode.
- The security/lock bit can be automatically programmed on the C8(A) and C9A but the C9 does not have a security/lock bit. This option must be selected when using the *SET TYPE* function.

- A secured 68HC705C8 cannot be detected by the M9000.

- The *Read Master* function takes about 15 seconds for the C8 because the device has to be read serially (30 seconds for the C9) but the C9A is read in parallel mode in 3 seconds.

- Data is programmed from RAM to the corresponding eprom address in the microcontroller. As there are gaps in the eprom map, it should be noted that the program checksum may not correspond to the RAM checksum.

- When using the 68HC705C8A in a 68HC705C8 application, program locations 1FF0H and 1FF1H to 00H.

- When using the 68HC705C9A in a 68HC705C9 application, program locations 3FF0H and 3FF1H to 00H.

- The 68HC705C9A requires the following modification to the PL700 and PL701 modules shipped before November 1996:-

    1. **PL700 modification**:
       Add a wire link between CS1.32, CS2.32, CS3.32, CS4.32 and PL1.83 (right-hand row, column 19 of DIN connector).

    2. **PL701 modification**:
       Add a wire link between CS1.29, CS2.29, CS3.29, CS4.29 and PL1.83 (right-hand row, column 19 of DIN connector).

14.6. PL874 & PL874 Mk2 Module

- In May 1993, a new version of the PL874 module was released. The new module can program the same devices as the old module (8748/9H) and also the 8741/2 family. The new version is known as the PL874 Mk2. The module can be identified by fitting it to a M9000 programmer and by pressing buttons B and 8. The M9000 displays 874_Mk2 for the new module or 874 for the old one. Software release 2.62 or later must be fitted for the new module. A mixture of new and old modules can be used to program 8748/9Hs. However, if new and old modules are fitted and an attempt is made to program devices other than 8748/9H, the M9000 will instruct the user to remove the old module.

14.7. PL875 & PL876 Modules For 8751 Family

- The PL875 and PL876 modules shipped before March 1993 may not be able to program Intel 87C51FC and other FX versions as standard. Consult Lloyd Research for further information. Some wire links can be added to early modules to correct this situation.

- The PL875 and PL876 modules shipped before July 1997 may not be able to program Atmel 89C55 as standard because additional signals are required. These can be added as follows:-

1. PL1.63 to all copy sockets pin 10 (PL875) or pin 11 (PL876).

2. Wire a small signal NMOS FET such as BSS100 as follows:-
   Source to PL1.47; Gate to PL1.17; Drain to PL1.46

- Lock bits can be programmed with software release 2.39 or later.

- Before reading a device, the M9000 attempts to find out whether or not there is a device in each socket. Before software version 2.71, this test involved running the micro and detecting the output on the ALE line. However, this test is not 100% perfect because this signal is not available if the PC exceeds the maximum address of the micro. We have, therefore, changed the device insertion test to read the Intelligent Identifier which is available in about 95% of cases. If a device does not have an Intelligent Identifier, then the old method is used.

- Use the *SET TYPE* function to select the following options:

  Dallas 87C520: *Watchdog Auto Enabled*

- The option bits in the Dallas 87C520 can be read or verified after programming. The *Check Master* function displays the state of the option bits. Bit 3 of the 8 bit Hex number represents the Watchdog Auto Enabled option. Therefore, FF = the default state - '*No*' and F7 = the blown state - '*Yes*'. The *Verify* function will verify the data and the option bits. If the data fails but the option bits pass, it will display '*Verify fail*'. If the data and option bits fail, it will display '*OPTIONS Verify fail*'. After Read Master, the current option set up is checked against the master device. If there is a difference, it will display '*Warning Check OPTIONS*'. In this case, change the option set up to the required state before programming more devices.

- Use the *SET TYPE* function to select the following options:

  Atmel 89S8252: *EEPROM address range 2000 to 27FF*
  Disable serial program

  The FLASH address range 0 to 1FFF is always enabled.

  Atmel 89S83: *Disable serial program*

  The *Device Erase* function is designed to erase both the FLASH and the EEPROM arrays in one operation. It is not possible to erase only the FLASH array.

- Use the *SET TYPE* function to select the following options:

Atmel 89C51Rx2:
Inhibit XRAM
Boot loader jump bit
X2 Mode
Boot status byte *xxH*
Software boot vector *xxH*
Software security byte *xxH*

The FLASH address range is always enabled.

14.8. PL71E/L Modules For Motorola Micros Such As 68HC711E9 & 711L6

- These modules are only available with two PLCC sockets.
- As the PL71E/L do not use Motorola's 'PROG' mode, early (as well as later devices) can be programmed.
- The user must select which parts of the device are to be programmed. There are three parts which can be programmed, namely the eprom, eeprom and Config. Register. (Some devices in the family do not have three parts.) At least one part must be selected!
- Devices can only be gang programmed.
- Data is read and written to RAM only if the user selects the relevant part. Therefore if, for example, the eprom is selected on the E9 variant, only that part will be read into RAM and only the eprom will be programmed.
- Data is read and written into RAM according to the memory map of each device. (S*ee following example.*)
- When the Config. Register is erased, an automatic erase of the eeprom is performed by the device.
- When the eeprom or Config. Register is erased, an automatic blank check is performed afterwards ***ONLY*** on the part which has been selected, i.e. the eeprom, the Config. Register or both (whichever is appropriate).
- When the security bit is programmed in the Config. Register, further attempts to read, blank check or verify the device will erase the eeprom and Config. Register.
- When the security bit is programmed in the Config. Register and the eprom area is not blank, the device will be secured and not recognised by the programmer. Motorola has specified that the eprom must be erased before the device is used again.

- When programming the Config. Register, some variants such as 68HC811E2 will continue to read some of the config. bits as 1 (1111PP11). Thus, a further verify of the Config. Register would result in a verify fail. After reading a master device into the programmer, the Config. location - RAM address 103F - must be edited to the required value. For instances, in order to set the EEPROM location to B800 – BFFF, the upper 4 bits of the Config. Register must be set to 1011.

- To program the security bit on the 68S711E9, use the *SET TYPE* function to set '*program lock bits*' to 1.

Memory map for Motorola 68HC711E2/E9/E20, 68HC11A1 & 68HC11A0

The **68HC711E9** and **EA9** have three arrays which can be selected or de-selected using the *SET TYPE* function.

Memory map	*Address*	*Size*
EPROM	D000 to FFFF	12K bytes
EEPROM	B600 to B7FF	512 bytes
CONFIG	103F	1 byte

The **68S711E9** has three arrays which can be selected or de-selected using the *SET TYPE* function.

Memory map	*Address*	*Size*
EPROM	D000 to FFFF	12K bytes
EEPROM	B600 to B7FF	512 bytes
CONFIG	103F	1 byte

Note: Bit 3 (NOSEC) is programmed when the '*Program lock bits*' is set to 1. It is not programmed from RAM address 103F.

The **68HC711E20** has three arrays which can be selected or de-selected using the *SET TYPE* function.

Memory map	*Address*	*Size*
EPROM	9000 to AFFF	8K bytes
	D000 to FFFF	12K bytes
EEPROM	B600 to B7FF	512 bytes
CONFIG	103F	1 byte

The **68HC711E32** has three arrays which can be selected or de-selected using the *SET TYPE* function.

Memory map	*Address*	*Size*
EPROM	7000 to AFFF	16K bytes
	C000 to FFFF	16K bytes
EEPROM	B600 to B7FF	512 bytes
CONFIG	103F	1 byte

The **68HC811E2** has two arrays which can be selected or de-selected using the *SET TYPE* function.

Memory map	*Address*	*Size*
EEPROM	F800 to FFFF	2K bytes
CONFIG	103F	1 byte

The **68HC11A1** has two arrays which can be selected or de-selected using the *SET TYPE* function.

Memory map	*Address*	*Size*
EEPROM	B600 to B7FF	512 bytes
CONFIG	103F	1 byte

The **68HC11A0** has one array.

Memory map	*Address*	*Size*
CONFIG	103F	1 byte

Memory map for Motorola 68HC711L6

The **68HC711L6** has three arrays which can be selected or de-selected using the *SET TYPE* function.

Memory map	*Address*	*Size*
EPROM	C000 to FFFF	16K bytes
EEPROM	B600 to B7FF	512 bytes
CONFIG	103F	1 byte

14.9. PL620 & PL62x Modules For ST/SGS_Thomson ST62 Family

- Devices can only be gang programmed.
- Data is read and written into RAM according to the memory map of each device. (S*ee over page.*)

- The option byte(s) will always be programmed and verified from the following memory map location except for the lock bit which is selected separately.

- When the security bit is programmed in the device without an option byte(s), further attempts to read the device will display - '*Locked @ checksum*' - and no data will be loaded into RAM. The *Verify* function will perform normally for locked devices without an option byte(s). The checksum is the same for secured and non-secured devices without an option byte(s).

- Some devices have a 'slave' copy of the option byte(s). Each bit of the slave is inverted. This means that the option byte(s) can only be programmed once. It is not possible to program another option bit at a later time as it will fail the illegal bit test. Hence, it is also not possible to secure a device at a later time. Normally the user is not concerned about the slave. In case of difficulty, the slave is also read into the M9000 RAM as shown on the following page. The slave byte is not counted in the checksum but the option byte(s) is.

- SGS_Thomson has not allocated a unique address space for each of the EPROM, EEPROM and OPTION arrays. The compiler generates three files - EPROM.HEX, EEPROM.HEX and OPTION.HEX. Therefore, the following memory maps have been implemented for these devices.

- A simple batch file can sort out the 3 files and manage the address map. (*Examples follow for the 62T30B, 62T25C and 62T60B.*)

Memory map for Motorola 68HC711L6

Memory Map	*Address Range*	*Type*
62T00C	0BA0 to 0F9F, 0FF0 to 0FF7, 0FFC to 0FFF	EPROM
	1000	LS MASTER OPTION
	1001	MS MASTER OPTION
	[1002 to 1003] READ FUNCTION ONLY	SLAVE OPTION
62E/T01C	0880 to 0F9F, 0FF0 to 0FF7, 0FFC to 0FFF	EPROM
	1000	LS MASTER OPTION
	1001	MS MASTER OPTION
	[1002 to 1003] READ FUNCTION ONLY	SLAVE OPTION
62E20	0000 to 0FF7, 0FFC to 0FFF	EPROM
62E25	0000 to 0FF7, 0FFC to 0FFF	EPROM
62T08C	0BA0 to 0F9F, 0FF0 to 0FF7, 0FFC to 0FFF	EPROM
	1000	LS MASTER OPTION
	1001	MS MASTER OPTION
	[1002 to 1003] READ FUNCTION ONLY	SLAVE OPTION
62T09C	0BA0 to 0F9F, 0FF0 to 0FF7, 0FFC to 0FFF	EPROM
	1000	LS MASTER OPTION
	1001	MS MASTER OPTION
	[1002 to 1003] READ FUNCTION ONLY	SLAVE OPTION
62T10/15	0880 to 0F9F, 0FF0 to 0FF7, 0FFC to 0FFF	EPROM
62T10C/15C &		
62T52C/53C	0880 to 0F9F, 0FF0 to 0FF7, 0FFC to 0FFF	EPROM
	1000	LS MASTER OPTION
	1001	MS MASTER OPTION
	[1002 to 1003] READ FUNCTION ONLY	SLAVE OPTION
62T20/25	0080 to 0F9F, 0FF0 to 0FF7, 0FFC to 0FFF	EPROM
62T20C/25C	0080 to 0F9F, 0FF0 to 0FF7, 0FFC to 0FFF	EPROM
	1000	LS MASTER OPTION
	1001	MS MASTER OPTION
	[1002 to 1003] READ FUNCTION ONLY	SLAVE OPTION
62T28C	0080 to 0F9F, 0FF0 to 0FF7, 0FFC to 0FFF	EPROM
	1010 to 17FF, 1810 to 1FFF	EPROM
	2000	LS MASTER OPTION
	2001	MS MASTER OPTION
	[2002 to 2003] READ FUNCTION ONLY	SLAVE OPTION
62T30B	0080 to 0F9F, 0FF0 to 0FF7, 0FFC to 0FFF	EPROM
	1010 to 17FF, 1810 to 1FFF	EPROM
	2000 to 207F	EEPROM
	2080	MASTER OPTION
	[2082] READ FUNCTION ONLY	SLAVE OPTION
62T55B	0080 to 0F9F, 0FF0 to 0FF7, 0FFC to 0FFF	EPROM
	1000	MASTER OPTION
62T55C	0080 to 0F9F, 0FF0 to 0FF7, 0FFC to 0FFF	EPROM
	1000	LS MASTER OPTION
	1001	MS MASTER OPTION
	[1002 to 1003] READ FUNCTION ONLY	SLAVE OPTION
62T60B/65B	0080 to 0F9F, 0FF0 to 0FF7, 0FFC to 0FFF	EPROM

	1000 to 107F	EEPROM
	1080	MASTER OPTION

Memory Map	*Address Range*	*Type*
62T60C/65C	0080 to 0F9F, 0FF0 to 0FF7, 0FFC to 0FFF	EPROM
	1000 to 107F	EEPROM
	1080	LS MASTER OPTION
	1081	MS MASTER OPTION
	[1082 to 1083] READ FUNCTION ONLY	SLAVE OPTION
62T62C/63C	0880 to 0F9F, 0FF0 to 0FF7, 0FFC to 0FFF	EPROM
	1000 to 103F	EEPROM
	1040	LS MASTER OPTION
	1041	MS MASTER OPTION
	[1042 to 1043] READ FUNCTION ONLY	SLAVE OPTION

Note: The *READ* function stores the slave option bytes into RAM for information only. The *VERIFY* and *PROGRAM* functions action the complement of the master option bytes.

Example of batch file for 62T30B using the Centronics port

This batch file selects the device type and downloads the three data files - EPROM.HEX, EEPROM.HEX and OPTION.HEX - into the correct RAM address range ready to program the device.

```
REM Batch Name 62T30B.BAT:-

REM File to select device type 62T30B and initialise RAM
copy 62T30B.txt prn:
REM File to set L/M9000 ready to receive EPROM data file from PC
REM and place in RAM address range 0000 to 1FFF
copy EPROM.TXT prn:
REM Copy EPROM data file to Centronics port
copy EPROM.HEX prn:
REM File to set L/M9000 ready to receive EEPROM data file from PC
REM and place in RAM address range 2000 to 207F
copy EEPROM.TXT prn:
REM Copy EEPROM data file to Centronics port
copy EEPROM.HEX prn:

REM File to set L/M9000 ready to receive OPTION data file from PC
REM and place in RAM address range 2080 to 2080
copy OPTION.TXT prn:
REM Copy OPTION data file to Centronics port
copy OPTION.HEX prn:
```

Contents of 62T30B.TXT:-

```
%                          ;Enter remote control without echo or prompt
E[G                        ;Select Production Mode
20G2080G0G                 ;Initialise RAM to 00.
T"62T30B SGS_Thomson"GG[G  ;Select device type 62T30B SGS Thomson
                           ; Lock bit – none
```

Contents of EPROM.TXT:-

```
IIG0G01FFFG0G              ;Download in Intel format from address 0
                           ;to 1FFF with data loading at RAM start 0
```

Contents of EEPROM.TXT:-

```
IIG0G07FG2000G             ;Download in Intel format from file address 0
                           ;to 07F with data loading at RAM start 2000
```

Contents of OPTION.TXT:-

```
IIG1700G1700G2080G          ;Download in Intel format from file address 1700
                            ;to 1700 with data loading at RAM start 2080

    ;Note all characters after the ';' to the end of line are ignored and
    ;can be used for notes.
```

Example of batch file for 62T60B or 62T65B using the Centronics port

This batch file selects the device type and downloads the three data files - EPROM.HEX, EEPROM.HEX and OPTION.HEX - into the correct RAM address range ready to program the device.

```
REM Batch Name 62T60B.BAT:-

REM File to select device type 62T60B and initialise RAM
copy 62T60B.60B prn:
REM File to set L9000 ready to receive EPROM data file from PC
REM and place in RAM address range 0000 to 0FFF
copy EPROM.60B prn:


REM Copy EPROM data file to Centronics port
copy EPROM.HEX prn:
REM File to set L9000 ready to receive EEPROM data file from PC
REM and place in RAM address range 1000 to 107F
copy EEPROM.60B prn:
REM Copy EEPROM data file to Centronics port
copy EEPROM.HEX prn:
REM File to set L9000 ready to receive OPTION data file from PC
REM and place in RAM address range 1080 to 1080
copy OPTION.60B prn:
REM Copy OPTION data file to Centronics port
copy OPTION.HEX prn:
```

Contents of 62T60B.60B:-

```
%                          ;Enter remote control without echo or prompt
E[G                        ;Select Production Mode
20G1080G0G                 ;Initialise RAM to 00.
T"62T60B SGS_Thomson"GG[G  ;Select device type 62T60B SGS Thomson
                           ;Lock bit – none
```

Contents of EPROM.60B:-

```
IIG0G00FFFG0G              ;Download in Intel format from address 0
                           ;to 0FFF with data loading at RAM start 0
```

Contents of EEPROM.60B:-

```
IIG0G07FG1000G             ;Download in Intel format from file address 0
                           ;to 07F with data loading at RAM start 1000
```

Contents of OPTION.60B:-

```
IIG0B80G0B80G1080G          ;Download in Intel format from file address 0B80
                            ;to 0B80 with data loading at RAM start 1080

    ;Note all characters after the ';' to the end of line are ignored and
    ;can be used for notes.
```

Example of batch file for 62T00C, 62E01C, 62T08C, 62T09C, 62T10C, 62T15C, 62T20C & 62T25C using the Centronics port

This batch file selects the device type and downloads the two data files -EPROM.HEX and OPTION.HEX - into the correct RAM address range ready to program the device. ***Note that*** the OPTION.HEX file has to be downloaded twice in order to place the bytes into the correct RAM address space.

```
REM Batch Name 62T25C.BAT:-

REM File to select device type 62T25C and initialise RAM
copy 62T25C.25C prn:
REM File to set L9000 ready to receive EPROM data file from PC
REM and place in RAM address range 0000 to 0FFF
copy EPROM.25C prn:
REM Copy EPROM data file to Centronics port
copy EPROM.HEX prn:
REM File to set L9000 ready to receive OPTION data file from PC
REM and place the least significant byte in RAM address range 1000
REM to 1000
copy OPTIONL.25C prn:
REM Copy OPTION data file to Centronics port
copy OPTION.HEX prn:
REM File to set L9000 ready to receive OPTION data file from PC
REM and place the most significant byte in RAM address range 1001
REM to 1001
copy OPTIONM.25C prn:
REM Copy OPTION data file to Centronics port
copy OPTION.HEX prn:
```

Contents of 62T25C.25C:-

```
%                          ;Enter remote control without echo or prompt
E[G                        ;Select Production Mode
20G1001G0G                 ;Initialise RAM to 00
T"62T25C SGS_Thomson"GG[G  ;Select device type 62T25C SGS Thomson
                           ;Lock bit - none
                           ;To select 62E01C, 62T08C, 62T09C,
                           ;62T10C, 62T15C or 62T20C replace the
                           ;device name with the required one in
                           ;the command string
```

Contents of EPROM.25C:-

```
IIG0G00FFFG0G              ;Download in Intel format from address 0
                           ;to 0FFF with data loading at RAM start 0
```

Contents of OPTIONL.25C:-

```
IIG0F80G0F80G1000G          ;Download in Intel format from file address 0F80
                            ;to 0F80 with data loading at RAM start 1000
```

Contents of OPTIONM.25C:-

```
IIG0780G0780G1001G          ;Download in Intel format from file address 0780
                            ;to 0780 with data loading at RAM start 1001
```

Example of batch file for 62T28C using the Centronics port

This batch file selects the device type and downloads the two data files - EPROM.HEX and OPTION.HEX - into the correct RAM address range ready to program the device. ***Note that*** the OPTION.HEX file has to be downloaded twice in order to place the bytes into the correct RAM address space.

REM Batch Name 62T28C.BAT:-

```
REM File to select device type 62T28C and initialise RAM
copy 62T28C.28C prn:
REM File to set L9000 ready to receive EPROM data file from PC
REM and place in RAM address range 0000 to 1FFF
copy EPROM.28C prn:
REM Copy EPROM data file to Centronics port
copy EPROM.HEX prn:
REM File to set L9000 ready to receive OPTION data file from PC
REM and place the least significant byte in RAM address range 2000
REM to 2000
copy OPTIONL.28C prn:
REM Copy OPTION data file to Centronics port
copy OPTION.HEX prn:
REM File to set L9000 ready to receive OPTION data file from PC
REM and place the most significant byte in RAM address range 2001
REM to 2001
copy OPTIONM.28C prn:
REM Copy OPTION data file to Centronics port
Copy OPTION.HEX prn:
```

Contents of 62T28C.28C:-

```
%                          ;Enter remote control without echo or prompt
E[G                        ;Select Production Mode
20G2001G0G                 ;Initialise RAM to 00
T"62T28C SGS_Thomson"GG[G  ;Select device type 62T28C SGS Thomson
                           ;Lock bit - none
```

Contents of EPROM.28C:-

```
IIG0G01FFFG0G              ;Download in Intel format from address 0
                           ;to 1FFF with data loading at RAM start 0
```

Contents of OPTIONL.28C:-

```
IIG005FG005FG2000G         ;Download in Intel format from file address 005F
                           ;to 005F with data loading at RAM start 2000
```

Contents of OPTIONM.28C:-

```
IIG004FG004FG2001G          ;Download in Intel format from file address 004F
                            ;to 004F with data loading at RAM start 2001

    ;Note all characters after the ';' to the end of line are ignored and
    ;can be used for notes.
```

Example of batch file for 62T55C using the Centronics port

This batch file selects the device type and downloads the two data files - EPROM.HEX and OPTION.HEX - into the correct RAM address range ready to program the device. ***Note that*** the OPTION.HEX file has to be downloaded twice in order to place the bytes into the correct RAM address space.

```
REM Batch Name 62T5555BAT:-

REM File to select device type 62T55C and initialise RAM
copy 62T55C.55C prn:
REM File to set L9000 ready to receive EPROM data file from PC
REM and place in RAM address range 0000 to 0FFF
copy EPROM.55C prn:
REM Copy EPROM data file to Centronics port
copy EPROM.HEX prn:
REM File to set L9000 ready to receive OPTION data file from PC
REM and place the least significant byte in RAM address range 1000
REM to 1000
copy OPTIONL.55C prn:
REM Copy OPTION data file to Centronics port
copy OPTION.HEX prn:
REM File to set L9000 ready to receive OPTION data file from PC
REM and place the most significant byte in RAM address range 1001
REM to 1001
copy OPTIONM.55C prn:
REM Copy OPTION data file to Centronics port
copy OPTION.HEX prn:
```

Contents of 62T55C.55C:-

```
%                          ;Enter remote control without echo or prompt
E[G                        ;Select Production Mode
20G1001G0G                 ;Initialise RAM to 00
T"62T55C SGS_Thomson"GG[G  ;Select device type 62T55C SGS Thomson
                           ;Lock bit – none
```

Contents of EPROM.55C:-

```
IIG0G00FFFG0G              ;Download in Intel format from address 0
                           ;to 0FFF with data loading at RAM start 0
```

Contents of OPTIONL.55C:-

```
IIG005FG005FG1000G         ;Download in Intel format from file address 005F
                           ;to 005F with data loading at RAM start 1000
```

Contents of OPTIONM.55C:-

```
IIG004FG004FG1001G          ;Download in Intel format from file address 004F
                            ;to 004F with data loading at RAM start 1001

    ;Note all characters after the ';' to the end of line are ignored and
    ;can be used for notes.
```

Example of batch file for 62T60C & 62T65C using the Centronics port

This batch file selects the device type and downloads the three data files - EPROM.HEX, PROM.HEX and OPTION.HEX - into the correct RAM address range ready to program the device. ***Note that*** the OPTION.HEX file has to be downloaded twice in order to place the bytes into the correct RAM address space.

REM Batch Name 62T65C.BAT:-

```
REM File to select device type 62T65C and initialise RAM
copy 62T65C.65C prn:
REM File to set L9000 ready to receive EPROM data file from PC
REM and place in RAM address range 0000 to 0FFF
copy EPROM.65C prn:
REM Copy EPROM data file to Centronics port
copy EPROM.HEX prn:
REM File to set L9000 ready to receive EEPROM data file from PC
REM and place in RAM address range 1000 to 107F
copy EEPROM.65C prn:
REM Copy EEPROM data file to Centronics port
copy EEPROM.HEX prn:
REM File to set L9000 ready to receive OPTION data file from PC
REM and place the least significant byte in RAM address range 1080
REM to 1080
copy OPTIONL.65C prn:
REM Copy OPTION data file to Centronics port
copy OPTION.HEX prn:
REM File to set L9000 ready to receive OPTION data file from PC
REM and place the most significant byte in RAM address range 1081
REM to 1081
copy OPTIONM.65C prn:
REM Copy OPTION data file to Centronics port
copy OPTION.HEX prn:
```

Contents of 62T65C.65C:-

```
%                          ;Enter remote control without echo or prompt
E[G                        ;Select Production Mode
20G1081G0G                 ;Initialise RAM to 00.
T"62T65C SGS_Thomson"GG[G  ;Select device type 62T65C SGS Thomson
                           ;Lock bit – none
```

Contents of EPROM.65C:-

```
IIG0G00FFFG0G              ;Download in Intel format from address 0
                           ;to 0FFF with data loading at RAM start 0
```

Contents of EEPROM.65B:-

```
IIG0G07FG1000G          ;Download in Intel format from file address 0
                        ;to 07F with data loading at RAM start 1000
```

Contents of OPTIONL.65C:-

```
IIG005FG005FG1080G      ;Download in Intel format from file address 005F
                        ;to 005F with data loading at RAM start 1080
```

Contents of OPTIONM.65C:-

```
IIG004FG004FG1081G      ;Download in Intel format from file address 004F
                        ;to 004F with data loading at RAM start 1081
```

```
;Note all characters after the ';' to the end of line are ignored and
;can be used for notes.
```

Example of batch file for 62T63C using the Centronics port

This batch file selects the device type and downloads the three data files - EPROM.HEX, EEPROM.HEX and OPTION.HEX - into the correct RAM address range ready to program the device. ***Note that*** the OPTION.HEX file has to be downloaded twice in order to place the bytes into the correct RAM address space.

REM Batch Name 62T63C.BAT:-

```
REM File to select device type 62T63C and initialise RAM
copy 62T63C.63C prn:
REM File to set L9000 ready to receive EPROM data file from PC
REM and place in RAM address range 0000 to 0FFF
copy EPROM.63C prn:
REM Copy EPROM data file to Centronics port
copy EPROM.HEX prn:
REM File to set L9000 ready to receive EEPROM data file from PC
REM and place in RAM address range 1000 to 103F
copy EEPROM.63C prn:
REM Copy EEPROM data file to Centronics port
copy EEPROM.HEX prn:
REM File to set L9000 ready to receive OPTION data file from PC
REM and place the least significant byte in RAM address range 1040
REM to 1040
copy OPTIONL.63C prn:
REM Copy OPTION data file to Centronics port
copy OPTION.HEX prn:
REM File to set L9000 ready to receive OPTION data file from PC
REM and place the most significant byte in RAM address range 1041
REM to 1041
copy OPTIONM.63C prn:
REM Copy OPTION data file to Centronics port
copy OPTION.HEX prn:
```

Contents of 62T63C.63C:-

```
%                          ;Enter remote control without echo or prompt
E[G                        ;Select Production Mode
20G1041G0G                 ;Initialise RAM to 00.
T"62T63C SGS_Thomson"GG[G  ;Select device type 62T63C SGS Thomson
                           ;Lock bit – none
```

Contents of EPROM.63C:-

```
IIG0G00FFFG0G              ;Download in Intel format from address 0
                           ;to 0FFF with data loading at ram start 0
```

Contents of EEPROM.63C:-

```
IIG0G03FG1000G          ;Download in Intel format from file address 0
                        ;to 03F with data loading at RAM start 1000
```

Contents of OPTIONL.63C:-

```
IIG005FG005FG1040G      ;Download in Intel format from file address 005F
                        ;to 005F with data loading at RAM start 1040
```

Contents of OPTIONM.63C:-

```
IIG004FG004FG1041G      ;Download in Intel format from file address 004F
                        ;to 004F with data loading at RAM start 1041
```

```
;Note all characters after the ';' to the end of line are ignored and
;can be used for notes.
```

14.10. PL71D Modules For Motorola Micros Such As MC68HC711D3

- This module is only available with four PLCC sockets.
- As the PL71D uses Motorola's 'BOOTSTRAP' mode, early XC parts may not be able to be programmed.
- Devices can only be gang programmed.
- Data is read and written into RAM according to the memory map of each device. (*See below*.)

Memory map for Motorola MC68HC711D3

The MC68HC711D3 has one array.

Memory map	*Address*	*Size*
EEPROM	F000 to FFFF	4K bytes

14.11. PL71K Modules For Motorola Micros Such As 68HC711KA4

- This module is only available with two PLCC sockets.
- As the PL71K does not use Motorola's 'PROG' mode, early (as well as later) devices can be programmed.
- The user must select which parts of the device are to be programmed. There are three parts which can be programmed, namely the eprom, eeprom and Config. Register. At least one part must be selected!
- Devices can only be gang programmed.
- Data is read and written to RAM only if the user selects the relevant part. If, for example, the eprom is selected on the KA4 variant, only that part will be read into RAM and only the eprom will be programmed.
- Data is read and written into RAM according to the memory map of each device as follows.
- When the eeprom or Conf. Register is erased, an automatic blank check is performed afterwards ***ONLY*** on the part which has been selected, i.e. the eeprom, the Config. Register or both (whichever is appropriate).

Memory map for Motorola 68HC711KA4

The 68HC711KA4 has three arrays which can be selected or de-selected using the *SET TYPE* function.

Memory map	*Address*	*Size*
EPROM	A000 to FFFF	24K bytes
EEPROM	0D80 to 0FFF	640 bytes
CONFIG	003F	1 byte

14.12. PL715/6/7 Modules For Motorola Micros Such As 68HC705B5 & B16

- This module is only available with two sockets.
- Only gang programming is allowed - no set programming.
- Data is read and written into RAM according to the memory map of each device as follows.
- Data is programmed from RAM to the corresponding eprom address in the microcontroller (except for the SEC bit in the OPTR Register). As there are gaps in the eprom map, it should be noted that the program checksum may not correspond to the RAM checksum.
- The user must select which parts of the device are to be programmed. There are two parts which can be programmed, namely the eprom and eeprom. (Some devices in the family do not have two parts.) At least one part must be selected!
- Data is read and written to RAM only if the user selects the relevant part. If, for example, the eprom is selected on the B16 variant, only that part will be read into RAM and only the eprom will be programmed.
- When the eeprom is erased, an automatic blank check is performed afterwards ***ONLY*** on the part which has been erased.
- The user must select '*Program Lock bits = 1*' in order to secure the device. This will program the SEC bit in the OPTR Register.
- When the security bit is programmed in the OPTR Register and the eprom area is not blank, the device will be secured and not recognised by the programmer. Motorola has specified that the eprom must be erased before the device is used again.
- ***Note that*** the OPTR (except the SEC bit) and MOR Registers will be programmed from RAM and must, therefore, be set up correctly before programming the device. A safe way to handle these registers is to read a blank device into RAM before downloading the required data file to RAM.

Memory map for Motorola 68HC705B5

Memory map	*Address*	*Size*
EPROM	0020 to 004F	48 bytes
EPROM	0100 to 01FF	256 bytes
EPROM	0800 to 1EFF	5888 bytes including OPTR at 1EFE
EPROM	1FF0 to 1FFF	16 bytes

Memory map for Motorola 68HC705B16 & 68HC705B16N

Memory map	*Address*	*Size*
EPROM	0020 to 004F	48 bytes
EPROM	0300 to 3DFF	15106 bytes including MOR at 3DFE
EPROM	3FF0 to 3FFF	16 bytes
EEPROM	0100 to 01FF	256 bytes including OPTR at 0100

Memory map for Motorola 68HC705B32/X32

Memory map	*Address*	*Size*
EPROM	0400 to 7DFF	31232 bytes
EPROM	7FDE	1 byte MOR
EPROM	7FF0 to 7FFF	16 bytes
EEPROM	0100 to 01FF	256 bytes including OPTR at 0100

14.13. PL860 & PL861 Modules For Zilog Micros Such As Z86E03/04/06/08 & PL863 & PL864 Modules For Zilog Micros Such As Z86E30

- Due to a change of specification by Zilog, module PL860 has been withdrawn and replaced by PL860 Mark 2. If an old module is used, the programmer will display '*Remove PL module 860*'.
- Only gang programming is allowed - no set programming.
- The user must select which options in the device are to be programmed.
- A secured or killed device cannot be detected by the programmer.

Use the *SET TYPE* function to select the following options:

Z86E03 & Z86E06: EPROM Protect (Program lock bits)

		Permanent WDT Disable
		Auto Latch Disable
		RC Oscillator
Z86E02, 04 & 08:		EPROM Protect (Program lock bits)
	L	Low Noise
	A	Auto Latch Disable
	W	Permanent WDT Enable
		Kill bit

Z86E02_1925:		EPROM Protect (Program lock bits)
	R	RC Oscillator
	L	Low Noise
	A	Auto Latch Disable
	W	Permanent WDT Enable
		Kill bit

Z86E30_1873	E	EPROM Protect (Program lock bits)
	O	RC Oscillator
	A	Auto Latch Disable
	W	Permanent WDT Enable
	F	Remove Osc Feedback Res
	R	RAM Protect

Z86E30
Note that 12MHz parts must have SL1873 next to the part number. Old 12MHz non-SL1873 parts cannot be programmed on the PL86x. 16MHz are similar to the SL1873 and can be programmed.

Z86733 & Z86E34		EPROM Protect (Program lock bits)
	X	XTAL Oscillator
	A	Auto Latch Disable
	W	Permanent WDT Enable
	L	Low Freq. oscillator
	R	Ram Protect
		Kill bit

The Option bits in the Z86E02, Z86E04, Z86E08 and Z86E30 can be read or verified after programming. The *Check Master* function displays the state of the option bits. The *Verify* function will verify the data and the option bits. If the data fails but the option bits pass, it will display '*Verify fail*'. If the data and option bits fail, it will display '*OPTIONS Verify fail*'. After Read Master, the current option set up is checked against the master device. If there is a difference, it will display – '*Warning Check OPTIONS*'. In this case change, the option set up to the required state before programming more devices.

14.14. PL720/21/22/3 Modules For Motorola Micros Such As 68HC705P6 & P9

- Only gang programming is allowed - no set programming.

- The erased data state is 00 for the P6 and P9, and FF for P18.
- Data is read into and programmed from RAM corresponding to the memory address map in the microcontroller as follows. As there are gaps in the memory map, ***note that*** the program checksum may not correspond to the ram checksum. It is a good idea to fill the RAM with 00 data before downloading the master data file.
- The whole device is programmed including the MOR byte. It is important to set the unused MOR bits to zero in order to pass the device *Verify* and *Program* functions.
- When using older 68HC705P6 devices, the MOR byte may fail to program on the first attempt. Mask set - 0E20Y - will program first time.
- The security/lock bit - SECURE - can be automatically programmed in the 68HC705P6A device. This option must be selected when using the *SET TYPE* function.
- The 68HC805P18 has very limited functions available. Only the *Program* function is enabled. The program cycle includes erase, program and a final verify. It is not possible to read a master device or do a separate verify.

Memory map for Motorola 68HC705P9

Memory map	*Address*	*Size*
EPROM	0020 to 004F	48 bytes
EPROM	0100 to 08FF	2048 bytes
EPROM MOR	0900 to 0900	1 byte (3 bits 0 - 2)
EPROM Vectors	1FF0 to 1FFF	16 bytes

Memory map for Motorola 68HC705P6

Memory map	*Address*	*Size*
EPROM	0020 to 004F	48 bytes
EPROM	0100 to 12FF	
EPROM MOR	1F00 to 1F00	1 byte (6 bits 0 - 5)
EPROM Vectors	1FF0 to 1FFF	16 bytes

Memory map for Motorola 68HC705P6A

Memory map	*Address*	*Size*
EPROM	0020 to 004F	48 bytes
EPROM	0100 to 12FF	
EPROM MOR	1EFF to 1F00	2 bytes
EPROM Vectors	1FF0 to 1FFF	16 bytes

Memory map for Motorola 68HC805P18

Memory map	*Address*	*Size*
EEPROM	0020 to 004F	48 bytes
EEPROM	1FC0 to 3EFF	8000 bytes
EEPROM MOR	3F00 to 3F01	2 bytes
EEPROM Vectors	3FF0 to 3FFF	16 bytes

14.15. PL65x, PL66x & PL67x Modules For PICs

- Be careful to select the correct device type as some of the numbers are misleading. Some 16C73A devices, for example, have the following marking:-

  PIC16C73
  A/JW

  If the wrong type is selected, the device will be destroyed.

- Also be careful to select the correct device name according to the specific device being used. Parts with suffix HS, XT or LP have been 'factory configured' to the relevant oscillator setting. Parts without these suffixes have not been 'factory configured'. It is important to select the exact part name before programming a device. If the wrong device type is selected, the device may fail the *Blank Check* function. If, for example, a PIC16C57 without a suffix is being programmed as a PIC16C57 XT, the device type - PIC16C57 without a suffix - must be selected.

- The M9000 can secure a device by programming a bit in the Configuration Register. Secured devices can still be read but the data is scrambled. When an attempt is made to read a secured device, the scrambled data is read into RAM, the socket LED flashes and the display indicates that the device is locked to warn the user the data is not valid.

- The security/lock bit can be automatically programmed in the device. This option must be selected when using the *SET TYPE* function.

- The checksum is calculated by simply adding up all the memory locations from 000H to the maximum user address (e.g. device address FFF for the PIC16C73) and the unprotected state of the configuration fuses. The addition takes place 'byte wide' which means that the low byte of a 14 bit wide memory location is added to the high byte (with the upper two bits always '*0*'). Any carry bits exceeding 16 bits are neglected.

- For some devices such as PIC12C508/9, the *Check Master* function will display the standard check sum and the Microchip PIC check sum.

- For devices with a word size of less than 16 bits, you must preset the unused RAM bits to zero. This will be set automatically if a master device is read into RAM. If the master is downloaded from a file, either read a blank device into RAM or fill RAM with zero before downloading the file or make sure the file contains data for the whole device.

- The RAM location for the configuration word must contain the correct data for your device (*see the following tables for the address of the configuration word*). The reserved bits must be set as specified by the manufacturer (usually set to 1). Some compilers do not automatically add the configuration byte(s) to the file. In such cases, the user is advised to add this information.

- Some devices have a calibration word pre-programmed by the manufacturer. In order to leave this word unchanged, set answer – '*No*' - to question – '*Program Calibration*' in the *SET TYPE* function. If you are using erasable devices, be careful to read the calibration word before erasing the device and program the same value back into the device. If you select '*Yes*', you must ensure that the correct value is in RAM before programming the device.

- Some FLASH devices such as the PIC10F20x, 16F505, 12F508, 12F509, 12F629 and 16F630 have a calibration word pre-programmed by the manufacturer. The *Erase* function reads the calibration word before erasing the devices and restores it after the devices have been erased. Some PL modules can only erase one device at a time.

- Some devices such as the PIC16C662 contain parity bits. These bits are generated by the programmer and must not be contained in the programmer RAM or data file. The parity bits will be validated during the *Verify* function and the error message - '*Parity verify fail*' - will be displayed if an error occurs.

- Some devices such as the PIC16F87x contain a Data Eeprom array. The user must select the Data Eeprom array if this part of the device is to be read and programmed. (***Note*** + sign in the following memory maps.)

- The two pass verify is carried out with Vcc at 5.5V (Hi) and 4.5V (Lo) for 'non flash' devices. Flash parts are verified once at nominal Vcc.

- PL651 and PL652 modules shipped before June 2005 require a hardware modification to enable PIC18F2431 to be used. ***Note that*** a wire link ***MUST*** be added between pins 7 and 11 on all sockets.

Memory map for 10F200 & 10F204

	Word Bits	***Code***	***Addresses ID***	***Configuration***	
Device	12	0-00FE	0100-3	01FF	Physical address
Ram	16	0-01FD	0200-7	1FFE-F	Logical address

##	***Word Bits***	***Reset Vector***	***Backup OSCCAL***

Device	12	00FF	0104
Ram	16	01FE-F	0208-9

Memory map for 12C508

	Word Bits	*Code*	*Addresses Calibration (See Note)*	*ID*	*Configuration*
Device	12	0-01FE	01FF	0200-3	0FFF
Ram	16	0-03FD	03FE-F	0400-7	1FFE-F

Memory map for 12C509 and 16C505

	Word Bits	*Code*	*Addresses Calibration (See Note)*	*ID*	*Configuration*
Device	12	0-03FE	03FF	0400-3	0FFF
Ram	16	0-07FD	07FE-F	0800-7	1FFE-F

Memory map for 12C671

	Word Bits	*Code*	*Addresses Calibration (See Note)*	*ID*	*Configuration*
Device	14	0-03FE	03FF	2000-3	2007
Ram	16	0-07FD	07FE-F	4000-7	400E-F

Memory map for 12C672

	Word Bits	*Code*	*Addresses Calibration (See Note)*	*ID*	*Configuration*
Device	14	0-07FE	07FF	2000-3	2007
Ram	16	0-0FFD	0FFE-F	4000-7	400E-F

Memory map for 12F508, 10F202 & 10F206

	Word Bits	*Code*	*Addresses ID*	*Configuration*	
Device	12	0-01FE	0200-3	03FF	Physical address
Ram	16	0-03FD	0400-7	1FFE-F	Logical address

##	*Word Bits*	*Reset Vector*	*Backup OSCCAL*
Device	12	01FF	0204
Ram	16	03FE-F	0408-9

Memory map for 16C62, 16C62A, 16C622A, 16CE625, 16C64, 16C64A & 16C72

	Word Bits	*Code*	*Addresses ID*	*Configuration*
Device	14	0-07FF	2000-3	2007
Ram	16	0-0FFF	4000-7	400E-F

Memory map for 16C63, 16C65, 16C65A, 16C662, 16C73, 16C73A, 16C73B, 16C74, 16C74A, 16C923 & 16C924

	Word Bits	*Code*	*Addresses ID*	*Configuration*
Device	14	0-0FFF	2000-3	2007
Ram	16	0-1FFF	4000-7	400E-F

Memory map for 16C66, 16C67, 16C76 and 16C77

	Word Bits	*Code*	*Addresses ID*	*Configuration*
Device	14	0-1FFF	2000-3	2007
Ram	16	0-3FFF	4000-7	400E-F

Memory map for 16F54

	Word Bits	*Code*	*Addresses ID*	*Configuration*	
Device	12	0-01FF	0200-3	03FF	Physical address
Ram	16	0-03FF	0400-7	1FFE-F	Logical address

Memory map for 16F505, 16F506, 12F509 and 12F510

	Word Bits	*Code*	*Addresses ID*	*Configuration*	
Device	12	0-03FE	0400-3	07FF	Physical address
Ram	16	0-07FD	0800-7	1FFE-F	Logical address

##	*Word Bits*	*Reset Vector*	*Backup OSCCAL*
Device	12	03FF	0404
Ram	16	07FE-F	0808-9

***Note that*** the *Read* function reads these values into RAM. The *Verify*, *Blank Check* and *Program* functions do not action these locations. The *Erase* function reads the values, erases the devices and restores the original values. ***ONLY*** one device can be erased at a time.

Memory map for 12F519

	Word Bits	*Code*	*Addresses Data*	*ID*	*Configuration*	
Device	12	0-03FE	0400-043F	0440-3	07FF	Physical address
Ram	16	0-07FD	0800-087F	0880-7	1FFE-F	Logical address

##	*Word Bits*	*Reset Vector*	*Backup OSCCAL*
Device	12	03FF	0444-7
Ram	16	07FE-F	0888-F

Note that the *Read* function reads these values into RAM. The *Verify*, *Blank Check* and *Program* functions do not action these locations. The *Erase* function reads the values, erases the devices and restores the original values. ***ONLY*** one device can be erased at a time.

Memory map for 16F84A

	Word Bits	***Code***	***ID***	***Configuration***	***Data EEPROM***
			Addresses		
Device	14	0-03FF	2000-3	2007	2100-213F +
Ram	16	0-07FF	4000-7	400E-F	4200-407F

Memory map for 16F872

	Word Bits	***Code***	***ID***	***Configuration***	***Data EEPROM***
			Addresses		
Device	14	0-07FF	2000-3	2007	2100-213F +
Ram	16	0-0FFF	4000-7	400E-F	4200-407F

Memory map for 16F73, 16F873 & 16F874

	Word Bits	***Code***	***ID***	***Configuration***	***Data EEPROM***
			Addresses		
Device	14	0-0FFF	2000-3	2007	2100-217F +
Ram	16	0-1FFF	4000-7	400E-F	4200-42FF

Memory map for 16F876 & 16F877

	Word Bits	***Code***	***ID***	***Configuration***	***Data EEPROM***
			Addresses		
Device	14	0-1FFF	2000-3	2007	2100-21FF +
Ram	16	0-3FFF	4000-7	400E-F	4200-43FF

Memory map for 16F1933 & 16F1934

	Word Bits	***Code***	***ID***	***Configuration***	***Data EEPROM***
			Addresses		
Device	14	0-0FFF	8000-3	8007-8	F000 F0FF
Ram	16	0-1FFF	10000-7	1000E-11	1E000-1E1FF

Memory map for 16F1936 & 16F1937

	Word		*Addresses*		
	Bits	***Code***	***ID***	***Configuration***	***Data EEPROM***
Device	14	0-1FFF	8000-3	8007-8	F000 F0FF
Ram	16	0-3FFF	10000-7	1000E-11	1E000-1E1FF

Memory map for 16F1938 & 16F1939

	Word		*Addresses*		
	Bits	***Code***	***ID***	***Configuration***	***Data EEPROM***
Device	14	0-3FFF	8000-3	8007-8	F000 F0FF
Ram	16	0-7FFF	10000-7	1000E-11	1E000-1E1FF

Memory map for 17C756

	Word	*Addresses*	
	Bits	***Code***	***Configuration***
Device	16	0-3FFF	FE00
Ram	16	0-7FFF	1FC00-1

Memory map for 18C242 & 18C442

	Word		*Addresses*	
	Bits	***Code***	***ID***	***Configuration***
Device	16	0-1FFF	200000-7	300000-7
Ram	16	0-3FFF	200000-7	300000-7

Note: 32M bits of RAM required

Memory map for 18C252, 18C452 & 18C858

	Word		*Addresses*	
	Bits	***Code***	***ID***	***Configuration***
Device	16	0-3FFF	200000-7	300000-7

Ram	16	0-7FFF	200000-7	300000-7

Note: 32M bits of RAM required

Memory map for 18Fx220 ***(See note *)***

	Word Bits	***Code***	***Addresses ID***	***Configuration***	***EEPROM***
Device	16	0-07FF	200000-7	300000-D *	F00000-FF
Ram	16	0-0FFF	200000-7	300000-D *	3F0000-FF +

Note: 32M bits of RAM required

Memory map for 18Fx320 ***(See note *)***

	Word Bits	***Code***	***Addresses ID***	***Configuration***	***EEPROM***
Device	16	0-0FFF	200000-7	300000-D *	F00000-FF
Ram	16	0-1FFF	200000-7	300000-D *	3F0000-FF +

Note: 32M bits of RAM required

Memory map for 18F24xx & 18C44xx ***(See note *)***

	Word Bits	***Code***	***Addresses ID***	***Configuration***	***EEPROM***
Device	16	0-1FFF	200000-7	300000-D *	F00000-FF
Ram	16	0-3FFF	200000-7	300000-D *	3F0000-FF +

Note: 32M bits of RAM required

Memory map for 18F25x5 & 18C45x5 ***(See note *)***

	Word Bits	***Code***	***Addresses ID***	***Configuration***	***EEPROM***
Device	16	0-5FFF	200000-7	300000-D *	F00000-FF
Ram	16	0-BFFF	200000-7	300000-D *	3F0000-FF +

Note: 32M bits of RAM required

Memory map for 18F25xx & 18C45xx ***(See note *)***

	Word Bits	*Code*	*Addresses ID*	*Configuration*	*EEPROM*
Device	16	0-3FFF	200000-7	300000-D *	F00000-FF
Ram	16	0-7FFF	200000-7	300000-D *	3F0000-FF +

Note: 32M bits of RAM required

Memory map for 18F26xx & 18C46xx ***(See note *)***

	Word Bits	*Code*	*Addresses ID*	*Configuration*	*EEPROM*
Device	16	0-7FFF	200000-7	300000-D *	F00000-FF
Ram	16	0-FFFF	200000-7	300000-D *	3F0000-FF +

Note: 32M bits of RAM required

Note 1 - *

Configuration bits - CPx (Code Sector Protection) and WRTx (Table Write Protection) - are programmed from the *SET TYPE* function setup and ***NOT*** from the RAM memory map. In order to select the CPx and WRTx bits, answer – '*Program lock bits?*' = '*Yes*'. The programmer will then ask you to select the required bits in the 'Sector *protect mask?*' The default setting is 'unprotect' where the mask bits are set to 1. In order to protect a sector, set the required bit to '*0*'. For example, the mask for PIC18Fxxx family is E00FC00F where the right-hand pair of Hex digits represents bits - CP3 to CP0 (device address 300008 bits 3 - 0), the next pair of Hex digits represents bits - CPD to CPB (device address 300009 bits 6 and 7), the next pair of Hex digits represents bits - WRT3 to WRT0 (device address 30000A bits 3 - 0) and the next pair of Hex digits represents bits - WRTD to WRTB (device address 30000B bits 7 - 5). Pressing the *STEP* > key will protect all sectors while pressing < *STEP* leaves all sectors unprotected.

Bit Number

Address	7	6	5	4	3	2	1	0		
300008	-	-	-	-	CP3	CP2	CP1	CP0	=	0FH
300009	CPD	CPB	-	-	-	-	-	-	=	C0H
30000A	-	-	-	-	WRT3	WRT2	WRT1	WRT0	=	0FH
30000B	WRTD	WRTB	WRTC	-	-	-	-	-	=	E0H

These bytes are not included in the device checksum.

Note 2 +

If it is required to program the eeprom, the eeprom data must be relocated to 3F0000 in the file ***BEFORE*** downloading to the programmer. If it is difficult to relocate the eeprom data, the file data can be downloaded a second time using the following settings:-

Load data from?	F00000
Load data to?	F000FF
Ram start?	3F0000

14.16. PL850 Module For National COP8SAx7 Family

- Only gang programming is allowed - no set programming.
- The erased data state is 00.
- Data is read into and programmed from RAM corresponding to the memory address map in the microcontroller as can be seen in the following tables.
- The whole device is programmed including the UES and ECON Register.
- The security/lock bit - SEC - can be automatically programmed in the COP8SA device. This option must be selected when using the *SET TYPE* function.

Memory map for National COP8SAA716/20/28

Memory map	*Device Address*	*Ram Address*	*Size*
EPROM	0000 to 03FF	0000 to 03FF	1K bytes
ECON	0420	0400	1 byte
UES	0400 to 0407	0401 to 0408	8 bytes

Memory map for National COP8SAB720/28

Memory map	*Device Address*	*Ram Address*	*Size*
EPROM	0000 to 07FF	0000 to 07FF	2K bytes
ECON	0820	0800	1 byte
UES	0800 to 0807	0801 to 0808	8 bytes

Memory map for National COP8SAC720/28/40

Memory map	*Device Address*	*Ram Address*	*Size*
EPROM	0000 to 0FFF	0000 to 0FFF	4K bytes
ECON	1020	1000	1 byte
UES	1000 to 1007	1001 to 1008	8 bytes

14.17. PL836 & PL849 Modules For Toshiba 87PS38/87PM40

- Only gang programming is allowed - no set programming.
- Data is read into and programmed from RAM corresponding to the memory address map in the microcontroller as can be seen in the following tables.

Memory map for Toshiba 87PS38

Memory map	*Device Address*	*Ram Address*
EPROM	4000 to 7FFF	4000 to 7FFF
EPROM	11100 to 1FFFF	11100 to 1FFFF

Memory map for Toshiba 87PM40AN

Memory map	*Device Address*	*Ram Address*
EPROM	4000 to 7FFF	4000 to 7FFF

14.18. PL732 Module For Motorola Micros Such As 68HC705F32

- This module is only available with two sockets.
- Only gang programming is allowed - no set programming.
- Data is read and written into RAM according to the memory map of each device as can be seen in the following table.
- Data is programmed from RAM to the corresponding eprom address in the microcontroller. As there are gaps in the eprom map, ***note that*** the program checksum may not correspond to the RAM checksum.

- The user must select which parts of the device are to be programmed. There are two parts which can be programmed, namely the eprom and eeprom. (Some devices in the family do not have two parts.) At least one part must be selected!

- Data is read and written to RAM only if the user selects the relevant part. If, for example, the eprom is selected on the F32 variant, only that part will be read into RAM and only the eprom will be programmed.

- When the eeprom is erased, an automatic blank check is performed afterwards ***ONLY*** on the part which has been erased.

Memory map for Motorola 68HC705F32

Memory map	*Address*	*Size*
EPROM	8000 to FDFF	32256 bytes
EPROM	FFF0 to FFFF	16 bytes
EEPROM	0400 to 04FF	256 bytes

14.19. PL836 Module For Motorola Micros Such As 68HC708XL36

- This module is only available with two sockets.

- Only gang programming is allowed - no set programming.

- Data is read and written into RAM according to the memory map of each device as can be seen in the following table.

- Data is programmed from RAM to the corresponding eprom address in the microcontroller. As there are gaps in the eprom map, ***note that*** the program checksum may not correspond to the RAM checksum.

- Motorola has implemented a security feature on this family of microcontrollers. In order to pass the security test, the M9000 RAM must contain the same data as the device data in the address range - FFF6 to FFFD. Therefore, before a *Read Master* or *Check Master* function can be performed, the user must enter the security data into the M9000 RAM. If the security fails (the RAM data does not equal the device data in the address range - FFF6 to FFFD), then the M9000 will display the error message:-

  Security fail. Enter code
  in RAM at FFF6 to FFFD

  (This assumes data starts at ram address 0.)

- All the devices in the PL836 have to contain the same security data. This means blank and programmed devices ***CANNOT*** be present in the programmer at the same time. If there is a mix of blank (00) and programmed security codes, the M9000 will display the error message :-

Security fail. Cannot mix
BLANK & PROGRAMMED ICs

- ***Note that*** devices with blank security data will pass the M9000 security check.

Memory map for Motorola 68HC708XL36

Memory map	*Address*	*Size*
EPROM	6E00 to FDFF	36864 bytes
EPROM	FFDE to FFFF	34 bytes

14.20. PL855 Module For National COP87Lxx Family

- Only gang programming is allowed - no set programming.
- The erased data state is FF.
- The security/lock byte can be automatically programmed in the COP87Lxx device. This option must be selected when using the *SET TYPE* function.
- The 16K device types - COP87L84C, COP87L84D, COP87L84E, COP87L84F and COP87L84G - are all grouped together under the name - COP87L84C_G.
- The 32K device types - COP87L84H, COP87L84I, COP87L84J, COP87L84K, COP87L84L, COP87L84M, COP87L84N, COP87L84O, COP87L84P, COP87L84Q and COP87L8R - are all grouped together under the name - COP87L84H_R.

14.21. PL229, PL230, PL231, PL233 & PL266 Modules For Atmel AVR Family

- The user must select which parts of the device are to be programmed. There are three parts which can be programmed, namely the Flash, Eeprom and Option Register. At least one part must be selected!
- Devices can only be gang programmed.
- Data is read and written to RAM only if the user selects the relevant part. If, for example, the Flash is selected, only that part will be read into RAM and only the Flash will be programmed.
- When the Flash or Eeprom is erased, an automatic erase of both the arrays is performed by the device even if only one array has been selected.
- Data is read and written into RAM according to the memory map of each device as can be seen in the following table.
- The user must select the option – Fuse Bits – by using the *SET TYPE* function. The fuse bits are not changed by the *Erase* function.
- When the security bit is programmed in the device, further attempts to read the device will display – '*Locked @ checksum*'.

- Atmel has not allocated a unique address space for the Flash and Eeprom arrays. The Atmel compiler generates 2 files - .HEX for Flash and .EEP for Eeprom. The following memory maps have, therefore, been implemented for these devices.
- A simple batch file can sort out the two files and manage the address map as can be seen in the following example for the 90S2313.

- For all devices except for the MEGA644(P) and XMEGAxxD4:- After *Read Master*, the current option set up is checked against the master device. If there is a difference, it will display – '*Warning Check OPTIONS*'. In this case, change the Option set up to the required state before programming more devices.

- The MEGA644(P) and XMEGAxxD4 reads, programs and verifies the OPTION bytes from RAM. See following memory map.

- The blank check does not test the state of the fuse bits as they can be supplied pre-programmed. They can also be programmed high or low in the program pass.

- The XMEGAxxD4 programs the LOCK bits from the SECTOR PROTECT mask. To implement this set "*Program lock bits*" to "*Yes*" and set the required bits to program low in the sector protect mask. The mask is a direct mapping of the NVM lock bits.

  Examples:

Sector Protect Mask	
000000FF	No lock bits set
00000000	All lock bits set
000000FC	LB[1:0] RWLOCK (both set to ZERO)

00

Memory map for Atmel 90S2313 family

			Address Range	
Memory Map	***Device***		***RAM***	***Type***
90S2313	0000 to 03FF (16 bit)		0000 to 07FF	FLASH
	0000 to 007F (8 bit)		0800 to 087F	EEPROM
	No address	++	0880 (Read only)	Fuse Bits

Note: Select the fuse bits by using the *SET TYPE* function.

Memory map for Atmel 90S8535 family

		Address Range	
Memory Map	***Device***	***RAM***	***Type***
90S8535	0000 to 0FFF (16 bit)	0000 to 1FFF	FLASH
	0000 to 01FF (8 bit)	2000 to 21FF	EEPROM

	No address	++2200 (Read only)	Fuse Bits

Note: i) Select the fuse bits by using the *SET TYPE* function.

iii) ++ The *Read Master* function will place the fuse and lock bits into the next RAM address for information only.

Bit 0: FSTRT Fuse (0 = programmed, 1 = erased)
Bit 5: SPIEN Fuse (0 = programmed, 1 = erased)
Bit 6: Lock Bit2 (0 = programmed)
Bit 7: Lock Bit1 (0 = programmed)

Memory map for Atmel MEGA8535 family

		Address Range	
Memory Map	***Device***	***RAM***	***Type***
MEGA8535	0000 to 0FFF (16 bit)	0000 to 1FFF	FLASH
	0000 to 01FF (8 bit)	2000 to 21FF	EEPROM
	No address	++ 2200/2 (Read only)	Fuse Bits

Note: Select the fuse bits by using the *SET TYPE* function.

Memory map for Atmel MEGA163 & MEGA16 families

		Address Range	
Memory Map	***Device***	***RAM***	***Type***
MEGA163	0000 to 1FFF (16 bit)	0000 to 3FFF	FLASH
	0000 to 01FF (8 bit)	4000 to 41FF	EEPROM
	No address	++ 4200/2 (Read only)	Fuse Bits

Note: i) Select the fuse bits by using the *SET TYP*E function.

ii) Re: MEGA16 & MEGA8535
If the External Crystal or External RC configuration is selected and the SPIEN fuse is disabled, the device cannot be read by the programmer.

It is very unlikely that customers will need to program devices in the M9000 more than once. However, we have found that, if the device is reprogrammed, the Device Identifier Code can be erased. However, if the user has left the Identifier Check on as part of the device selection, the part will be rejected but, if the Identifier Check is turned off,

the part can be reprogrammed. Also ***note that*** the *Erase* function does ***NOT*** erase the configuration bytes which can simply be reprogrammed. Devices with programmed configuration bytes will ***NOT*** fail a blank check.

iii) ++ The *Read Master* function will place the fuse and lock bits into the next RAM addresses for information only.

Ram 4200 Fuse low bits (0 = programmed, 1 = erased)

	MEGA163	***MEGA16***	***MEGA8535***
Bit 0:	CKSEL0	CKSEL0	CKSEL0
Bit 1:	CKSEL1	CKSEL1	CKSEL1
Bit 2:	CKSEL2	CKSEL2	CKSEL2
Bit 3:	CKSEL3	CKSEL3	CKSEL3
Bit 4:	-	SUT0	SUT0
Bit 5:	SPIEN	SUT1	SUT1
Bit 6:	BODEN	BODEN	BODEN
Bit 7:	BODLEVEL	BODLEVEL	BODLEVEL

Ram 4201 Fuse high bits (0 = programmed, 1 = erased)

	MEGA163	***MEGA16***	***MEGA8535***
Bit 0:	BOOTRST	BOOTRST	BOOTRST
Bit 1:	BOOTSZ0	BOOTSZ0	BOOTSZ0
Bit 2:	BOOTSZ1	BOOTSZ1	BOOTSZ1
Bit 3:	-	EESAVE	EESAVE
Bit 4:	-	CKOPT	CKOPT
Bit 5:	-	SPIEN	SPIEN
Bit 6:	-	JTAGEN	WDTON
Bit 7:	-	OCDEN	S8535C

Ram 4202 Lock bits (0 = programmed, 1 = erased)

	MEGA163	***MEGA16 & MEGA8535***
Bit 0:	Lock bit 1	Lock bit 1
Bit 1:	Lock bit 2	Lock bit 2
Bit 2:	Boot lock bit 01	Boot lock bit 01
Bit 3:	Boot lock bit 02	Boot lock bit 02
Bit 4:	Boot lock bit 11	Boot lock bit 11
Bit 5:	Boot lock bit 12	Boot lock bit 12

Memory map for Atmel MEGA644P & MEGA644 families

Device	***Address Range RAM***	***Type***
00000 to 07FFF (16 bit)	00000 to 0FFFF	FLASH
00000 to 007FF (8 bit)	10000 to 107FF	EEPROM
00000 to 00003	10800 to 10803	OPTION Fuse Bits

Ram 10800 Fuse low bits (0 = programmed, 1 = erased)

Bit 0:	CKSEL0
Bit 1:	CKSEL1
Bit 2:	CKSEL2
Bit 3:	CKSEL3
Bit 4:	SUT0
Bit 5:	SUT1
Bit 6:	CKOUT
Bit 7:	CKDIV8

Ram 10801 Fuse high bits (0 = programmed, 1 = erased)

Bit 0:	BOOTRST
Bit 1:	BOOTSZ0
Bit 2:	BOOTSZ1
Bit 3:	EESAVE
Bit 4:	WDTON
Bit 5:	SPIEN
Bit 6:	JTAGEN
Bit 7:	OCDEN

Ram 10802 Fuse extended bits (0 = programmed, 1 = erased)

Bit 0:	BODLEVEL0
Bit 1:	BODLEVEL1
Bit 2:	BODLEVEL2
Bit 3:	-
Bit 4:	-
Bit 5:	-
Bit 6:	-
Bit 7:	-

Ram 10803Lock bit byte (0 = programmed, 1 = erased)

Bit 0:	Lock bit 1	** see below*
Bit 1:	Lock bit 2	** see below*
Bit 2:	Boot lock bit 01	
Bit 3:	Boot lock bit 02	
Bit 4:	Boot lock bit 11	
Bit 5:	Boot lock bit 12	

** Lock bits 1 & 2 are programmed by setting the Program lock bits in the 'Set Type' function.*

Memory map for Atmel TINY26 family

		Address Range	
Memory Map	***Device***	***RAM***	***Type***

TINY26	0000 to 03FF (16 bit)	0000 to 07FF	FLASH
	0000 to 007F (8 bit)	0800 to 087F	EEPROM
	No address	++ 0880/1 (Read only)	Fuse Bits

Note: i) Select the fuse bits by using the *SET TYPE* function.

ii) ++ The *Read Master* function will place the fuse bits into the next RAM addresses for information only.

Ram 0880 Fuse low bits (0 = programmed, 1 = erased)
TINY26

Bit 0:	CKSEL0
Bit 1:	CKSEL1
Bit 2:	CKSEL2
Bit 3:	CKSEL3
Bit 4:	SUT0
Bit 5:	SUT1
Bit 6:	CKOPT
Bit 7:	PLLCK

Ram 0881 Fuse high bits (0 = programmed, 1 = erased)
TINY26

Bit 0:	BODEN
Bit 1:	BODLEVEL
Bit 2:	EESAVE
Bit 3:	SPIEN
Bit 4:	RSTDISBL
Bit 5:	-
Bit 6:	-
Bit 7:	-

Memory map for Atmel XMEGA32D4 & XMEGA64D4

		Address Range	
Memory Map	***Device***	***RAM***	***Type***
XMEGA32D4	800000 - 808FFF	00000 - 08FFF	FLASH
	8C0000 - 8C03FF	09000 - 093FF	EEPROM
	8F0020 - 8F0025	09400 - 09405	Fuse Bytes
XMEGA64D4	800000 - 810FFF	00000 - 10FFF	FLASH
	8C0000 - 8C07FF	11000 - 117FF	EEPROM
	8F0020 - 8F0025	11800 - 11805	Fuse Bytes

Example of batch file for 90S2313 using the Centronics port

This batch file selects the device type with option bit – SPIEN - and downloads the two data files - ATFLASH.HEX and ATEEPROM.EEP - into the correct RAM address range ready to program the device.

```
REM Batch Name 90S2313.BAT:-
REM File to select device type 90S2313 and initialise RAM.
copy 90S2313.TXT prn:

REM File to set L9000 ready to receive FLASH data file from PC
REM and place in RAM address range 0000 to 07FF
copy ATFLASH.TXT prn:
REM Copy FLASH data file to Centronics port
copy ATFLASH.HEX prn:
REM File to set L9000 ready to receive EEPROM data file from PC
REM and place in RAM address range 0800 to 087F
copy ATEEPROM.TXT prn:
REM Copy EEPROM data file to Centronics port
copy ATEEPROM.EEP prn:
REM File to program the device
copy PROGRAM.TXT prn:
```

Contents of 90S2313.TXT:-

```
%                          ;Enter remote control without echo or prompt
E[G                        ;Select Production Mode
20G087FG0G                 ;Initialise RAM to 00
T"90S2313 Atmel"           ;Select device type 90S2313 Atmel
GG                         ;1 IC per set, 16 Bit word
]]G                        ;Identifier - Compatible
[[G                        ;Lock bit   - none
]G                         ;FLASH      - Yes
]G                         ;EEPROM     - Yes
]G                         ;OPTION     - Yes
]G                         ;SPIEN      - Yes
[G                         ;FSTRT      - No

        ; Note the set type string could all be on one command line as:-
        ; T"90S2313 Atmel"GG]]G[[G]G]G]G]G[G
```

Contents of ATFLASH.TXT:-

```
IIG0G07FFG0G               ; Download in Intel format from address 0
                           ; to 07FF with data loading at ram start 0
```

Contents of ATEEPROM.TXT:-

```
IIG0G07FG0800G          ;Download in Intel format from file address 0
                        ;to 07F with data loading at ram start 0800
```

Contents of PROGRAM.TXT:-

```
P                       ;Program the device

    ;Note all characters after the ';' to the end of line are ignored and
    ;can be used for notes.
```

14.22. PL992 Module For Sony CPX7500P10 Family

- Only gang programming is allowed – no set programming.
- Data is read into and programmed from RAM corresponding to the memory address map in the microcontroller as can be seen in the following table.

Memory map for Sony 7500P10S

Memory map	*Device Address*	*Ram Address*
EPROM	2000 to 1FFFF	2000 to 1FFFF

14.23. PL707 & PL708 Module For Motorola Micros Such As 68HC705JJ7/JP7

- Only gang programming is allowed - no set programming.
- Data is read and written into RAM according to the memory map of each device as can be seen in the following table.
- Data is programmed from RAM to the corresponding eprom address in the microcontroller (except for the EPMSEC bit in the COP Register). As there are gaps in the eprom map, it should be noted that the program checksum may not correspond to the RAM checksum.
- The user must select which parts of the device are to be programmed. There are two parts which can be programmed, namely the EPROM and PEPROM. (Some devices in the family do not have two parts.) At least one part must be selected!
- Data is read and written to ram only if the user selects the relevant part. If, for example, only the eprom is selected, only that part will be read into RAM and programmed from RAM.
- ***Note that*** the MOR Register is part of the EPROM area and will be programmed from RAM if the EPROM is selected and must, therefore, be set up correctly before programming the device. A safe way to handle this Register is to read a blank device into RAM before down loading the required data file to RAM.

- The user must select '*Program Lock bits = 1*' to secure the device. This will program the EPMSEC bit in the COP Register.

- When the security bit is programmed in the COP Register, the device will be secured and not recognised by the programmer.

- The PEPROM matrix is not part of the Motorola memory map. The M9000 programmer uses the RAM address allocated in the following table for the PEPROM matrix.

Memory map for Motorola 68HC705JJ7/JP7

Memory map	*Device Address*	*Ram Address*
EPROM	0700 to 1EFF	6144 bytes
EPROM	1FF1 to 1FFF	15 bytes including MOR at 1FF1
PEPROM	0000 to 0007	64 bits

Note:	Address 0000 = PEPROM	Column 0, Row 0 = bit 0, Row 7 = bit 7, etc.
	0001 =	Column 1
	0007 =	Column 7

14.24. PL308 Module For 8 Pin Serial EE

- Atmel 25xxx family requires module issue PL308A. Special Function B8 will display the module issue number. Exchange module pcbs are available. Software version 2.D4 or later is required.

14.25. PL76x Module For Motorola Micros Such As 68HC908xxx

- Be careful when up grading from a 68HC908AZ60 to an AZ60A. The AZ60A cannot be programmed with the same master data as it uses different internal registers and a different memory map as can be seen in the following example.

- Programmed 68HC908xxx devices can usually be erased without knowing the security code. However, this facility may not be available for all mask sets.

- The user must select which parts of the device are to be programmed. The AZ60x has four parts which can be programmed, namely the FLASH1, FLASH2, EEPROM1 and EEPROM2. The other parts only have FLASH1. At least one part must be selected! The default state is program/verify FLASH1 and FLASH2. EPROM1 and EEPROM2 are not programmed and the M9000 skips over pages which do not need to be programmed.

- Devices can only be gang programmed.

- Data is read and written to RAM only if the user selects the relevant part. If, for example, just the FLASH1 is selected, only that part will be read into RAM and only the FLASH1 will be programmed.

- Data is read and written into RAM according to the memory map of each device as can be seen in the following table. As there are gaps in the map, it should be noted that the program checksum may not correspond to the RAM checksum.

- When a device is erased, an automatic blank check is performed afterwards ***ONLY*** on the part which has been selected.

- Motorola has implemented a security feature on this family of microcontrollers. In order to pass the security test, the M9000 RAM must contain the same data as the device data in the address range - FFF6 to FFFD. Therefore, before a *Read Master* or *Check Master* function can be performed, the user must enter the security data into the M9000 RAM. If the security fails (i.e. the RAM data does not equal the device data in the address range FFF6 to FFFD), then the M9000 will display the error message:-

  Security fail.Enter code
  in RAM at FFF6 to FFFD

  This assumes data starts at ram address 0.

- All the devices in the PL76x have to contain the same security data. This means that blank and programmed devices ***CANNOT*** be present in the programmer at the same time. If there is a mix of blank – 00 - and programmed security codes, the M9000 will display the error message:-

  Security fail. Cannot mix
  BLANK & PROGRAMMED ICs

- ***Note that*** brand new devices and erased devices will pass the M9000 security check.

- Motorola has made three different versions of the AZ60 device, each with a different internal MONITOR. The PL760 supports all three versions, but only one version can be present in the PL760 at a time.

  Version 1: Mask set H62A
  Version 2: Mask set J61D and J74Y
  Version 3: Not released

  If there is a mix of versions, the M9000 will display the following error message and flash the LED by the higher version types.

  Security fail.Enter code
  in RAM at FFF6 to FFFD

Memory map for Motorola 68HC908AB32

The 68HC908AB32 has two arrays which can be selected or de-selected by using the *SET TYPE* function.

	Address	*Size*	*Blank*	*Description*
FLASH1:-	8000 to FDFF	32256 Bytes	FF	
	FF7E	1 Bytes	FF	Block Protect Registers
	FFD0 to FFFF	48 Bytes	FF	User Vectors

	Address	*Size*	*Blank*	*Description*
EEPROM1:-	0800 to 09FF	512 Bytes	FF	
	FE10 to FE11	2 Bytes	FF	EEDIVNVR
	FE1C	1 Byte	FF	EENVR array configuration

Note that data bit 4 of EENVR can only be programmed once and cannot be erased.

Memory map for Motorola 68HC908AS60A

The 68HC908AS60A has four arrays which can be selected or de-selected by using the *SET TYPE* function.

	Address	*Size*	*Blank*	*Description*
FLASH1:-	8000 to FDFF	32256 Bytes	FF	
	FF80 to FF81	2 Bytes	FF	Block Protect Registers
	FFD2 to FFD3	2 Bytes	FF	User Vector
	FFDA to FFFF	38 Bytes	FF	User Vectors
FLASH2:-	0450 to 05FF	432 Bytes	FF	
	0E00 to 7FFF	29184 Bytes	FF	
EEPROM1:-	0800 to 09FF	512 Bytes	FF	
	FE10 to FE11	2 Bytes	FF	EE1DIVNVR
	FE1C	1 Byte	F0	EE1NVR array configuration (*See following note*)
EEPROM2:-	0600 to 07FF	512 Bytes	FF	
	FF70 to FF71	2 Bytes	FF	EE2DIVNVR
	FF7C	1 Byte	F0	EE2NVR array configuration (*See following note*)

Note that data bit 4 of EEnNVR and bit 7 of EEnDIVHNVR can only be programmed once and cannot be erased.

Memory map for Motorola 68HC908AZ32A

The 68HC908AZ32A has two arrays which can be selected or de-selected by using the *SET TYPE* function.

	Address	***Size***	***Blank***	***Description***
FLASH1:-	8000 to FDFF	32256 Bytes	FF	
	FF80	1 Byte	FF	Block Protect Registers
	FFCC to FFFF	52 Bytes	FF	User Vectors

	Address	***Size***	***Blank***	***Description***
EEPROM1:-	0800 to 09FF	512 Bytes	FF	
	FE10 to FE11	2 Bytes	FF	EE1DIVNVR
	FE1C	1 Byte	FF	EE1NVR array configuration (See following note)

Note that data bit 4 of EEnNVR and bit 7 of EEnDIVHNVR can only be programmed once and cannot be erased.

Memory map for Motorola 68HC908AZ60

The 68HC908AZ60 has four arrays which can be selected or de-selected by using the *SET TYPE* function.

	Address	***Size***	***Blank***	***Description***
FLASH1:-	8000 to FDFF	32256 Bytes	00	
	FF80 to FF81	2 Bytes	00	Block Protect Registers
	FFCC to FFFF	52 Bytes	00	User Vectors
FLASH2:-	0450 to 04FF	176 Bytes	00	
	0580 to 05FF	128 Bytes	00	
	0E00 to 7FFF	29184 Bytes	00	
EEPROM1:-	0800 to 09FF	512 Bytes	FF	
	FE1C to FE1C	1 Byte	70	EENVR1 array configuration (*See following note*)
EEPROM2:-	0600 to 07FF	512 Bytes	FF	
	FE18 to FE18	1 Byte	70	EENVR2 array configuration (*See following note*)

Note that data bit 4 of both EENVR1 and EENVR2 can only be programmed once and it cannot be erased.

Memory map for Motorola 68HC908AZ60A

The 68HC908AZ60A has four arrays which can be selected or de-selected by using the *SET TYPE* function.

	Address	*Size*	*Blank*	*Description*
FLASH1:-	8000 to FDFF	32256 Bytes	FF	
	FF80 to FF81	2 Bytes	FF	Block Protect Registers
	FFCC to FFFF	52 Bytes	FF	User Vectors

	Address	*Size*	*Blank*	*Description*
FLASH2:-	0450 to 04FF	176 Bytes	FF	
	0580 to 05FF	128 Bytes	FF	
	0E00 to 7FFF	29184 Bytes	FF	
EEPROM1:-	0800 to 09FF	512 Bytes	FF	
	FE10 to FE11	2 Bytes	FF	EE1DIVNVR
	FE1C	1 Byte	F0	EE1NVR array configuration (*See following note*)
EEPROM2:-	0600 to 07FF	512 Bytes	FF	
	FF70 to FF71	2 Bytes	FF	EE2DIVNVR
	FF7C	1 Byte	F0	EE2NVR array configuration (*See following note*)

Note that data bit 4 of EEnNVR and bit 7 of EEnDIVHNVR can only be programmed once and cannot be erased.

Memory map for Motorola 68HC908GR4

The 68HC908GR4 has one array which is always selected.

	Address	*Size*	*Blank*	*Description*
FLASH1:-	EE00 to FDFF	4096 Bytes	FF	
	FF7E	1 Byte	FF	Block Protect Register
	FFDC to FFFF	36 Bytes	FF	User Vectors

Memory map for Motorola 68HC908GR8

The 68HC908GR8 has one array which is always selected.

	Address	*Size*	*Blank*	*Description*
FLASH1:-	E000 to FDFF	7680 Bytes	FF	
	FF7E	1 Byte	FF	Block Protect Register
	FFDC to FFFF	36 Bytes	FF	User Vectors

Memory map for Motorola 68H(R)C908JK3(E) & 68H(R)C908JL3(E)

The 68H(R)C908JL3(E)/JK3(E) has one array which is always selected.

	Address	*Size*	*Blank*	*Description*
FLASH1:-	EC00 to FBFF	4096 Bytes	FF	
	FFD0 to FFFF	48 Bytes	FF	User Vectors

Memory map for Motorola 68HC908JK8

The 68HC908JK8 has one array which is always selected.

	Address	*Size*	*Blank*	*Description*
FLASH:-	DC00 to FBFF	8192 Bytes	FF	
	FFCF	1 Byte	FF	Block Protect Register
	FFD0	1 Byte	FF *	Mask Option Register MOR
	FFDC to FFFF	36 Bytes	FF	User Vectors

* ***Note that*** when the OSCSEL bit in the Mask Option Register (MOR) is programmed, the *Blank Check* function on the programmer does not recognise a device in a socket. The device has to be erased before programming again.

14.26. PL740 Module For Motorola Micro 68HC705KJ1

- Only gang programming is allowed - no set programming.
- Data is read and written into RAM according to the memory map of each device as can be seen in the following table.

- Data is programmed from RAM to the corresponding eprom address in the microcontroller (except for the EPMSEC bit in the MOR Register). As there are gaps in the eprom map, ***note that*** the program checksum may not correspond to the RAM checksum.
- The user must select '*Program Lock bits = 1*' to secure the device. This will program the EPMSEC bit in the MOR Register.

- ***Note that*** the MOR (except the EPMSEC bit) Register will be programmed from RAM and must, therefore, be set up correctly before programming the device. A safe way to handle these Registers is to read a blank device into RAM before downloading the required data file to RAM.

Memory map for Motorola 68HC705KJ1

Memory map	*Address*	*Size*
EPROM	0300 to 07CF	1232 bytes
EPROM	07F1 to 07F1	1 byte MOR
EPROM	07F8 to 07FF	8 bytes

14.27. PL306 & PL310 Modules For 93CXX/24CXX Eeproms 'In Circuit'

- The programmer will work normally if the *PROGRAM* button is pressed.

- For a pin test, either press a sequence of buttons or send a remote control command which is the preferred method. For remote control, send BDT (C/R) and optionally L/F where C/R is carriage return and L/F is linefeed. For these instructions to work, a PL306 or PL310 module must be fitted with a device programmable on the relevant module.

- The programmer will respond with a '+' if no errors are found or a '-' if errors are found. The LCD will display:-

Bytes 1=0000000000000000
Bytes 2=0000000000000000

The first 00 after the '=' refers to the 'leftmost' socket of the left-hand module and the 'rightmost' 00 refers to the right-hand socket of the right-hand module. If only one module is fitted, 00000000 appears on both lines for the missing module.

The meaning of Byte 1 and 2 is explained later in this section. If both bytes are 00, there is no error.

- If the programmer responds with a '-', then the LCD can be read in the usual way by sending '*/(C/R)*' to the programmer.

- ♦ Devices failing the pin test will have flashing lights and devices passing the pin test will have permanently illuminated lights. By sending '*L (C/R)*' to the programmer, the state of the lights can be read.

- ♦ The pin test sequence can be started manually by pressing button B and then button D followed by *SET TYPE*.

NOTES:

- It is possible to join together remote commands into a string. For example, in order to perform the pin test, program devices in Production Mode and read back the lights, a single string is required – '*BDTPL (C/R)*'. If an error is found during the pin test or programming, a '-' will be received, otherwise a '+' will be sent back. If a '-' is sent to the programmer, the exact cause of the error will be displayed on the LCD which can be read in the normal manner.

- Each remote control command must be followed by a C/R to start execution. A byte other than C/R can be selected by using Special Function 9 (button B followed by button 9).

PL306 & PL310 Pin Test Procedure

ACTION	***TEST***
Pull Dout low Do ***NOT*** apply Vcc Set all outputs low Let GND float	Test GND level for 1 = bad GND contact or no device Abort and set bit 0 of error word Byte 2 of error word = 0 This test is used for device insertion.
Pull Dout high Do ***NOT*** apply Vcc Set all outputs high Let GND float	Test GND level for 1 = bad VCC contact Abort and set bit 6 of error word Byte 2 of error word = 0
Apply GND Pull Dout hi Set all outputs hi Apply Vcc	Test VCC level for 0 = Short on Vcc pin Abort and set bit 1 of error word Byte 2 of error word = 0
Let GND float Apply Vcc Pull Dout hi All outputs hi	Test outputs for 0 = Driver fault

	Abort and set bit 2 of error word Bits set in byte 2 for error pins (*See following error mask*)
ACTION	***TEST***
Apply GND Apply Vcc Pull Dout hi Set all outputs hi	Test outputs for 0 = Short to GND Abort and set bit 3 of error word Bits set in byte 2 for error pins (*See following error mask*)
Apply GND Apply Vcc Pull Dout Hi Walking 0 test	All outputs high = Probable short More than 1 output 0 = Probable short
Apply GND Apply Vcc Pull Dout lo Walking 1 test	All outputs low = Probable short More than 1 output 1 = Probable short Combine errors from walking 1 and 0 tests and report then abort Set bit 4 of error word Bits set in byte 2 for error pins (*See following error mask*)
Let GND float Apply Vcc All outputs except one hi Dout pulled hi	Test GND for 1 = Open circuit
Repeat for other outputs and Dout	Combine all open circuit errors and set bit 5 of error word Bits set in byte 2 for open circuit pins (*See following error mask*)

NOTES:

- The term – *abort* - means that no further tests will be applied to the socket being tested.

- These tests are applied to each socket in turn. ***Note that*** even a short on the power supply will not stop the other sockets from being used.

- PL306
  The output signals are CLK, Data in, Chip select (CS) and pins 6 and 7 which are not used for most devices.

- PL310
  The output signals are clock (SCL), Data and Write Protect (WP) except for Smart Card devices which do not have a WP pin.

- The walking 1 test applies a 1 to each output in turn while keeping the other outputs at 0. This test must be applied eight times as a short may cause oscillation. No failures are acceptable.

- The walking 0 test applies a 0 to each output in turn while keeping the other outputs at 1. Apply test eight times as above.

- The walking 1 and 0 tests include Data out which can be pulled hi or lo through a high value resistor.

- The pin test will fail if there is an empty socket.

- The error word consists of two bytes. If the first byte is 0, there is no error. The first byte contains the error code and the second byte may contain extra data depending on the nature of the error. If the second byte is not relevant, then it will be set to 0.

PL306 Pin Connections

- The module has four IDC sockets with long latches. Ribbon cables should be fitted with clamps.

- Every other connection is a 'screen'. These connections should be left unconnected at the programming end.

- Depending on the length and capacitance of the cable, it may be necessary to wire a resistor of about 150R in series with the SDO pin. This resistor must be fitted near the device being programmed. It is also recommended that a 100nF capacitor is fitted across chip ground and chip Vcc near the chip.

FUNCTION	*IDC Pin No.*	*93CXX Pin No.*	*Error Mask*
Chip select	1	1	01
Serial clock	3	2	02
Data in	5	3	04
Data out	7	4	08
Chip ground	9	5	10
Not defined	11	6	20
Not defined	13	7	40
Chip Vcc	15	8	80

- All even numbered pins on the IDC connector are connected to ground. It is essential that these are not connected to the chip ground.

PL310 Pin Connections

- The module has four IDC sockets with long latches. Ribbon cables should be fitted with clamps.

- Pins 6, 8, 10, 12, 14 and 16 are chip ground. These connections should be connected together at the chip end. On Mk2 versions of the module (***note that*** Mk2 is printed on the PCB track), pins 2 and 4 are connected to true ground with four wire links - LK1 to LK4 - on the component side. It is recommended that a 100nF capacitor is connected between true ground and chip Vcc of each circuit. If Mk2 versions are used for earlier applications, pins 2 and 4 can be isolated by removing the links.

- Depending on the length and capacitance of the cable, it may be necessary to wire a resistor of about 56R in series with the serial data pin near the device.

FUNCTION	***IDC Pin No.***	***24CXX Pin No.***			***Error Mask***
		8 pin	***5 pin***	***Smart card***	
A0	1	1	-	-	20
A1	3	2	-	-	01
A2	5	3	-	-	40
Serial data	9	5	3	C7	04
Serial clock	11	6	1	C3	02
Write protect	13	7	5	-	08
Chip Vcc	15	8	4	C1	80
Chip ground	6,8,10,12,14 & 16	4	2	C5	10

- It is essential that the chip ground for one part is not connected to the chip ground for another part, etc.

- Currently, the Vcc for all devices programmed by the PL310 module is 5V while programming. The devices are verified at a few per cent over their lowest operating voltage. The PL310 module supports devices operating down to 1.8V.

- This release of software has been designed for parts which have not been fitted with address pins A0, A1 and A2. It would, however, be possible to provide diagnostic tests for devices with A0, A1 and A2 connected at a later date.

14.28. PL480 Module For Holtek Micros

- Devices can only be gang programmed.

- The security/lock bit can be automatically programmed in the device. This option must be selected when using the *SET TYPE* function.

- The M9000 can secure a device by programming a bit in the options array. When an attempt is made to read a secured device, blank data is read from the data array into ram, the socket LED flashes and the display indicates that the device is locked to warn the user the data is not valid.

- For devices with a word size of less than 16 bits, you must preset the unused RAM bits to zero. This will be set automatically if a master device is read into RAM. If the master is downloaded from a file, either read a blank device into RAM or fill RAM with zero before downloading the file or make sure the file contains data for the whole device.

- The RAM location for the option words must contain the correct data for your device. (*See the following tables for the addresses of the option words.*)

Memory map for 48R06A

	Word Bits	***Addresses Code***	***Option***	***Lock bit Read only***
Device	14	0-03FF	8001-800F	8000
Ram	16	0-07FF	8002-801F	8000-8001

14.29. PL376 Module For Mitsubishi Micros Such As 30624FGA/M & R5F3640

- Devices can only be gang programmed.

- Some devices such as the R5F3640 have multiple arrays which can be selected or de-selected using the *SET TYPE* function.

- To speed up programming it is recommended to fill RAM with data FF before downloading the file.

- To download a file which is set up with the micro address range, the programmer must extract the data from the file and place it into the correct RAM address to program the device.

Memory map for 30624

Device address range	C0000 - FFFFE
RAM address range	00000 - 3FFFE

To down load the file set:-
Download from address = 0C0000
Download to address = 0FFFFF
RAM start address = 000000

Memory map for R5F3640

Array	*Device/RAM Address*
User 0	80000-FFFFF
User 1 (Program 2 ROM)	10000-13FFF
Data A	0E000-0EFFF
Data B	0F000-0FFFF

To download the file set:-

Download from address	=	00000000
Download to address	=	000FFFFF
RAM start address	=	00000000

14.30. PL665 Module For Programming 'In Circuit' PICs

- It is recommended that all assemblies to be programmed by this module are designed to comply with Microchip's recommendations for 'in-circuit' programming as published on their web site.

- This module can gang program up to four OTP or flash PICs which use Microchip's serial programming method, such part numbers which typically begin 16C, 16F and 18F.

- It is recommended that boards are tested ***BEFORE*** programming because all circuits use the same power supplies for Vcc and Vpp. A short, therefore, on any power supply will cause any function to abort. Clock and data lines are isolated from other circuits.

- Most programming operations are performed at 5.0V while verification is performed at Microchip's recommended levels for each device. This can mean that Vcc can be as high as 5.5V in the case of 'non flash' devices and 5V for flash parts or as low as 2.2V. ***Note that*** Vpp can be about 13V. It is, therefore, important to ensure that other circuitry on the assembly will not be damaged by these voltages and that there are no constraints with regard to applying these voltages. 'Shunt' regulator circuits for Vcc will, for example, cause the programmer to supply excessive amounts of power when Vcc is 5.5V. However, a small series resistor could overcome this problem at the expense of reducing the high verify voltage.

- When selecting the device type, the user can specify if devices are to be secured if this facility is available. Assuming that **ALL** devices program, the programmer will

automatically secure all devices after programming. However, if one or more devices fail, the programmer will display the message – '*Verify fail, none secured*'. The failing device(s) will be indicated by flashing green LEDs and good devices by continuously illuminated green LEDs. If the programmer is changed to R&D mode, the good devices can just be reprogrammed and secured. It is also suggested that a second attempt be made to reprogram the failing device.

- It is recommended that the programmer is used in Production mode which will ensure that 'non flash'devices are verified at both high and low Vcc limits. Flash parts are verified once at nominal Vcc. In Production mode, devices must be blank.

- With flash devices, the verify low can be turned off using Special Function F. (Press button B for Special Functions.)

- After power is applied, the PIC is put into programming mode by raising Vpp to about 13V with a rise time of typically less than 1uS and without the normal oscillator working. Unless both conditions are met, programming will fail. Capacitors on the Vpp line could cause this condition to fail.

- As RB6 and RB7 are used for programming, the impedance presented to the programmer should be as high as possible. The output impedance of the programmer is:-

	Clock	***Data***
Logic 0	56R	560R
Logic 1	620R	1K1

- If the target board has a resistor above 50R in the data line, the PL665 may not function because there is a pull up circuit in the PL665. These circuits are fitted to counteract any pull down elements in the user circuit. If there is nothing pulling down the data line, the effect of the series resistor on the target board can be reduced by changing the value of R8, R9, R18 and R20. The suggested value of the resistors is ten times the value of the series resistor on the target board. The effect of changing this resistor is to make the programmer more susceptible to noise. The resistors are conventional 'through hole' components.

- It is unlikely that the programmer will work if there are any capacitors connected to these pins.

- The programmer supplies a relay contact closure for each circuit when Vcc is applied. The voltage applied should not exceed 50V above ground and the current is limited to a maximum of 1A assuming a resistive load.

- The recommended method of controlling the programmer is over the RS232 port.

- Individual PICs can be programmed by other modules such as the PL651 upwards. Details about various PICs are included in other sections of this manual.

- Each module can program up to four circuits assuming that the maximum current from Vcc and Vpp is not exceeded. Capacitors connected to these supplies might

reduce the number of circuits which can be programmed. Up to about 90mA per circuit is available for Vcc and about 50mA for Vpp. The only simple answer is to experiment. Two modules can be fitted if required.

- The sockets on each module are designed to accept 16 way IDC ribbon connectors with strain relief clips fitted. The wiring follows the standard convention with pin 1 being marked.

LV	1	2	GND
Data	3	4	GND
Vcc	5	6	GND
Vcc	7	8	GND
Vpp	9	10	GND
Clk	11	12	GND
Relay	13	14	GND
Relay	15	16	GND

The ribbon cables between the programmer and the assemblies being programmed should be kept as short as possible - under 20 cms is suggested.

The PL665 hardware has been designed so that low voltage programming can be introduced at a later date. In this case, the 'LV' would be needed.

14.31. PL997 Memory Module

- ***In order to use the PL997 with 2 files (one for each eprom)***, use the *SET TYPE* button or PC software to select the device type to an 8M device such as 27C801 SGS _Thomson with 2 '*Devices per set*' and 8 '*Bits per word*'.

  Download the file for the low order bytes with the following set up:-

Load data from	00000000
Load data to	000FFFFF
Ram start	00000000

  Download the file for the high order bytes with the following set up:-

Load data from	00000000
Load data to	000FFFFF
Ram start	00100000

- ***In order to use the PL997 with one 16 bit data file***, use the *SET TYPE* button or PC software to select the device type to an 8M device such as 27C801 SGS_Thomson with 2 '*Devices per set*' and 16 'B*its per word*'.

  Download the file with the following set up:-

Load data from	00000000
Load data to	001FFFFF
Ram start	00000000

- The programmer treats each PL997 module as two separate devices and will add two to the total devices Pass/Fail count for each module inserted. It will also light or flash one LED for each device in the module. The pair of LEDs must both be on at the end of the programming cycle to indicate that the module has passed.

- The right LED of each pair represents the eprom with the low order data and the left LED represents the eprom with the high order data.

- The checksum shown after programming or verification is the checksum for both devices. In order to display individual checksums, use the < *STEP* > keys. (This is done automatically when using the PC software.)

14.32. PL227 MODULE For 'In Circuit' Programming Atmel AVR Family

- The user must select which parts of the device are to be programmed. There are three parts which can be programmed, namely the Flash, Eeprom and Option Register. ***Note that*** at least one part must be selected!

- Data is read and written to RAM only if the user selects the relevant part. If the Flash, for example, is selected, only that part will be read into RAM and only the Flash will be programmed.

- When the Flash or Eeprom is erased, an automatic erase of both the arrays is performed by the device even if only one array has been selected.

- The Eeprom can be reprogrammed without effecting the Flash or Option bytes if only the Eeprom is selected.

- Data is read and written into RAM according to the memory map of each device.

- The user must select the option – Fuse bits and the Boot lock bits - by entering them in RAM as per the following memory map. The fuse bits are not changed by the *Erase* Function but the Boot lock bits are erased.

- Atmel has not allocated a unique address space for the Flash and Eeprom arrays. The Atmel compiler generates two files - .HEX for Flash and .EEP for Eeprom. The following memory maps have, therefore, been implemented for these devices.

- The blank check does not test the state of the fuse bits as they can be supplied pre-programmed and they can also be programmed high or low in the program pass.

- In R&D mode, an automatic erase of the Flash, Eeprom and Lock bits will occur if an attempt is made to program the Flash.

- If it is required to make the devices unreadable, the user must select to program the lock bits during the device selection.

- When using set programming, the first device will be in the right-hand position of the right module (socket 1) and will correspond to the following memory map. Socket 8 is the left-hand position of the left-hand module. These positions apply even if no module has been fitted.

Each subsequent device in the set will be offset by RAM address 8000 Hex as follows:-

Device No. in Set	**MEGA8_ICP** *RAM Address*	**TINY2313_ICP** *RAM Address*
1	00000 - 02202	00000 - 00883
2	08000 - 0A202	08000 - 08883
3	10000 - 12202	10000 - 10883
4	18000 - 1A202	18000 - 18883
5	20000 - 22202	20000 - 20883
6	28000 - 2A202	28000 - 28883
7	30000 - 32202	30000 - 30883
8	38000 - 3A202	38000 - 38883

Memory map for Atmel MEGA8

		Address Range	
Type	***Device***	***RAM***	***Type***
MEGA8	0000 to 0FFF (16 bit)	0000 to 1FFF	Flash
	0000 to 01FF (8 bit)	2000 to 21FF	Eeprom
	No address	2200 to 2201	Fuse bits
	No address	2202	Boot lock Bits

Ram 2200 Fuse low bits (0 = programmed, 1 = erased)
Bit 0: CKSEL0
Bit 1: CKSEL1
Bit 2: CKSEL2
Bit 3: CKSEL3
Bit 4: SUTO
Bit 5: SUT1
Bit 6: BODEN
Bit 7: BODLEVEL

Ram 2201 Fuse high bits (0 = programmed, 1 = erased)
Bit 0: BOOTRST
Bit 1: BOOTSZ0
Bit 2: BOOTSZ1
Bit 3: EESAVE
Bit 4: CKOPT
Bit 5: SPIEN
Bit 6: WDTON
Bit 7: RSTDISBL

Ram 2202 Lock bits (0 = programmed, 1 = erased)
Bit 0: Lock bit 1 ++ Read only, use Set Type to program
Bit 1: Lock bit 2 ++ Read only, use Set Type to program
Bit 2: Boot lock bit 01
Bit 3: Boot lock bit 02
Bit 4: Boot lock bit 11
Bit 5: Boot lock bit 12

Memory map for Atmel TINY2313

	Address Range		
Type	***Device***	***RAM***	***Type***
TINY2313	0000 to 03FF (16 bit)	0000 to 07FF	FLASH
	0000 to 007F (8 bit)	0800 to 087F	EEPROM
	No address	0880 to 0882	Fuse Bits
	No address	0883 (Read only)	Lock Bits

Ram 0880 Fuse low bits (0 = programmed, 1 = erased)
Bit 0: CKSEL0
Bit 1: CKSEL1
Bit 2: CKSEL2
Bit 3: CKSEL3
Bit 4: SUTO
Bit 5: SUT1
Bit 6: CKOUT
Bit 7: CKDIV8

Ram 0881 Fuse high bits (0 = programmed, 1 = erased)
Bit 0: RSTDISBL
Bit 1: BODLEVEL0
Bit 2: BODLEVEL1
Bit 3: BODLEVEL2
Bit 4: WDTON
Bit 5: SPIEN
Bit 6: EESAVE
Bit 7: DWEN

Ram 0882 Fuse extended bits (0 = programmed, 1 = erased)
Bit 0: SELFPRGEN
Bit 1: 1
Bit 2: 1
Bit 3: 1
Bit 4: 1
Bit 5: 1
Bit 6: 1
Bit 7: 1

Ram 0883 Lock bits (0 = programmed, 1 = erased)

Bit 0: Lock bit 1	++ Read only, use Set Type to program
Bit 1: Lock bit 2	++ Read only, use Set Type to program
Bit 2: 1	
Bit 3: 1	
Bit 4: 1	
Bit 5: 1	
Bit 6: 1	
Bit 7: 1	

PL227 IDC connections

There are four IDC male headers fitted to each module. The female mating sockets should be fitted with strain relief clips to ensure correct operation of the eject mechanism.

Pin 1 is indicated on the socket by an arrow and a '1' on the cover.
Pin 2 is to the right of pin 1.
Pin 3 is below pin 1, etc.
All even numbered pins are connected to 0V.

The remaining functions are:-

1. Reset	7. Vcc
3. Mosi (Data input to Atmel micro)	9. Leave unconnected
5. Mosi (Data output from Atmel micro)	11. Clock

14.33. PL990 Memory Board

- The PL990 module must ***NOT*** be fitted or removed from the programmer whilst it is switched on.
- For the eprom memory board, the device type should be 27C256 AMD, 27C256R Atmel or 27C256 Atmel. Use the *SET TYPE* button or PC software to select the device type with 2 '*Devices per set*' and 16 '*Bits per word*'.
- For the Eeprom memory board, the device type should be 28C256 Atmel, 28C256 Xicor or 28256 Xicor. Use the *SET TYPE* button or PC software to select the device type with 2 '*Devices per set*' and 16 '*Bits per word*'.
- The programmer treats each module as a pair of (e)eproms - the first being programmed with the low bytes and the second with the high bytes. Two is, therefore, added to the total devices' pass/fail count for each module programmed. Both green LEDs must be on at the end of the programming cycle to indicate that the module has passed.
- The right-hand green LED indicates success or failure for the low bytes and the left-hand LED indicates success or failure for the high bytes.
- After programming using the PC software, the checksums of the low and high bytes are displayed. If the total checksum is required, an extra verify can be performed.

Alternatively, the program function can be started by pressing the Program button on the M9000 even if the programmer is being controlled by the PC software.

- After manually starting the *Program* function, the programmer will blank check, program and verify the (e)eproms, and then display the total checksum. Individual checksums can be obtained by using the *STEP* keys.

- Data can be uploaded to a PC's hard disc or downloaded from a hard disc using the PC software.
  Upload the file with the following set up:-

Load data from	00000000
Load data to	0005FFFF
RAM start	00000000

Download the file with the following set up:-

Load data from	00000000
Load data to	0005FFFF
RAM start	00000000

14.34. PL630 & PL631 Modules For ST/SGS Thomson ST7FLITE Family

- Devices can only be gang programmed.

- Data is read and written into RAM according to the memory map of each device as can be seen in the following example.

- The Option byte(s) will always be programmed and verified from the memory map location below with the exception of the lock bits which are selected separately.

- When the security bit is programmed in the device, further attempts to read the device will display – '*Locked @ checksum*' - and zero data will be loaded into RAM.

- Some devices have a slave copy of the option byte(s). Each bit of the slave is inverted. Normally, the user is not concerned about the slave. However, in case of difficulty, the slave is also read into the M9000 RAM as can be seen in the following example. The slave byte is not counted in the checksum but the unprotected state of the option byte(s) is.

- Unique address space for the option array has not been allocated by SGS_Thomson. Therefore, the following memory maps have been implemented for these devices.

- The RCCR calibration area can be allocated to RCCR or CODE.

RCCR calibration = No	allocates it to code
RCCR calibration = Yes	allocates it to RCCR

If '*Yes*' is selected, the programmer will leave this area alone for all device functions except *Erase*. The *Erase* function will reset the RCCR calibration to the factory set original state.

- As the flash array is actually an EEPROM, there is no blank state. Only the read and write protection bits are checked by the *Blank Check* function. A protected device will be shown as programmed and will need to be erased before it can be programmed again.

Memory map for ST/SGS_Thomson ST7FLITE family

Memory map	*Address Range*	*Type*
ST7FLITEBC	FC00 to FFFF	FLASH
	10000	+ MASTER OPTION BYTE 1
	10001	MASTER OPTION BYTE 0
	[1002 to 1003] READ FUNCTION ONLY	SLAVE OPTION
ST7FLITE02	FA00 to FFDD, FFE0 to FFFF	FLASH
	FFDE to FFDF	RCCR CALIBRATION/FLASH
	10000	+ MASTER OPTION BYTE 1
	10001	MASTER OPTION BYTE 0
	[1002 to 1003] READ FUNCTION ONLY	SLAVE OPTION
ST7FLITE09	1000 to 1001	RCCR CALIBRATION/EEPROM
	1002 to 107F	EEPROM
	FA00 to FFDD, FFE0 to FFFF	FLASH
	FFDE to FFDF	RCCR CALIBRATION/FLASH
	10000	+ MASTER OPTION BYTE 1
	10001	MASTER OPTION BYTE 0
	[1002 to 1003] READ FUNCTION ONLY	SLAVE OPTION
ST7FLITE15	F000 to FFDD, FFE0 to FFFF	FLASH
	FFDE to FFDF	RCCR CALIBRATION/FLASH
	10000	+ MASTER OPTION BYTE 1
	10001	MASTER OPTION BYTE 0
	[1002 to 1003] READ FUNCTION ONLY	SLAVE OPTION
ST7FLIT15BF1	F000 to FFFF	FLASH
	10000	+ MASTER OPTION BYTE 1
	10001	MASTER OPTION BYTE 0
	[1002 to 1003] READ FUNCTION ONLY	SLAVE OPTION

ST7FLITE19	1000 to 1001	RCCR CALIBRATION/EEPROM
	1002 to 107F	EEPROM
	F000 to FFDD, FFE0 to FFFF	FLASH
	FFDE to FFDF	RCCR CALIBRATION/FLASH
	10000	+ MASTER OPTION BYTE 1
	10001	MASTER OPTION BYTE 0
	[1002 to 1003] READ FUNCTION ONLY	SLAVE OPTION

Memory map	***Address Range***	***Type***
ST7FLIT19BF1	1000 to 107F	EEPROM
	E000 to FFFF	FLASH
	10000	+ MASTER OPTION BYTE 1
	10001	MASTER OPTION BYTE 0
	[1002 to 1003] READ FUNCTION ONLY	SLAVE OPTION
ST7FLITE25	E000 to FFDD, FFE0 to FFFF	FLASH
	FFDE to FFDF	RCCR CALIBRATION/FLASH
	10000	+ MASTER OPTION BYTE 1
	10001	MASTER OPTION BYTE 0
	[1002 to 1003] READ FUNCTION ONLY	SLAVE OPTION
ST7FLITE29	1000 to 1001	RCCR CALIBRATION/EEPROM
	1002 to 107F	EEPROM
	E000 to FFDD, FFE0 to FFFF	FLASH
	FFDE to FFDF	RCCR CALIBRATION/FLASH
	10000	+ MASTER OPTION BYTE 1
	10001	MASTER OPTION BYTE 0
	[1002 to 1003] READ FUNCTION ONLY	SLAVE OPTION
ST7FLITE35	E000 to FFFF	FLASH
	10000	+ MASTER OPTION BYTE 1
	10001	MASTER OPTION BYTE 0
	[1002 to 1003] READ FUNCTION ONLY	SLAVE OPTION
ST7FLITE39	1000 to 10FF	EEPROM
	E000 to FFFF	FLASH
	10000	+ MASTER OPTION BYTE 1
	10001	MASTER OPTION BYTE 0
	[1002 to 1003] READ FUNCTION ONLY	SLAVE OPTION

+ The FMP_R (Read out protection) & FMP_W (Flash write protection) are ***NOT*** programmed from RAM. In order to program these two options, use the *SET TYPE* function.

  The FMP_R is programmed if '*Program lock bits*' is set to '*1*' whereas the FMP_W is programmed if '*Flash write protection*' is set to '*Yes*'.

Note that the *Read* function stores the Slave Option bytes into RAM for information only. Both the *Verify* and *Program* functions action the complement of the Master Option bytes.

14.35. PL521 MODULE For Renesas Micro R5F2127x Family

- Devices can only be gang programmed.
- The user must select which parts of the device are to be programmed. There are four parts which can be programmed, namely the User 0, User 1, Data A and Data B. ***Note that*** at least one part must be selected!
- Data is read and written to RAM only if the user selects the relevant part. If the User 0, for example, is selected, only that part will be read into RAM and only the User 0 will be programmed.
- Data is read and written into RAM according to the memory map of each device.
- If it is required to make the devices unreadable, the user must select to program the lock bits during the device selection. Locked devices cannot be erased.

Memory map for Renesas Micros R5F2127x

Device Type	*Block*	*Address Range*
R5F21272	Data A	2400 to 27FF
	Data B	2800 to 2BFF
	User 1	none
	User 0	E000 to FFFF
R5F21274	Data A	2400 to 27FF
	Data B	2800 to 2BFF
	User 1	none
	User 0	C000 to FFFF
R5F21275	Data A	2400 to 27FF
	Data B	2800 to 2BFF
	User 1	A000 to BFFF
	User 0	C000 to FFFF
R5F21275	Data A	2400 to 27FF
	Data B	2800 to 2BFF
	User 1	8000 to BFFF
	User 0	C000 to FFFF

14.36. PL522 MODULE For Renesas Micro R5F2L38x Family

- Devices can only be gang programmed.
- The user blocks are always programmed.
- The user must select which optional data blocks of the device are to be programmed. There are four optional parts which can be programmed, namely the Data A, Data B, Data C and Data D.
- Data is read and written into RAM according to the memory map of each device.
- If it is required to make the devices unreadable, the user must select to program the lock bits during the device selection. Locked devices cannot be erased.

Memory map for Renesas Micro R5F2L38x

Device Type	*Block*	*Address Range*
R5F2L387	User	4000 to FFFF
	Data A	3000 to 33FF
	Data B	3400 to 37FF
	Data C	3800 to 3BFF
	Data D	3C00 to 3FFF

15. APPENDIX

15.1. Password Levels

When the keyboard is locked, functions with a password level higher than the password level cannot be selected until the keyboard is unlocked.

Each user has his/her own password and also his/her corresponding password level.

Level 1:
- Program from RAM
- Verify with RAM
- Verify with port

Level 2:
- Set type
- Check master

Level 3:
- Set comms
- Read master into RAM
- Change mode
- Change language
- Complement RAM (1s complement)
- Download data
- Fill RAM
- Print RAM
- Program from port

Level 4:
- Change/edit RAM
- Find/replace character string
- Copy block
- Split and merge RAM

Level 5:
- Upload data

If, for example, the keyboard has been locked at Level 1, all Level 2, 3, 4 and 5 functions would be locked out. However, if the keyboard has been locked at Level 4, only Level 5 functions would be locked out.

Functions not mentioned, such as Checksum RAM, can always be used.

THE DEFAULT PASSWORD FOR ALL USERS IS 000. IF THE PASSWORD HAS BEEN FORGOTTEN, CONTACT THE MANUFACTURER OR YOUR DISTRIBUTOR.

15.2. Programming Parameters (Software Revision L/M9000 V5.41)

PLEASE ENQUIRE IF THE DEVICE YOU NEED IS NOT LISTED AS NEW DEVICES ARE ADDED EACH MONTH. (The latest device list is available for our programmers at www.lloyd-research.com)

Device	*Make*	*Int. Ident.*	*Programming Parameters* *Initial Pulse*	*Overprog. Multiplier*	*Vpp V*	*Comment*
- Size 0 - 0000000F	**PL308, PL309 or PL310 module**					
24C00	Microchip	No	4mS	x 0	5	1 byte page
- Size 0 - 0000007F	**PL308, PL309 or PL310 module**					
85C72	Microchip	No	10mS	x 0	5	2 byte page
24C01	Atmel	No	10mS	x 0	5	4 byte page
24C01	SGS_Thomson	No	10mS	x 0	5	4 byte page
24C01	ST	No	10mS	x 0	5	4 byte page
24C01A	Xicor	No	10mS	x 0	5	4 byte page
- Size 0 - 000000FF	**PL308, PL309 or PL310 module**					
24C02	Atmel	No	5mS	x 0	5	8 byte page
24C02_2_7	Atmel	No	5mS	x 0	5	8 byte page
24C02	Exel	No	10mS	x 0	5	4 byte page
24C02	Xicor	No	10mS	x 0	5	4 byte page
24C02SC	Atmel	No	5mS	x 0	5	8 byte page
24C02SC_2_7	Atmel	No	5mS	x 0	5	8 byte page
24X02A	Microchip	No	2mS	x 0	5	2 byte page
24C02A	SGS_Thomson	No	20mS	x 0	5	8 byte page
24C02A	ST	No	20mS	x 0	5	8 byte page
24C02C	SGS_Thomson	No	20mS	x 0	5	8 byte page
24C02C	ST	No	20mS	x 0	5	8 byte page
24LC02B	Microchip	No	20mS	x 0	5	8 byte page
8582	Philips	No	50mS	x 0	5	8 byte page
8582	Signetics	No	50mS	x 0	5	8 byte page
- Size 0 - 000007FF	**PL300 module**					
2004	Intel	No	10uS	x 0	5	
2004	Xicor	No	10uS	x 0	5	

- Size 0 - 000001FF PL308, PL309 or PL310 module

24AA04	Microchip	No	5mS	x 0	5	16 byte page
24AA04T_2_1	Microchip	No	5mS	x 0	5	16 byte page
24C04	Atmel	No	5mS	x 0	5	16 byte page
24C04_1_8	Atmel	No	5mS	x 0	5	16 byte page
24C04_2_1	Atmel	No	5mS	x 0	5	16 byte page
24C04_2_7	Atmel	No	5mS	x 0	5	16 byte page
24C04	Exel	No	10mS	x 0	5	16 byte page
24C04	National	No	10mS	x 0	5	16 byte page
24C04	Ramtron	No	1mS	x 0	5	256 byte page
24C04	SGS Thomson	No	10mS	x 0	5	16 byte page

Device	*Make*	*Int. Ident.*	*Programming Parameters*			*Comment*
			Initial Pulse	*Overprog. Multiplier*	*Vpp V*	

- Size 0 - 000001FF PL308, PL309 or PL310 module *(Cont.)*

24C04	ST	No	10mS	x 0	5	16 byte page
24C04_W	SGS Thomson	No	5mS	x 0	5	16 byte page
24C04	Xicor	No	10mS	x 0	5	16 byte page
24C04A	Atmel	No	5mS	x 0	5	16 byte page
24C04A_1_8	Atmel	No	5mS	x 0	5	16 byte page
24C04A_2_7	Atmel	No	5mS	x 0	5	16 byte page
24C04A	Microchip	No	8mS	x 0	5	8 byte page
24C04C	SGS Thomson	No	10mS	x 0	5	8 byte page
24C04C	ST	No	10mS	x 0	5	8 byte page

- Size 0 - 000001FF PL308, PL309 or PL310 module

24C04L	SGS Thomson	No	10mS	x 0	5	16 byte page
24C04L	ST	No	10mS	x 0	5	16 byte page
24C04R	SGS Thomson	No	10mS	x 0	5	16 byte page
24C04R	ST	No	10mS	x 0	5	16 byte page
24C04W	SGS Thomson	No	10mS	x 0	5	16 byte page
24C04W	ST	No	10mS	x 0	5	16 byte page
24C04L_W	SGS Thomson	No	5mS	x 0	5	16 byte page
24C04L_W	ST	No	5mS	x 0	5	16 byte page
24C04R_W	SGS Thomson	No	5mS	x 0	5	16 byte page
24C04R_W	ST	No	5mS	x 0	5	16 byte page
24C04W_W	SGS Thomson	No	5mS	x 0	5	16 byte page
24C04W_W	ST	No	5mS	x 0	5	16 byte page
24LC04B	Microchip	No	5mS	x 0	5	16 byte page

- Size 0 - 000003FF PL308, PL309 or PL310 module

24C08	Atmel	No	10mS	x 0	5	16 byte page
24C08	Catalyst	No	10mS	x 0	5	16 byte page
24C08	National	No	10mS	x 0	5	16 byte page
24C08	Xicor	No	10mS	x 0	5	16 byte page
24LC08B	Microchip	No	10mS	x 0	5	16 byte page

- Size 0 - 000007FF PL308, PL309 or PL310 module

24C16	Atmel	No	10mS	x 0	5	16 byte page
24C16	Exel	No	10mS	x 0	5	16 byte page
24C16	National	No	10mS	x 0	5	16 byte page
24C16W	SGS Thomson	No	5mS	x 0	5	16 byte page

24C16W	ST	No	5mS	x 0	5	16 byte page
24C16	Xicor	No	10mS	x 0	5	16 byte page
24C164	Atmel	No	10mS	x 0	5	16 byte page
24LC164	Microchip	No	10mS	x 0	5	16 byte page
24LC16B	Microchip	No	10mS	x 0	5	16 byte page

- Size 0 - 000007FF PL308 or PL309 module

24165	Xicor	No	10mS	x 0	5	32 byte page Write Protect

Device	*Make*	*Int. Ident.*	*Programming Parameters*			*Comment*
			Initial Pulse	*Overprog. Multiplier*	*Vpp V*	

- Size 0 - 00000FFF PL308, PL309 or PL310 module

24C32	Atmel	No	10mS	x 0	5	32 byte page
24C32	Microchip	No	50mS	x 0	5	64 byte page
24LC32	Microchip	No	50mS	x 0	5	64 byte page

- Size 0 - 00000FFF PL308A module ***(See Section 14.24.)***

25320	Atmel	No	5mS	x 0	5	32 byte page Write Protect

- Size 0 - 00001FFF PL308, PL309 or PL310 module

24C64	Atmel	No	10mS	x 0	5	32 byte page
24C64W	SGS Thomson	No	5mS	x 0	5	32 byte page
24C64W	ST	No	5mS	x 0	5	32 byte page
24C65	Microchip	No	10mS	x 0	5	8 byte page
24AA65	Microchip	No	50mS	x 0	5	64 byte page
24LC65	Microchip	No	10mS	x 0	5	8 byte page

- Size 0 - 00001FFF PL308A module ***(See Section 14.24.)***

25640A	Atmel	No	5mS	x 0	5	32 byte page WP
25LC640	Microchip	No	5mS	x 0	5	32 byte page WP
95640	SGS Thomson	No	5mS	x 0	5	32 byte page WP
95640	ST	No	5mS	x 0	5	32 byte page WP WP = Write Protect

- Size 0 - 00003FFF PL308, PL309 or PL310 module

24C128	Atmel	No	10mS	x 0	5	64 byte page

- **Size 0 - 00007FFF PL308, PL309 or PL310 module**

24256BW	SGS Thomson	No	10mS	x 0	5	64 byte page
24256BW	ST	No	10mS	x 0	5	64 byte page
24C256	Atmel	No	10mS	x 0	5	64 byte page
24LC256	Microchip	No	5mS	x 0	5	64 byte page

- Size 0 - 00007FFF PL308A module ***(See Section 14.24.)***

25C256	Catalyst	No	10mS	x 0	5	64 byte page

- Size 0 - 00007FFF PL308 or PL309 module

24LC512	Microchip	No	5mS	x 0	5	64 byte page

- Size 0 - 000007FF PL300 module

2716	Various	No	Single 50mS pulse	25	
2716B	AMD	Yes	1mS x 3	12.5	
27C16	National	No	Single 50mS pulse	25	
27C16	Various	No	Single 50mS pulse	25	
27C16B	National	Yes	100uS x 0	12.7	
27C292	Texas	Yes	100uS x 24	13.5	No blank check
281_16K	Greenwich	No	1uS x 0	5	Non-Volatile RAM
2816A	Various	No	10mS x 0	5	Eeprom
28C16A	Microchip	No	1mS x 0	5	Data Polling
2817A	Seeq	No	10mS x 0	5	Eeprom

Device	***Make***	***Int. Ident.***	***Programming Parameters***			***Comment***
			Initial Pulse	***Overprog. Multiplier***	***Vpp V***	

- Size 0 - 00000FFF PL300 module

2532	Texas	No	Single 50mS pulse	25	
2532A	Texas	No	Single 10mS pulse	21	
2732	Various	No	Single 50mS pulse	25	
2732A	Various	No	Single 50mS pulse	21	
2732B	AMD	Yes	1mS x 3	12.7	
27C32	National	No	Single 50mS pulse	25	
27C32	Various	No	Single 50mS pulse	25	
27C32B	National	Yes	100Us x 0	12.7	

- Size 0 - 00001FFF PL300, PL328 or PL460 module

2564	Texas	No	Single 50mS pulse	25	
2764	AMD	No	1mS x 4	21	
2764	Fujitsu	No	1mS x 1	21	
2764	Greenwich	No	20us x 0	12.7	Emulator
2764	Hitachi	No	1mS x 4	21	
2764	Intel	No	1mS x 4	21	
2764	Mitsubishi	No	1mS x 4	21	
2764	NEC	No	1mS x 4	21	
2764	Renesas	No	1mS x 4	21	
2764	Seeq	Yes	1mS x 4	21	
2764	Texas	No	Single 50mS pulse	21	
2764	Toshiba	No	1mS x 1	21	
2764A	AMD	Yes	1mS x 3	12.7	
2764A	Intel	Yes	1mS x 3	12.5	
2764A	SGS_Thomson	Yes	1mS x 3	12.5	
2764A	ST	Yes	1mS x 3	12.5	
27C64	AMD	Yes	100uS x 0	12.7	Flashrite
27C64_OTP	AMD	Yes	100uS x 0	12.7	Flashrite
27C64	Fujitsu	No	1mS x 1	21	
27C64	Gen Instr	Yes	100uS x 0	13.0	
27C64	Hitachi	No	1mS x 4	21	
27C64	Hyundai	Yes	1mS + 2	13.0	
27C64	Intel	Yes	100uS x 0	12.7	
27C64	Microchip	Yes	100uS x 0	13.0	
27C64	National	Yes	500uS x 0	12.7	
27C64	NEC	No	1mS x 4	21	
27C64	Renesas	No	1mS x 4	21	

27C64	Texas	Yes	100uS	x 0	13	TI Snap
27C64A	Philips	Yes	100uS	x 0	12.7	2 Pass Verify
27C64A	SGS_Thomson	Yes	1mS	x 3	12.5	
27C64A	ST	Yes	1mS	x 3	12.5	
27C64A	Signetics	Yes	100uS	x 0	12.7	2 Pass Verify
27C64F	Waferscale	No	1mS	x 1	13.5	
27C64L	Waferscale	No	200uS	+ 1mS	12.7	
27HC64	Atmel	Yes	1mS	x 3	12.5	
27HC64	Gen.Instr	Yes	100uS	x 0	13.0	

Device	*Make*	*Int. Ident.*	*Programming Parameters*			*Comment*
			Initial Pulse	*Overprog. Multiplier*	*Vpp V*	
- Size 0 - 00001FFF PL300, PL328 or PL460 module *(Cont.)*						
27HC64	Microchip	Yes	100uS	x 0	13.0	
27HC65	NEC	No	400uS	x 1	12.5	NEC approved
27PC64	Texas	Yes	100uS	x 0	13	TI Snap
P2764A	Intel	Yes	100uS	x 0	12.7	Quick Pulse
27F64	Intel	Yes	100uS	x 0	12.7	Flash
2864	Various	No	10mS	x 0	5.0	
2864	Seeq	No	10mS	x 0	5.0	
2864A	AMD	Yes	1mS	x 0	5.0	Data Polling
2864A	Xicor	No	1mS	x 0	5.0	Data Polling
2864B	AMD	Yes	1mS	x 0	5.0	Data Polling
2864B	Xicor	No	1mS	x 0	5.0	Data Polling
28C64	Atmel	No	1mS	x 0	5.0	Data Polling
28C64	Xicor	No	1mS	x 0	5.0	Data Polling
28C64_SDP	Xicor	No	1mS	x 0	5.0	Data Protection
28C64A	Microchip	No	1mS	x 0	5.0	Data Polling
28C64B	Catalyst	No	1mS	x 0	5.0	Data Polling
28C64B_SDP	Catalyst	No	1mS	x 0	5.0	Data Protection
28C64C	SGS_Thomson	No	1mS	x 0	5.0	Data Polling
28C64C	ST	No	1mS	x 0	5.0	Data Polling
28HC64	Xicor	No	1mS	x 0	5.0	Data Polling
28HC64_SDP	Xicor	No	1mS	x 0	5.0	Data Protection
48Z08_64K	SGS_Thomson	No	1uS	x 0	5.0	Non-Volatile RAM
48Z08_64K	ST	No	1uS	x 0	5.0	Non-Volatile RAM
48Z09_64K	SGS_Thomson	No	1uS	x 0	5.0	Non-Volatile RAM
48Z09_64K	ST	No	1uS	x 0	5.0	Non-Volatile RAM
5762	Sharp	No	1mS	x 3	12.5	
5763	Sharp	No	1mS	x 3	12.5	
5763	Sharp	No	100uS	x 0	12.7	
- Size 0 - 00001FFF PL300 module						
57C49B	Waferscale	No	1mS	x 1	13.5	
- Size 0 - 00001FFF PL300, PL328 or PL460 module						
57C64F	Waferscale	No	1mS	x 1	13.5	
87C64	Intel	Yes	1mS	x 3	12.5	
881_64K	Greenwich	No	1uS	x 0	5.0	Non-Volatile RAM
DS1213D	Dallas	No	1uS	x 0	5.0	Non-Volatile RAM

DS1216B	Dallas	No	1uS	x 0	5.0	Non-Volatile RAM
DS1225Y	Dallas	No	1uS	x 0	5.0	Non-Volatile RAM

- Size 0 - 00000FFF (16 bits) PL400, PL444 or PL445 module

57C65	Waferscale	Yes	1mS	x 1	13.5	

Device	*Make*	*Int. Ident.*	*Programming Parameters*			*Comment*
			Initial Pulse	*Overprog. Multiplier*	*Vpp V*	

- Size 0 - 00003FFF PL300, PL328 or PL460 module

27128	AMD	No	1mS	x 4	21	
27128	Fujitsu	Yes	1mS	x 1	21	
27128	Greenwich	No	20uS	x 0	12.7	Emulator
27128	Hitachi	No	1mS	x 4	21	
27128	Intel	Yes	1mS	x 4	21	
27128	Mitsubishi	No	1mS	x 4	21	
27128	NEC	No	1mS	x 4	21	
27128	Renesas	No	1mS	x 4	21	
27128	Seeq	Yes	1mS	x 4	21	
27128	Texas	No	1mS	x 4	21	
27128	Toshiba	No	1mS	x 1	21	
27128A	AMD	Yes	1mS	x 3	12.7	
27128A	Fujitsu	No	1ms	x 3	12.5	
27128A	Hitachi	No	1ms	x 3	12.5	
27128A	Intel	Yes	1ms	x 3	12.5	
27128A	Renesas	No	1ms	x 3	12.5	
27128A	SGS_Thomson	Yes	1mS	x 3	12.5	
27128A	ST	Yes	1mS	x 3	12.5	
27128A	Toshiba	Yes	1ms	x 3	12.5	
27C128	AMD	Yes	100uS	x 0	12.7	Flashrite
27C128_OTP	AMD	Yes	100uS	x 0	12.7	Flashrite
27C128	Atmel	No	1mS	x 3	12.5	
27C128	Fujitsu	Yes	1mS	x 1	21	
27C128	Gen Instr	Yes	100uS	x 0	13.0	
27C128	Hitachi	No	1mS	x 4	21	
27C128	Microchip	Yes	100uS	x 0	13.0	
27C128	National	Yes	100uS	x 0	12.7	
27C128	Renesas	No	1mS	x 4	21	
27C128	Texas	Yes	100uS	x 0	13	TI Snap
27C128B	National	Yes	100uS	x 0	12.7	
27C128F	Waferscale	No	1mS	x 1	13.5	
27C128L	Waferscale	No	200uS	+ 1mS	12.7	
27PC128	Texas	Yes	100uS	x 0	13	TI Snap
P27128A	Intel	Yes	100uS	x 0	12.7	Quick Pulse
57126	Sharp	No	1mS	x 3	12.5	
57127	Sharp	No	1mS	x 3	12.5	
57128	Sharp	No	100uS	x 0	12.7	

57C128F	Waferscale	No	1mS	x 1	13.5	

- Size 0 - 00007FFF PL300, PL328 or PL460 module

27256	AMD	Yes	1mS	x 3	12.7	
27256	Fujitsu	Yes	1mS	x 3	12.5	
27256	Greenwich	No	20uS	x 0	12.7	Emulator
27256	Hitachi	No	1mS	x 3	12.5	
27256	Intel	Yes	1mS	x 3	12.5	
27256	Mitsubishi	Yes	1mS	x 3	12.5	
27256	NEC	Yes	1mS	x 1	21	
27256	Renesas	No	1mS	x 3	12.5	

Device	***Make***	***Int. Ident.***	***Programming Parameters***			***Comment***
			Initial Pulse	***Overprog. Multiplier***	***Vpp V***	

- Size 0 - 00007FFF PL300, PL328 or PL460 module *(Cont.)*

27256	SGS_Thomson	Yes	1mS	x 3	12.5	
27256	ST	Yes	1mS	x 3	12.5	
27256	Texas	No	1mS	x 3	12.5	
27256	Toshiba	Yes	1mS	x 1	21	
27256A	Toshiba	Yes	1mS	x 3	12.5	
27C256	AMD	Yes	100uS	x 0	12.7	Flashrite
27C256_OTP	AMD	Yes	100uS	x 0	12.7	Flashrite
27C256	Atmel	Yes	1mS	x 3	12.5	
27C256	Cypress	Yes	200uS	x 2	12.5	
27C256	Fujitsu	Yes	1mS	x 1	21	
27C256	Gen Instr	Yes	100uS	x 0	13.0	
27C256	Hitachi	Yes	1mS	x 3	12.5	
27C256	Intel	Yes	100uS	x 0	12.7	(Die P624)
27C256_	Intel	Yes	100uS	x 0	12.7	(Die P629)
27C256	Microchip	Yes	100uS	x 0	13.0	
27C256	Mitsubishi	Yes	1mS	x 3	12.5	
27C256	MXIC	Yes	100uS	x 0	12.7	
27C256	National	No	500uS	x 0	12.7	
27C256	NEC	No	1mS	x 1	21	
27C256	Philips	Yes	100uS	x 0	12.7	2 Pass Verify
27C256	Renesas	Yes	1mS	x 3	12.5	
27C256	Seeq	Yes	500uS	x 3	12.5	
27C256	SGS_Thomson	No	1mS	x 3	12.5	
27C256	ST	No	1mS	x 3	12.5	
27C256	Signetics	Yes	100uS	x 0	12.7	2 Pass Verify
27C256	Texas	Yes	100uS	x 0	13	TI Snap
27C256A	Cypress	Yes	100uS	x 0	12.7	
27C256A	Fujitsu	Yes	1mS	x 3	12.5	
27C256A	Mitsubishi	Yes	1mS	x 3	12.5	
27C256A	NEC	Yes	1mS	x 1	12.5	
27C256A	Renesas	Yes	1mS	x 3	12.5	
27C256AFP	Hitachi	Yes	200uS	x 1	12.5	OTP
27C256AFP	Renesas	Yes	200uS	x 1	12.5	OTP
27C256AG	Hitachi	Yes	200uS	x 1	12.5	
27C256AG	Renesas	Yes	200uS	x 1	12.5	
27C256B	National	Yes	100uS	x 0	12.7	
27C256B	SGS_Thomson	Yes	100uS	x 0	12.7	*

27C256B	ST	Yes	100uS	x 0	12.7	*
27C256F	Waferscale	No	1mS	x 1	13.5	
27C256HG	Hitachi	Yes	200uS	x 1	12.5	
27C256HG	Renesas	Yes	200uS	x 1	12.5	
27C256L	Waferscale	No	200uS	+ 1mS	12.7	
27C256R	Atmel	Yes	100uS	+ A/R	13.0	
27HC256	Atmel	Yes	1mS	x 3	12.5	
27HC256	Gen Instr	Yes	100uS	x 0	13.0	
27HC256	Microchip	Yes	100uS	x 0	13.0	
27HC256R	Atmel	Yes	100uS	+ A/R	13.0	
27LV256	Gen Instr	Yes	100uS	x 0	13.0	

Device	***Make***	***Int. Ident.***	***Programming Parameters***			***Comment***
			Initial Pulse	***Overprog. Multiplier***	***Vpp V***	
- Size 0 - 00007FFF PL300, PL328 or PL460 module *(Cont.)*						
27LV256	Microchip	Yes	100uS	x 0	13.0	
27PC256	Texas	Yes	100uS	x 0	13	TI Snap
P27256	Intel	Yes	100uS	x 0	12.7	Quick Pulse
27E257	Winbond	Yes	100uS	x 0	12.0	Flash
27F256	Intel	Yes	100uS	x 0	12.7	Flash
27SF256	SST	Yes	25uS	x 0	12.0	Flash
28256	Xicor	No	1mS	x 0	5.0	Data Polling
28C256	Atmel	No	1mS	x 0	5.0	Data Polling
28C256	Xicor	No	1mS	x 0	5.0	Data Polling
				(* See programming parameter notes)		
- Size 0 - 00007FFF PL232, PL300, PL332, PL333 or PL450 module						
28F256	AMD	Yes	10uS	x 0	12.0	Flash
28F256	Intel	Yes	100uS	x 0	12.0	Flash
28F256	SGS_Thomson	Yes	100uS	x 0	12.0	Flash
28F256	ST	Yes	100uS	x 0	12.0	Flash
28F256A	AMD	Yes	10uS	x 0	12.0	" Data Polling
28F256A	Intel	Yes	10uS	x 0	12.0	Flash
28F256A	SGS_Thomson	Yes	10uS	x 0	12.0	Flash
28F256A	ST	Yes	10uS	x 0	12.0	Flash
29C257	Atmel	Yes	1mS	x 0	5.0	) Page write only
29C257_SDP	Atmel	Yes	1mS	x 0	5.0	) Max set size = 2
		_SDP for Data Protection				) (***See notes***)
29C256	Atmel	Yes	1mS	x 0	5.0	) Page write only
29C256_SDP	Atmel	Yes	1mS	x 0	5.0	) Max set size = 2
29LV256A	Atmel	Yes	1mS	x 0	5.0	) Page write only
29LV256A_SDP	Atmel	Yes	1mS	x 0	5.0	) Max set size = 2
		_SDP for Data Protection				) (***See notes***)
CY7C271A	Cypress	No	100uS	x 0	12.7	
DS1230Y_256	Dallas	No	1uS	x 0	5.0	Non-Volatile RAM
DS1235Y_256	Dallas	No	1uS	x 0	5.0	Non-Volatile RAM
3281_256K	Greenwich	No	1uS	x 0	5.0	Non-Volatile RAM
- Size 0 - 00007FFF PL300, PL328 or PL460 module						
57256	Sharp	No	100uS	x 0	12.7	
57256	Toshiba	Yes	1mS	x 1	21.0	
57256A	Toshiba	Yes	100uS	x 0	12.7	

57C256A	Waferscale	No	1mS	x 1	13.5	
58C256	Hitachi	No	1mS	x 0	5.0	Data Polling
58C256	Renesas	No	1mS	x 0	5.0	Data Polling
87C257	Intel	Yes	100uS	x 0	12.7	Quick Pulse
26C512A	MXIC	Yes	100uS	x 0	12.7	Flash

- Size 0 - 0000FFFF PL300, PL328 or PL460 module

27512	AMD	Yes	1mS	x 3	12.7	
27512	Greenwich	No	20uS	x 0	12.7	Emulator
27512	Hitachi	Yes	1mS	x 3	12.5	
27512	Intel	Yes	1mS	x 3	12.5	

Device	***Make***	***Int. Ident.***	***Programming Parameters***			***Comment***
			Initial Pulse	***Overprog. Multiplier***	***Vpp V***	

- Size 0 - 0000FFFF PL300, PL328 or PL460 module *(Cont.)*

27512	Mitsubishi	Yes	1mS	x 3	12.5	
27512	Renesas	Yes	1mS	x 3	12.5	
27512	SGS_Thomson	Yes	1mS	x 3	12.5	
27512	ST	Yes	1mS	x 3	12.5	
27512	Toshiba	Yes	1mS	x 3	12.5	
27512A	Toshiba	Yes	100uS	x 0	12.7	
27512P	Hitachi	Yes	1mS	x 3	12.5	OTP
27512P	Renesas	Yes	1mS	x 3	12.5	OTP
27C512	AMD	Yes	100uS	x 0	12.7	Flashrite
27C512_	OTP AMD	Yes	100uS	x 0	12.7	Flashrite
27C512	Atmel	Yes	1mS	x 3	12.5	
27C512	Fujitsu	Yes	1mS	x 3	12.5	
27C512_EXP	Gen Instr	Yes	100uS	x 0	13.0	
27C512_FAST	Gen Instr	Yes	1mS	x 3	12.5	
27C512	Holtek	No	75uS	x 0	12.2	
27C512	Intel	Yes	100uS	x 0	12.7	Quick Pulse
27C512_FAST	Microchip	Yes	1mS	x 3	12.5	
27C512_EXP	Microchip	Yes	100uS	x 0	13.0	
27C512	MXIC	Yes	100uS	x 0	12.7	
27C512	National	Yes	100uS	x 0	12.7	
27C512	NEC	No	100uS	x 0	12.7	
27C512	Philips	Yes	100uS	x 0	12.7	2 Pass Verify
27C512	SGS_Thomson	Yes	100uS	x 0	12.7	*
27C512	ST	Yes	100uS	x 0	12.7	*
27C512	Signetics	Yes	100uS	x 0	12.7	2 Pass Verify
27C512	Texas	Yes	100uS	x 0	13	TI Snap

(**** See programming parameter notes***)

- Size 0 - 0000FFFF PL994 module

27C512	Oki	No	100uS	x 0	12.7	

- Size 0 - 0000FFFF PL300, PL328 or PL460 module

27C512A	Mitsubishi	Yes	1mS	x 3	12.5	
27C512A	National	Yes	100uS	x 0	12.7	
27C512A	Renesas	Yes	1mS	x 3	12.5	
27C512F	Waferscale	No	1mS	x 1	13.5	
27C512L	Waferscale	No	200uS	+ 1mS	12.7	

27C512R	Atmel	Yes	100uS	+ A/R	13.0	
27E512	Winbond	Yes	100uS	x 0	12.0	Flash
27HC512	ISSI	Yes	100uS	x 0	12.7	
27PC512	Texas	Yes	100uS	x 0	13.0	TI Snap
27SF512	SST	Yes	25uS	x 0	12.0	Flash
27V512	SGS_Thomson	Yes	100uS	x 0	12.7	*
27V512	ST	Yes	100uS	x 0	12.7	*
276308A	AMIC	Yes	100uS	x 0	12.7	
P27512	Intel	Yes	100uS	x 0	12.7	Quick Pulse

(**** See programming parameter notes***)

Device	***Make***	***Int. Ident.***	***Programming Parameters***			***Comment***
			Initial Pulse	***Overprog. Multiplier***	***Vpp V***	
- Size 0 - 0000FFFF PL232, PL300, PL332, PL333 or PL450 module						
28C512	Xicor	No	1mS	x 0	5.0	Data Polling
28F512	AMD	Yes	10uS	x 0	12.0	Flash
28F512	Intel	Yes	10uS	x 0	12.0	Flash
28F512	SGS_Thomson	Yes	10uS	x 0	12.0	Flash
28F512	ST	Yes	10uS	x 0	12.0	Flash
28F512A	AMD	Yes	10uS	x 0	12.0	" Data Polling
29C512	Atmel	Yes	1mS	x 0	5.0	) Page write only
29C512_SDP	Atmel	Yes	1mS	x 0	5.0	) Max set size = 2
29EE512	Greenliant	Yes	1mS	x 0	5.0	Page write only
29EE512_SDP	Greenliant	Yes	1mS	x 0	5.0	) Max set size = 2
		_SDP for Data Protection				) (***See notes***)
29EE512	SST	Yes	1mS	x 0	5.0	Page write only
29EE512_SDP	SST	Yes	1mS	x 0	5.0	) Max set size = 2
		_SDP for Data Protection				) (***See notes***)
- Size 0 - 0000FFFF PL300, PL328 or PL460 module						
57C512F	Waferscale	No	1mS	x 1	13.5	
- Size 0 - 00003FFF x 4 PL300, PL328 or PL460 module						
27513	Greenwich	No	20uS	x 0	12.7	Emulator
27513	Intel	Yes	1mS	x 3	12.5	4 Pages
27C513	Atmel	Yes	1mS	x 3	12.5	4 Pages
27C513	Intel	Yes	100uS	x 0	12.7	4 Pages
27C513R	Atmel	Yes	100uS	x 0	12.7	4 Pages
P27513	Intel	Yes	100uS	x 0	12.7	4 Pages
- Size 0 - 0001FFFF PL232, PL300, PL332, PL333 or PL450 module						
26C1000A	MXIC	Yes	50uS	x 0	12.7	Flash
26C1000B	MXIC	+Yes	10uS	x 0	12.5	Flash
(***Note that +*** *old versions of this device have incorrect device code CE)*						
27010	Greenwich	No	20uS	x 0	12.7	Emulator
27010	Intel	Yes	100uS	x 0	12.7	Quick Pulse
27C010	Amd	Yes	100uS	x 0	12.7	
27C010	Atmel	Yes	100uS	+ A/R	13.0	
27C010	Cypress	Yes	100uS	x 0	12.7	
27C010	Holtek	No	75uS	x 0	12.5	
27C010	Intel	Yes	100uS	x 0	12.7	Quick Pulse

27C010	National	Yes	100uS	x 0	12.7	
27C010	Philips	No	100uS	x 1	12.7	2 Pass Verify
27C010	Texas	Yes	500uS	+ A/R	12.5	32 bit prog
27C010	Signetics	No	100uS	x 1	12.7	2 Pass Verify
27C010	Winbond	Yes	100uS	x 0	12.0	Flash
27C010A	Intel	Yes	10uS	x 0	12.0	
27C010A	Texas	Yes	100uS	x 0	12.7	
27C010L	Waferscale	No	100uS	x 0	12.7	
27H010	AMD	Yes	100uS	x 0	12.7	

Device	***Make***	***Int. Ident.***	***Programming Parameters***			***Comment***
			Initial Pulse	***Overprog. Multiplier***	***Vpp V***	
- Size 0 - 0001FFFF		**PL232, PL300, PL332, PL333 or PL450 module** ***(Cont.)***				
27HB010	AMD	Yes	100uS	x 0	12.7	
27HC010	ISSI	Yes	100uS	x 0	12.7	
27C010	ICT	Yes	100uS	x 0	12.7	
27C100	AMD	Yes	100uS	x 0	12.7	
27C100	Mitsubishi	Yes	200uS	x 1	12.5	
27C100	Renesas	Yes	200uS	x 1	12.5	
27C101	Hitachi	Yes	1mS	x 3	12.5	
27C101	Mitsubishi	Yes	200uS	x 1	12.5	
27C101	Renesas	Yes	1mS	x 3	12.5	
27C101AG	Hitachi	Yes	200uS	x 1	12.5	
27C101AG	Renesas	Yes	200uS	x 1	12.5	
27C101AP	Hitachi	Yes	200uS	x 1	12.5	OTP
27C101AP	Renesas	Yes	200uS	x 1	12.5	OTP
27C101G	Hitachi	Yes	200uS	x 1	12.5	
27C101G	Renesas	Yes	200uS	x 1	12.5	
27C101P	Hitachi	Yes	200uS	x 1	12.5	OTP
27C101P	Renesas	Yes	200uS	x 1	12.5	OTP
27C1000	Fujitsu	Yes	500uS	x 3	12.5	
27C1000A	Fujitsu	Yes	100uS	x 0	12.5	
27C1000	MXIC	Yes	100uS	x 0	12.7	
27C1000A	NEC	No	100uS	x 0	12.5	
27C1000	SGS_Thomson	Yes	100uS	x 0	12.7	
27C1000	ST	Yes	100uS	x 0	12.7	
27C1001	Fujitsu	Yes	500uS	x 3	12.5	
27C1001A	Fujitsu	Yes	100uS	x 0	12.5	
27C1001A	NEC	No	100uS	x 0	12.5	
27C1001	SGS_Thomson	Yes	100uS	x 0	12.7	*
27C1001	ST	Yes	100uS	x 0	12.7	*
27C301	Hitachi	Yes	1m	x 3	12.5	
27C301	Renesas	Yes	1m	x 3	12.5	
27C301AG	Hitachi	Yes	200uS	x 1	12.5	
27C301AG	Renesas	Yes	200uS	x 1	12.5	
27C301AP	Hitachi	Yes	200uS	x 1	12.5	OTP
27C301AP	Renesas	Yes	200uS	x 1	12.5	OTP
27C301G	Hitachi	Yes	200uS	x 1	12.5	
27C301G	Renesas	Yes	200uS	x 1	12.5	
27C301P	Hitachi	Yes	200uS	x 1	12.5	OTP

27C301P	Renesas	Yes	200uS	x 1	12.5	OTP
27V101	SGS_Thomson	Yes	100uS	x 0	12.7	*
27V101	ST	Yes	100uS	x 0	12.7	*
278308A	AMIC	Yes	100uS	x 0	12.7	
27E010	Winbond	Yes	100uS	x 0	12.0	Flash
27SF010	SST	Yes	25uS	x 0	12.0	Flash
28C010	Atmel	No	1mS	x 0	5.0	Data Polling
28F001BX_B	Intel	Yes	Polled		12.0	Flash
28F001BX_T	Intel	Yes	Polled		12.0	Flash

Device	*Make*	*Int. Ident.*	*Programming Parameters*			*Comment*
			Initial Pulse	*Overprog. Multiplier*	*Vpp V*	
- Size 0 - 0001FFFF PL232, PL300, PL332, PL333 or PL450 module ***(Cont.)***						
28F010	AMD	Yes	10uS	x 0	12.0	Flash
28F010	Intel	Yes	10uS	x 0	12.0	Flash
28F010	ISSI	Yes	10uS	x 0	12.0	Flash
28F010A	AMD	Yes	10uS	x 0	12.0	" Data Polling
28F101	SGS_Thomson	Yes	10uS	x 0	12.0	Flash
28F101	ST	Yes	10uS	x 0	12.0	Flash
28F1000	MXIC	Yes	10uS	x 0	12.0	" Data Polling
28F1000P	MXIC	Yes	10uS	x 0	12.0	" Data Polling
29C010	Atmel	Yes	1mS	x 0	5.0	) Page write only
29C010_SDP	Atmel	Yes	1mS	x 0	5.0	) Max set size = 2
29C010A	Atmel	Yes	1mS	x 0	5.0	) Page write only
29C010A_SDP	Atmel	Yes	1mS	x 0	5.0	) Max set size = 2
29EE010	Greenliant	Yes	1mS	x 0	5.0	) Page write only
29EE010_SDP	Greenliant	Yes	1mS	x 0	5.0	) Max set size=2
29EE010	SST	Yes	1mS	x 0	5.0	) Page write only
29EE010_SDP	SST	Yes	1mS	x 0	5.0	) Max set size=2
				(**** See programming parameter notes***)		
- Size 0 - 0001FFFF PL232 module						
29LE010	SST	Yes	1mS	x 0	3.0	)
29LE010_SDP	SST	Yes	1mS	x 0	3.0	) Page write only
29LE010_	SST	Yes	1mS	x 0	3.0	) Max set size = 2
29LE010_SDP_	SST	Yes	1mS	x 0	3.0	)
29LV010A_3V	Atmel	Yes	1mS	x 0	3.0	)
- Size 0 - 0001FFFF PL232, PL300, PL332, PL333 or PL450 module						
29LV010A	Atmel	Yes	1mS	x 0	5.0	)
29EE011	Winbond	Yes	1mS	x 0	5.0	) Page write only
29EE011_SDP	Winbond	Yes	1mS	x 0	5.0	) Max set size = 2
		_SDP for Data Protection				) (***See notes***)
29F010	Amd	Yes	10uS	x 0	5.0	Flash Polling
29F010	NexFlash	Yes	10uS	x 0	5.0	Flash Polling
29F010A	AMD	Yes	10uS	x 0	5.0	Flash Polling
29F010B	AMD	Yes	10uS	x 0	5.0	Flash Polling
29F010B	SGS_Thomson	Yes	10uS	x 0	5.0	Flash Polling
29F010B	ST	Yes	10uS	x 0	5.0	Flash Polling

- Size 0 - 0001FFFF PL232, PL332 Mk2 or PL333 module

29F010_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
29F010A_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
29F010B_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect

- Size 0 - 0001FFFF PL232 module

29LV010B	AMD	Yes	10uS	x 0	3.0	Flash Polling

Device	*Make*	*Int. Ident.*	*Programming Parameters*			*Comment*
			Initial Pulse	*Overprog. Multiplier*	*Vpp V*	

- Size 0 - 0001FFFF PL232, PL300, PL332, PL333 or PL450 module

39SF010	SST	Yes	10uS	x 0	5.0	Flash Polling
39SF010A	SST	Yes	10uS	x 0	5.0	Flash Polling
49F010	Atmel	Yes	10uS	x 0	5.0	Flash Polling
571000	Toshiba	Yes	100uS	x 0	12.7	
571001	Toshiba	Yes	100uS	x 0	12.7	
57H1000A	Toshiba	Yes	100uS	x 0	12.7	
12883_1M	Greenwich	No	1uS	x 0	5.0	Non-Volatile RAM

- Size 0 - 0001FFFF x 2 PL996 module

2X0033	Puma	No	200uS	x 1	12.5	Hitachi
2X0558	Puma	No	10uS	x 0	5.0	AMD Flash Polling

- Size 0 - 0001FFFF x 4 PL996 module

2F4000	Puma	Yes	25uS	x 0	12.0	Hitachi FLASH
2U4000	Puma	Yes	200uS	x 1	12.5	Hitachi
FTV12832V_S1	Puma *	No	100uS	x 1	12.7	Force Technology

(**** Note that** _S1 indicates a 'non standard' address pin out. This entry has been added using the same address pin out as the 2U4000.*)

- Size 0 - 00003FFF x 8 PL300, PL328 or PL460 module

27011	Intel	Yes	100uS	x 0	12.7	8 Pages
27C011	Intel	Yes	100uS	x 0	12.7	8 Pages

- Size 0 - 0000FFFF (16 bits) PL400, PL444 or PL445 module

27210	Intel	Yes	1mS	x 3	12.7	
27C210	Intel	Yes	100uS	x 0	12.7	Quick Pulse
27C210	National	Yes	100uS	x 0	12.7	
27C210	Philips	Yes	100uS	x 0	12.7	2 Pass Verify
27C210	Signetics	Yes	100uS	x 0	12.7	2 Pass Verify
27C210	Texas	Yes	500uS	+ A/R	12.5	
27C210L	Waferscale	No	200uS	+ 1mS	12.7	
27C1024	AMD	Yes	100uS	x 0	12.7	
27C1024	Atmel	No	100uS	x 0	12.7	
27C1024	Fujitsu	Yes	500uS	x 3	12.5	
27C1024	Hitachi	Yes	200uS	x 1	12.5	
27C1024	MXIC	Yes	100uS	x 0	12.7	
27C1024	NEC	No	100uS	x 4	12.5	

27C1024	Renesas	Yes	200uS	x 1	12.5	
27C1024	SGS_Thomson	Yes	100uS	x 0	12.7	*
27C1024	ST	Yes	100uS	x 0	12.7	*
27C1024A	Fujitsu	Yes	100uS	x 0	12.5	
27C1024A	NEC	No	100uS	x 0	12.5	
27C102K	Mitsubishi	Yes	200uS	x 1	12.5	
27C102K	Renesas	Yes	200uS	x 1	12.5	
28F102	SGS_Thomson	Yes	10uS	x 0	12.0	Flash
28F102	ST	Yes	10uS	x 0	12.0	Flash
29F102B	SGS_Thomson	Yes	10uS	x 0	12.0	Flash

Device	***Make***	***Int. Ident.***	***Programming Parameters***			***Comment***
			Initial Pulse	***Overprog. Multiplier***	***Vpp V***	
- Size 0 - 0000FFFF (16 bits)		**PL400, PL444 or PL445 module** *(Cont.)*				
29F102B	ST	Yes	10uS	x 0	12.0	Flash
571024	Toshiba	Yes	100uS	x 0	12.7	
57H1024A	Toshiba	Yes	100uS	x 0	12.7	
57H1025A	Toshiba	Yes	100uS	x 0	12.7	
			(* See programming parameter notes)			
- Size 0 - 0000FFFF (16 bits)		**PL420 module**				
27C1100	MXIC	Yes	100uS	x 0	12.7	
- Size 0 - 0000FFFF (16 bits multiplexed)		**PL600 module**				
27C1028	Fujitsu	No	500uS	x 3	12.5	
- Size 0 - 0003FFFF		**PL232, PL300, PL332, PL333 or PL450 module**				
26C2000B	MXIC	+Yes	10uS	x 0	12.5	Flash
(***Note that*** *+ old versions of this device have incorrect device code CF*)						
27C020	AMD	Yes	100uS	x 0	12.7	
27C020	Atmel	Yes	100uS	+ A/R	13.0	
27C020	Holtek	No	75uS	x 0	12.5	
27C020	Intel	Yes	100uS	x 0	12.7	Quick Pulse
27C020	National	Yes	100uS	x 0	12.7	
27C020	Texas	Yes	100uS	+ A/R	13.0	
27LV020	Atmel	Yes	100uS	+ A/R	13.0	
27SF020	SST	Yes	25uS	x 0	12.0	Flash
27C201	Mitsubishi	Yes	100uS	x 0	12.7	
27C201	Renesas	Yes	100uS	x 0	12.7	
27C2000	MXIC	Yes	100uS	x 0	12.7	
27C2000A	MXIC	Yes	10uS	x 0	12.7	
27C2000	Oki	Yes	100uS	x 0	12.7	
27C2001	Fujitsu	Yes	100uS	x 0	12.5	
27C2001	NEC	No	100uS	x 0	12.7	
27C2001	SGS_Thomson	Yes	100uS	x 0	12.7	*
27C2001	ST	Yes	100uS	x 0	12.7	*
27E020	Winbond	Yes	100uS	x 0	12.0	Flash
27V201	SGS_Thomson	Yes	100uS	x 0	12.7	*
27V201	ST	Yes	100uS	x 0	12.7	*
27W201	SGS_Thomson	Yes	100uS	x 0	12.7	
27W201	ST	Yes	100uS	x 0	12.7	

(* ***See programming parameter notes***)

- Size 0 - 0003FFFF PL334 module

28F002BL_B	Intel	Yes	10uS	x 0	12.0	Flash
28F002BX_B	Intel	Yes	10uS	x 0	12.0	Flash
28F002BV_T	Intel	Yes	10uS	x 0	12.0	Flash

Device	***Make***	***Int. Ident.***	***Programming Parameters***			***Comment***
			Initial Pulse	***Overprog. Multiplier***	***Vpp V***	

- Size 0 - 0003FFFF PL232, PL300, PL332 or PL333 module

29F002B	AMD	Yes	10uS	x 0	12.0	) Temporary Sector
29F002T	AMD	Yes	10uS	x 0	12.0	) Unprotect
29F002B_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
29F002T_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
29F002B	Fujitsu	Yes	10uS	x 0	12.0	) Temporary Sector
29F002T	Fujitsu	Yes	10uS	x 0	12.0	) Unprotect
29F002B_SP	Fujitsu	Yes	10uS	x 0	12.0	Sector Protect
29F002T_SP	Fujitsu	Yes	10uS	x 0	12.0	Sector Protect
29F002B	MXIC	Yes	10uS	x 0	12.0	) Temporary Sector
29F002T	MXIC	Yes	10uS	x 0	12.0	) Unprotect
29F002B_SP	MXIC	Yes	10uS	x 0	12.0	Sector Protect
29F002T_SP	MXIC	Yes	10uS	x 0	12.0	Sector Protect
				(***See programming parameter notes***)		

- Size 0 - 0003FFFF PL334 Mk2 module

29LV002B	Fujitsu	Yes	10uS	x 0	12.0	) Temporary Sector
29LV002T	Fujitsu	Yes	10uS	x 0	12.0	) Unprotect
29LV002B_SP	Fujitsu	Yes	10uS	x 0	12.0	Sector Protect
29LV002T_SP	Fujitsu	Yes	10uS	x 0	12.0	Sector Protect
				(***See programming parameter notes***)		

- Size 0 - 0003FFFF PL232, PL300, PL332, PL333 or PL450 module

28F020	AMD	Yes	10uS	x 0	12.0	Flash
28F020	Catalyst	Yes	10uS	x 0	12.0	Flash
28F020	Intel	Yes	10uS	x 0	12.0	Flash
28F020A	AMD	Yes	10uS	x 0	12.0	" Data Polling
28F201	SGS_Thomson	Yes	10uS	x 0	12.0	Flash
28F201	ST	Yes	10uS	x 0	12.0	Flash
28F2000P	MXIC	Yes	10uS	x 0	12.0	" Data Polling
29C020	Atmel	Yes	1mS	x 0	5.0	)
29C020_SDP	Atmel	Yes	1mS	x 0	5.0	)
29C020	Winbond	Yes	1mS	x 0	5.0	) Page write only
29C020_SDP	Winbond	Yes	1mS	x 0	5.0	) Max set size = 2
29EE020	SST	Yes	1mS	x 0	5.0	)
29EE020_SDP	SST	Yes	1mS	x 0	5.0	)
29LV020	Atmel	Yes	1mS	x 0	5.0	)
29SF020	Greenliant	Yes	10uS	x 0	5.0	Flash Polling

39SF020	SST	Yes	10uS	x 0	5.0	Flash Polling
39SF020A	SST	Yes	10uS	x 0	5.0	Flash Polling
49F020	Atmel	Yes	10uS	x 0	5.0	Flash Polling
		_SDP for Data Protection				(***See notes***)

- Size 0 - 0001FFFF (16 bits) PL400, PL444 or PL445 module

27C202	Mitsubishi	Yes	200uS	x 1	12.5	
27C202	Renesas	Yes	200uS	x 1	12.5	

Device	***Make***	***Int. Ident.***	***Programming Parameters***			***Comment***
			Initial Pulse	***Overprog. Multiplier***	***Vpp V***	
- Size 0 - 0001FFFF (16 bits)		**PL400, PL444 or PL445 module *(Cont.)***				
27C220	Intel	Yes	100uS	x 0	12.7	Quick Pulse
27C2048	AMD	Yes	100uS	x 0	12.7	
27C2048	Atmel	Yes	50uS	x 0	13.0	
27C2048	Fujitsu	Yes	100uS	x 0	12.5	
27C2048	MXIC	Yes	100uS	x 0	12.7	
- Size 0 - 0001FFFF (16 bits)		**PL420 module**				
27C2100	MXIC	Yes	100uS	x 0	12.7	
- Size 0 - 0001FFFF (16 bits)		**PL490 Mk2, Mk3 or PL491 Mk2 module**				
28F200BV_B	Intel	Yes	10uS	x 0	12.0	
28F200BV_T	Intel	Yes	10uS	x 0	12.0	
- Size 0 - 0001FFFF (16 bits)		**PL490, Mk2, Mk3, PL491 or Mk2 module**				
29F200B	AMD	Yes	10uS	x 0	12.0	) Temporary Sector
29F200T	AMD	Yes	10uS	x 0	12.0	) Unprotect
29F200B_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
29F200BA	Fujitsu	Yes	10uS	x 0	12.0	) Temporary Sector
29F200TA	Fujitsu	Yes	10uS	x 0	12.0	) Unprotect
29F200BA_SP	Fujitsu	Yes	10uS	x 0	12.0	Sector Protect
29F200TA_SP	Fujitsu	Yes	10uS	x 0	12.0	Sector Protect
29F200B	SGS_Thomson	Yes	10uS	x 0	12.0	) Temporary Sector
29F200B	ST	Yes	10uS	x 0	12.0	) Temporary Sector
29F200T	SGS_Thomson	Yes	10uS	x 0	12.0	) Unprotect
29F200T	ST	Yes	10uS	x 0	12.0	) Unprotect
				(***See programming parameter notes***)		
- Size 0 - 0001FFFF (16 bits)		**PL490 Mk2, Mk3, PL491 Mk2 module**				
29AL002_1	Spansion	Yes	10uS	x 0	12.0	) Temporary Sector
29AL002_2	Spansion	Yes	10uS	x 0	12.0	) Unprotect
29AL002_1_SP	Spansion	Yes	10uS	x 0	12.0	Sector Protect
29AL002_2_SP	Spansion	Yes	10uS	x 0	12.0	Sector Protect
29LV200B	AMD	Yes	10uS	x 0	12.0	)
29LV200B	Fujitsu	Yes	10uS	x 0	12.0	) Temporary Sector
29LV200T	AMD	Yes	10uS	x 0	12.0	) Unprotect
29LV200T	Fujitsu	Yes	10uS	x 0	12.0	)
29LV200B_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect

29LV200B_SP	Fujitsu	Yes	10uS	x 0	12.0	Sector Protect
29LV200T_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
29LV200T_SP	Fujitsu	Yes	10uS	x 0	12.0	Sector Protect
29LV200BB	AMD	Yes	10uS	x 0	12.0	)
29LV200BT	AMD	Yes	10uS	x 0	12.0	) Unprotect
29LV200BB_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
29LV200BT_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
				(***See programming parameter notes***)		

Device	***Make***	***Int. Ident.***	***Programming Parameters***			***Comment***
			Initial Pulse	***Overprog. Multiplier***	***Vpp V***	
- Size 0 - 0003FFFF (16 bits)		**PL400, PL444 or PL445 module**				
27C240	Intel	Yes	100uS	x 0	12.7	Quick Pulse
27C240	Philips	No	100uS	x 0	12.7	2 Pass Verify
27C240	Signetics	No	100uS	x 0	12.7	2 Pass Verify
27C240	Texas	Yes	100uS	x 0	12.7	
27C402	Mitsubishi	Yes	100uS	x 0	12.7	
27C402	Renesas	Yes	100uS	x 0	12.7	
27C4002	SGS_Thomson	Yes	100uS	x 0	12.7	*
27C4002	ST	Yes	100uS	x 0	12.7	*
27C4096	AMD	Yes	100uS	x 0	12.7	
27C4096	Atmel	Yes	50uS	+ A/R	13.0	
27C4096	Fujitsu	Yes	100uS	x 0	12.5	
27C4096	Hitachi	Yes	50uS	x 1	12.5	
27C4096	MXIC	Yes	100uS	x 0	12.7	
27C4096	NEC	No	100uS	x 0	12.5	
27C4096	Renesas	Yes	50uS	x 1	12.5	
				(* See programming parameter notes)		
- Size 0 - 0003FFFF (16 bits)		**PL420 module**				
27C400	AMD	Yes	100uS	x 0	12.7	
27C4000	NEC	No	100uS	x 0	12.5	
27C4100	MXIC	Yes	100uS	x 0	12.7	
574200	Toshiba	Yes	50uS	x 1	12.5	
- Size 0 - 0003FFFF (16 bits)		**PL490 Mk2, Mk3 or PL491 Mk2 module**				
28F400AS_B	Texas	Yes	10uS	x 0	12.0	
28F400AS_T	Texas	Yes	10uS	x 0	12.0	
28F400B5_B	Intel	Yes	10uS	x 0	12.0	
28F400B5_T	Intel	Yes	10uS	x 0	12.0	
28F400BV_B	Intel	Yes	10uS	x 0	12.0	
28F400BV_T	Intel	Yes	10uS	x 0	12.0	
28F400BX_B	Intel	Yes	10uS	x 0	12.0	
28F400BX_T	Intel	Yes	10uS	x 0	12.0	
28F400SUL	Sharp	Yes	10uS	x 0	12.0	
28F400SUL_SP	Sharp	Yes	10uS	x 0	12.0	Sector Protect
				(***See programming parameter notes***)		
- Size 0 - 0003FFFF (16 bits)		**PL989 module**				

DR0016	IGG	Yes	10uS	x 0	5.0	

- Size 0 - 0003FFFF (16 bits) PL490, Mk2, Mk3, PL491 or Mk2 module

29F400B	AMD	Yes	10uS	x 0	12.0	)
29F400T	AMD	Yes	10uS	x 0	12.0	) Temporary Sector
29F400T	Hyundai	Yes	10uS	x 0	12.0	) Unprotect
29F400AB	AMD	Yes	10uS	x 0	12.0	)
29F400AT	AMD	Yes	10uS	x 0	12.0	)
29F400AB_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
29F400AT_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect

Device	*Make*	*Int. Ident.*	*Programming Parameters*			*Comment*
			Initial Pulse	*Overprog. Multiplier*	*Vpp V*	

- Size 0 - 0003FFFF (16 bits) PL490, Mk2, Mk3, PL491 or Mk2 module *(Cont.)*

29F400BA	Fujitsu	Yes	10uS	x 0	12.0	) Temporary Sector
29F400TA	Fujitsu	Yes	10uS	x 0	12.0	) Unprotect
29F400BA_SP	Fujitsu	Yes	10uS	x 0	12.0	Sector Protect
29F400TA_SP	Fujitsu	Yes	10uS	x 0	12.0	Sector Protect
29F400B	SGS_Thomson	Yes	10uS	x 0	12.0	) Temporary Sector
29F400B	ST	Yes	10uS	x 0	12.0	) Temporary Sector
29F400T	SGS_Thomson	Yes	10uS	x 0	12.0	) Unprotect
29F400T	ST	Yes	10uS	x 0	12.0	) Unprotect
29F400BB	AMD	Yes	10uS	x 0	12.0	) Temporary Sector
29F400BT	AMD	Yes	10uS	x 0	12.0	) Unprotect
29F400BB_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
29F400BT_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
				(***See programming parameter notes***)		

- Size 0 - 0003FFFF (16 bits) PL490 Mk2, Mk3 or PL491 Mk2 module

29AL004_1	Spansion	Yes	10uS	x 0	12.0	) Temporary Sector
29AL004_2	Spansion	Yes	10uS	x 0	12.0	) Unprotect
29AL004_1_SP	Spansion	Yes	10uS	x 0	12.0	Sector Protect
29AL004_2_SP	Spansion	Yes	10uS	x 0	12.0	Sector Protect
29DL400BB	AMD	Yes	10uS	x 0	12.0	) Temporary Sector
29DL400BT	AMD	Yes	10uS	x 0	12.0	) Unprotect
29DL400B_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
29DL400T_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
29LV400B	AMD	Yes	10uS	x 0	12.0	)
29LV400B	Fujitsu	Yes	10uS	x 0	12.0	) Temporary Sector
29LV400B	MXIC	Yes	10uS	x 0	12.0	)
29LV400T	AMD	Yes	10uS	x 0	12.0	) Unprotect
29LV400T	Fujitsu	Yes	10uS	x 0	12.0	)
29LV400T	MXIC	Yes	10uS	x 0	12.0	) Temporary Sector
29LV400BB	AMD	Yes	10uS	x 0	12.0	) Unprotect
29LV400BT	AMD	Yes	10uS	x 0	12.0	)
29LV400B_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
29LV400B_SP	Fujitsu	Yes	10uS	x 0	12.0	Sector Protect
29LV400B_SP	MXIC	Yes	10uS	x 0	12.0	Sector Protect
29LV400T_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
29LV400T_SP	Fujitsu	Yes	10uS	x 0	12.0	Sector Protect
29LV400T_SP	MXIC	Yes	10uS	x 0	12.0	Sector Protect

29LV400BB_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
29LV400BT_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
29W400B	SGS_Thomson	Yes	10uS	x 0	12.0	) Temporary Sector
29W400B	ST	Yes	10uS	x 0	12.0	) Temporary Sector
29W400T	SGS_Thomson	Yes	10uS	x 0	12.0	) Unprotect
29W400T	ST	Yes	10uS	x 0	12.0	) Unprotect
				(***See programming parameter notes***)		

Device	***Make***	***Int. Ident.***	***Programming Parameters***			***Comment***
			Initial Pulse	***Overprog. Multiplier***	***Vpp V***	
- Size 0 - 0007FFFF	**PL232, PL300, PL332, PL333 or PL450 module**					
27C040	AMD	Yes	100uS	x 0	12.7	
27C040	Atmel	Yes	100uS	+ A/R	13.0	
27C040	Holtek	No	75uS	x 0	12.5	
27C040	Intel	Yes	100uS	x 0	12.7	Quick Pulse
27C040	National	Yes	100uS	x 0	12.7	
27C040	Texas	Yes	100uS	+ A/R	13.0	
27E040	Winbond	Yes	100uS	x 0	12.0	Flash
27LV040	Atmel	Yes	100uS	+ A/R	13.0	
27PC040	Texas	Yes	100uS	x 0	13.0	OTP
27401	Mitsubishi	Yes	100uS	x 0	12.7	
27401	Renesas	Yes	100uS	x 0	12.7	
27C401	Mitsubishi	Yes	100uS	x 0	12.7	
27C401	Renesas	Yes	100uS	x 0	12.7	
27C4000	MXIC	Yes	100uS	x 0	12.7	
27C4000A	MXIC	Yes	20uS	x 0	12.7	
27C4001	Fujitsu	Yes	100uS	x 0	12.5	
27C4001	Hitachi	Yes	50uS	x 0	12.5	
27C4001	NEC	No	100uS	x 0	12.5	
27C4001	Renesas	Yes	50uS	x 0	12.5	
27C4001	SGS_Thomson	Yes	100uS	x 0	12.7	*
27C4001	ST	Yes	100uS	x 0	12.7	*
27V401	SGS_Thomson	Yes	100uS	x 0	12.7	*
27V401	ST	Yes	100uS	x 0	12.7	*
27W401	SGS_Thomson	Yes	100uS	x 0	12.7	*
27W401	ST	Yes	100uS	x 0	12.7	*
				(* See programming parameter notes)		
- Size 0 - 0007FFFF	**PL334 or PL334 Mk2 module**					
28F004BX_B	Intel	Yes	10uS	x 0	12.0	Flash
28F004BX_T	Intel	Yes	10uS	x 0	12.0	Flash
28F004BV_B	Intel	Yes	10uS	x 0	12.0	Flash
28F004BV_T	Intel	Yes	10uS	x 0	12.0	Flash
- **Size 0 - 0007FFFF**	**PL232, PL300, PL332, PL333 or PL450 module**					
28SF040	SST	Yes	10uS	x 0	5.0	Flash Polling
28SF040A	SST	Yes	10uS	x 0	5.0	Flash Polling

- Size 0 - 0007FFFF PL334 Mk2 module

29LV004B	AMD	Yes	10uS	x 0	12.0	)
29LV004B	Fujitsu	Yes	10uS	x 0	12.0	) Temporary Sector
29LV004T	AMD	Yes	10uS	x 0	12.0	) Unprotect
29LV004T	Fujitsu	Yes	10uS	x 0	12.0	)
29LV004B_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
29LV004B_SP	Fujitsu	Yes	10uS	x 0	12.0	Sector Protect
29LV004T_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
29LV004T_SP	Fujitsu	Yes	10uS	x 0	12.0	Sector Protect
				(***See programming parameter notes***)		

Device	***Make***	***Int. Ident.***	***Programming Parameters***			***Comment***
			Initial Pulse	***Overprog. Multiplier***	***Vpp V***	

- Size 0 - 0007FFFF PL232, PL300, PL332, PL333 or PL450 module

28F4000	MXIC	Yes	Polled		12.0	Flash
29C040	Atmel	Yes	1mS	x 0	5.0	)
29C040_SDP	Atmel	Yes	1mS	x 0	5.0	)
29C040A	Atmel	Yes	1mS	x 0	5.0	) Page write only
29C040A_SDP	Atmel	Yes	1mS	x 0	5.0	) Max set size = 2
29C040	Winbond	Yes	1mS	x 0	5.0	)
29C040_SDP	Winbond	Yes	1mS	x 0	5.0	)
		_SDP for Data Protection				) (***See notes***)
29F040	AMD	Yes	10uS	x 0	5.0	Flash Polling
29F040	MXIC	Yes	10uS	x 0	5.0	Flash Polling
29F040	SGS_Thomson	Yes	10uS	x 0	5.0	Flash Polling
29F040	ST	Yes	10uS	x 0	5.0	Flash Polling
29040A	AMIC	Yes	10uS	x 0	5.0	Flash Polling

- Size 0 - 0007FFFF PL232, PL332 Mk2 or PL333 module

29F040_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
29F040_SP	SGS_Thomson	Yes	10uS	x 0	12.0	Sector Protect
29F040_SP	ST	Yes	10uS	x 0	12.0	Sector Protect
				(***See programming parameter notes***)		

- Size 0 - 0007FFFF PL232, PL300, PL332, PL333 or PL450 module

29F040A	Fujitsu	Yes	10uS	x 0	5.0	Flash Polling
29F040A	Hyundai	Yes	10uS	x 0	5.0	Flash Polling
29F040B	AMD	Yes	10uS	x 0	5.0	Flash Polling
29F040B	SGS_Thomson	Yes	10uS	x 0	5.0	Flash Polling
29F040B	ST	Yes	10uS	x 0	5.0	Flash Polling
29F040C	AMD	Yes	10uS	x 0	5.0	Flash Polling
29F040C	Fujitsu	Yes	10uS	x 0	5.0	Flash Polling

- Size 0 - 0007FFFF PL232, PL332 Mk2A or Mk4 module

29F040A_SP	Fujitsu	Yes	10uS	x 0	3.0	Sector Protect
29F040A_SP	Hyundai	Yes	10uS	x 0	3.0	Sector Protect
29F040B_SP	AMD	Yes	10uS	x 0	3.0	Sector Protect
29F040B_SP	SGS_Thomson	Yes	10uS	x 0	3.0	Sector Protect
29F040B_SP	ST	Yes	10uS	x 0	3.0	Sector Protect
29F040C_SP	AMD	Yes	10uS	x 0	3.0	Sector Protect
29F040C_SP	Fujitsu	Yes	10uS	x 0	3.0	Sector Protect

(***See programming parameter notes***)

- Size 0 - 0007FFFF PL232 module

29LV040	EonSSI	Yes	10uS	x 0	3.0	
29LV040	MXIC	Yes	10uS	x 0	3.0	
29LV040_SP	MXIC	Yes	10uS	x 0	3.0	Sector Protect
29LV040A	EonSSI	Yes	10uS	x 0	3.0	
29LV040B	AMD	Yes	10uS	x 0	3.0	
29LV040B_SP	AMD	Yes	10uS	x 0	3.0	Sector Protect
29W040B	SGS_Thomson	Yes	10uS	x 0	3.0	

Device	*Make*	*Int. Ident.*	*Programming Parameters*			*Comment*
			Initial Pulse	*Overprog. Multiplier*	*Vpp V*	

- Size 0 - 0007FFFF PL232 module *(Cont.)*

29W040B	ST	Yes	10uS	x 0	3.0	
29W040B_SP	SGS_Thomson	Yes	10uS	x 0	3.0	Sector Protect
29W040B_SP	ST	Yes	10uS	x 0	3.0	Sector Protect

- Size 0 - 0007FFFF PL232, PL300, PL332, PL333 or PL450 module

29LV040	Atmel	Yes	1mS	x 0	5.0	Max set size = 2
29LV040A	Atmel	Yes	1mS	x 0	5.0	Max set size = 2
49F040	Atmel	Yes	10uS	x 0	5.0	Flash Polling
574000	Toshiba	Yes	50uS	x 1	12.5	

- Size 0 - 000FFFFF PL232, PL300, PL332, PL333 or PL450 module

27C080	Atmel	Yes	50uS	+ A/R	13.0	
27C8000	MXIC	Yes	100uS	x 0	12.7	
27C8001	NEC	No	50uS	x 0	12.5	
27C801	SGS_Thomson	Yes	50uS	x 0	12.7	*
27C801	ST	Yes	50uS	x 0	12.7	*

(****See programming parameter notes***)

- Size 0 - 000FFFFF PL334 Mk2 module

28F008BV_B	Intel	Yes	10uS	x 0	12.0	Flash
28F008BV_T	Intel	Yes	10uS	x 0	12.0	Flash

- Size 0 - 000FFFFF PL335 module

28F008SA	Intel	Yes	Polled		12.0	Flash
28F008SA_L	Intel	Yes	Polled		12.0	Flash
28F008SC	Sharp	Yes	Polled		12.0	Flash) ***See***
28F008S3	Intel	Yes	Polled		12.0	Flash) ***Below***

(***Note that*** b*lock locking/unlocking not supported in this version.*)

- Size 0 - 000FFFFF PL334 Mk2 module

29LV008B	AMD	Yes	10uS	x 0	12.0	)
29LV008B	Fujitsu	Yes	10uS	x 0	12.0	)
29LV008BA	Fujitsu	Yes	10uS	x 0	12.0	) Temporary Sector
29LV008T	AMD	Yes	10uS	x 0	12.0	) Unprotect
29LV008T	Fujitsu	Yes	10uS	x 0	12.0	)
29LV008B_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
29LV008B_SP	Fujitsu	Yes	10uS	x 0	12.0	Sector Protect

29LV008T_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
29LV008T_SP	Fujitsu	Yes	10uS	x 0	12.0	Sector Protect
29LV008BB	AMD	Yes	10uS	x 0	12.0	) Temporary Sector
29LV008BT	AMD	Yes	10uS	x 0	12.0	) Unprotect
29LV008BB_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
29LV008BT_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
				(***See programming parameter notes***)		

Device	*Make*	*Int. Ident.*	*Programming Parameters* *Initial Pulse*	*Overprog. Multiplier*	*Vpp V*	*Comment*
- Size 0 - 0007FFFF (16 bits)	**PL420 module**					
27C800	SGS_Thomson	Yes	50uS	x 0	12.5	*
27C800	ST	Yes	50uS	x 0	12.5	*
27C8000	NEC	No	50uS	x 0	12.5	
27C8100	MXIC	Yes	100uS	x 0	12.7	
27C8111	MXIC	Yes	100uS	x 0	12.7	
578200	Toshiba	Yes	50uS	x 1	12.5	
				(* ***See programming parameter notes***)		
- Size 0 - 000FFFFF	**PL335 and PL336 module**					
29F080	AMD	Yes	10uS	x 0	12.0	) Temporary Sector
29F080	Fujitsu	Yes	10uS	x 0	12.0	) Unprotect
29F080_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
29F080_SP	Fujitsu	Yes	10uS	x 0	12.0	Sector Protect
29F080B	AMD	Yes	10uS	x 0	12.0	) Temporary Sector
29F080B_SP	AMD	Yes	10uS	x 0	12.0	) Sector Protect
29LV081	AMD	Yes	10uS	x 0	12.0	) Temporary Sector
29LV080	Fujitsu	Yes	10uS	x 0	12.0	) Unprotect
29LV081_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
29LV080_SP	Fujitsu	Yes	10uS	x 0	12.0	Sector Protect
49BV008A	Atmel	Yes	10uS	x 0	12.0	) Temporary Sector) Unprotect
				(***See programming parameter notes***)		
- Size 0 - 0007FFFF (16 bits)	**PL490 Mk3 module**		***(See Section 14.3.)***			
28F800B3_B	Intel	Yes	10uS	x 0	12.0	
- Size 0 - 0007FFFF (16 bits)	**PL490 Mk2, Mk3 or PL491 Mk2 module**					
29800T	AMIC	Yes	10uS	x 0	12.0	)
29800U	AMIC	Yes	10uS	x 0	12.0	) Temporary Sector
29DL800BB	AMD	Yes	10uS	x 0	12.0	) Unprotect
29DL800BT	AMD	Yes	10uS	x 0	12.0	)
29DL800BB_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
29DL800BT_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
- Size 0 - 0007FFFF (16 bits)	**PL490 Mk2, Mk3, PL491 or Mk2 module**					
29AL008_1	Spansion	Yes	10uS	x 0	12.0	)
29AL008_2	Spansion	Yes	10uS	x 0	12.0	)
29F800AB	SGS_Thomson	Yes	10uS	x 0	12.0	)

29F800AB	ST	Yes	10uS	x 0	12.0	)
29F800BB	AMD	Yes	10uS	x 0	12.0	)
29F800B	Fujitsu	Yes	10uS	x 0	12.0	) Temporary Sector
29F800BT	AMD	Yes	10uS	x 0	12.0	) Unprotect
29F800T	Fujitsu	Yes	10uS	x 0	12.0	)
29AL008_1_SP	Spansion	Yes	10uS	x 0	12.0	Sector Protect
29AL008_2_SP	Spansion	Yes	10uS	x 0	12.0	Sector Protect
29F800BB_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
29F800B_SP	Fujitsu	Yes	10uS	x 0	12.0	Sector Protect
29F800BT_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
29F800T_SP	Fujitsu	Yes	10uS	x 0	12.0	Sector Protect
				(***See programming parameter notes***)		

Device	***Make***	***Int. Ident.***	***Programming Parameters***			***Comment***
			Initial Pulse	***Overprog. Multiplier***	***Vpp V***	
- Size 0 - 0007FFFF (16 bits)		**PL490 Mk2, Mk3 or PL491 Mk2 module**				
29LV800BB	AMD	Yes	10uS	x 0	12.0	) Temporary Sector
29LV800BT	AMD	Yes	10uS	x 0	12.0	) Unprotect
29LV800B	Fujitsu	Yes	10uS	x 0	12.0	)
29LV800B	MXIC	Yes	10uS	x 0	12.0	)
29LV800BA	Fujitsu	Yes	10uS	x 0	12.0	) Temporary Sector
29LV800T	Fujitsu	Yes	10uS	x 0	12.0	) Unprotect
29LV800T	MXIC	Yes	10uS	x 0	12.0	)
29LV800BB_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
29LV800BT_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
29LV800B_SP	Fujitsu	Yes	10uS	x 0	12.0	Sector Protect
29LV800B_SP	MXIC	Yes	10uS	x 0	12.0	Sector Protect
29LV800T_SP	Fujitsu	Yes	10uS	x 0	12.0	Sector Protect
29LV800T_SP	MXIC	Yes	10uS	x 0	12.0	Sector Protect
				(***See programming parameter notes***)		
- Size 0 - 0007FFFF (16 bits)		**PL490 Mk3 module**		***(Max set size = 1)***		
29WB800	Hitachi	Yes	Polled		3.0	128 Word Page
29WB800	Renesas	Yes	Polled		3.0	128 Word Page
29WB800_SP	Hitachi	Yes	Polled		12.0	128 Word Page
29WB800_SP	Renesas	Yes	Polled		12.0	128 Word Page Sector Protect
29WT800	Hitachi	Yes	Polled		3.0	128 Word Page
29WT800	Renesas	Yes	Polled		3.0	128 Word Page
29WT800_SP	Hitachi	Yes	Polled		12.0	128 Word Page
29WT800_SP	Renesas	Yes	Polled		12.0	128 Word Page Sector Protect
				(***See programming parameter notes***)		
- Size 0 - 0007FFFF (16 bits)		**PL490 Mk2, Mk3 or PL491 Mk2 module**				
49LV8192A	Atmel	Yes	10uS	x 0	12.0	Temporary Sector Unprotect
- Size 0 - 001FFFFF		**PL335 and PL336 module**				
29F016	AMD	Yes	10uS	x 0	12.0	) Temporary Sector) Unprotect
29F016_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
29F016	Fujitsu	Yes	10uS	x 0	12.0	) Temporary Sector

						) Unprotect
29F016_SP	Fujitsu	Yes	10uS	x 0	12.0	Sector Protect
29F016B	AMD	Yes	10uS	x 0	12.0	) Temporary Sector
29F016B	SGS_Thomson	Yes	10uS	x 0	12.0	) Unprotect
29F016B	ST	Yes	10uS	x 0	12.0	)
29F016B_SP	AMD	Yes	10uS	x 0	12.0	Sector Protect
				(***See programming parameter notes***)		

- Size 0 - 001FFFFF PL334 Mk2 module

29LV116DB	AMD	Yes	10uS	x 0	12.0	) Temporary Sector
29LV116DT	AMD	Yes	10uS	x 0	12.0	) Unprotect
39VF016	SST	Yes	20uS	x 0	3.0	

Device	***Make***	***Int. Ident.***	***Programming Parameters***			***Comment***
			Initial Pulse	***Overprog. Multiplier***	***Vpp V***	

- Size 0 - 0000FFFF PL334 Mk2 module

39VF016_64K	SST	Yes	20uS	x 0	3.0	

- Size 0 - 000FFFFF (16 bits) PL491 Mk2 module

27C1602	Oki	No	25uS	x 0	11.5	
27V1602	Oki	No	10uS	+ A/R	9.8	

- Size 0 - 000FFFFF (16 bits) PL420 module

27C160	SGS_Thomson	Yes	50uS	x 0	12.5	*
27C160	ST	Yes	50uS	x 0	12.5	*
5716200	Toshiba	Yes	25uS	x 1	12.5	
				(* See programming parameter notes)		

- Size 0 - 000FFFFF (16 bits) PL493 module

28F016SA	Intel	Yes	Polled		12.0	Flash

- Size 0 - 000FFFFF (16 bits) PL490 Mk3 module

29AL016_1	Spansion	Yes	10uS	x 0	12.0	)
29AL016_2	Spansion	Yes	10uS	x 0	12.0	)
29F160DB	AMD	Yes	10uS	x 0	12.0	)
29F160DB	Spansion	Yes	10uS	x 0	12.0	)
29F160DT	AMD	Yes	10uS	x 0	12.0	)
29F160DT	Spansion	Yes	10uS	x 0	12.0	)
29LV160BB	AMD	Yes	10uS	x 0	12.0	) Temporary Sector
29LV160B	Fujitsu	Yes	10uS	x 0	12.0	) Unprotect
29LV160B	MXIC	Yes	10uS	x 0	12.0	)
29LV160BT	AMD	Yes	10uS	x 0	12.0	)
29LV160T	Fujitsu	Yes	10uS	x 0	12.0	)
29LV160T	MXIC	Yes	10uS	x 0	12.0	)
29LV160DB	AMD	Yes	10uS	x 0	12.0	)
29LV160DT	AMD	Yes	10uS	x 0	12.0	)
29W160EB	SGS_Thomson	Yes	10uS	x 0	12.0	)
29W160EB	ST	Yes	10uS	x 0	12.0	)
29W160ET	SGS_Thomson	Yes	10uS	x 0	12.0	)
29W160ET	ST	Yes	10uS	x 0	12.0	)

- Size 0 - 0009FFFF (16 bits) PL490 Mk3 module

29LV160B_10M	Pan_Fujitsu	Yes	10uS	x 0	12.0	) Temporary Sector

	The first 10 Meg of 29LV160B Fujitsu					) Unprotect
- Size 0 - 000FFFFF (16 bits)	**PL993 module**					
49BV1611T	Atmel	Yes	10uS	x 0	5.0	Flash Polling
- Size 0 - 000FFFFF (16 bits)	**PL490 Mk3 module**					
49LV1614	Atmel	Yes	10uS	x 0	12.0	Flash Polling
- Size 0 - 000FFFFF (16 bits)	**PL501 module**					
84VD21081	Fujitsu	Yes	10uS	x 0	12.0	) Temporary Sector) Unprotect

Device	***Make***	***Int. Ident.***	***Programming Parameters***			***Comment***
			Initial Pulse	***Overprog. Multiplier***	***Vpp V***	
- Size 0 - 000FFFFF (16 bits)	**PL500 module**					
FB16S2TP	Mitsubishi	Yes	Polled		3.0	) 128 Word Page
FB16S2TP	Renesas	Yes	Polled		3.0	) Temporary Unlock
- Size 0 - 0009FFFF (16 bits)	**PL500 module**					
FB16S2TP_10M	Pan_Mitsubi	Yes	Polled		3.0	) 128 Word Page
	The first 10 Meg of FB16S2TP Mitsubishi					) Temporary Unlock
- Size 0 - 001FFFFF (16 bits)	**PL420 module**					
27C322	SGS_Thomson	Yes	50uS	x 0	12.0	
27C322	ST	Yes	50uS	x 0	12.0	
- Size 0 - 00000F (16 bits)	**PL306, PL308 or PL309 module**					***Use Rear Position***
93C06	National	No	10mS	x 0	5	
- Size 0 - 00003F (16 bits)	**PL306, PL308 or PL309 module**					***Use Rear Position***
93C14_1K	Atmel	No	10mS	x 0	5	
- Size 0 - 00003F (16 bits)	**PL306, PL308 or PL309 module**					***Use Rear Position***
93C46	National	No	10mS	x 0	5	
93C46	SGS_Thomson	No	10mS	x 0	5	
93C46	ST	No	10mS	x 0	5	
93C46B	Atmel	No	10mS	x 0	5	
- Size 0 - 00003F (16 bits)	**PL306 module**					
93C14_3V_1K	Atmel	No	10mS	x 0	3	
93C46B_3V	Atmel	No	10mS	x 0	3	
- Size 0 - 00007F (16 bits)	**PL306, PL308 or PL309 module**					***Use Rear Position***
93C56	National	No	10mS	x 0	5	
93C56	SGS_Thomson	No	10mS	x 0	5	
93C56	ST	No	10mS	x 0	5	
93CS56	SGS_Thomson	No	10mS	x 0	5	
93CS56	ST	No	10mS	x 0	5	

- Size 0 - 0000FF (16 bits) PL306, PL308 or PL309 module

						Use Rear Position
93C66	Atmel	No	10mS	x 0	5	
93C66	Catalyst	No	10mS	x 0	5	
93C66	National	No	10mS	x 0	5	
93C66	SGS_Thomson	No	10mS	x 0	5	
93C66	ST	No	10mS	x 0	5	

Device	***Make***	***Int. Ident.***	***Programming Parameters***			***Comment***
			Initial Pulse	***Overprog. Multiplier***	***Vpp V***	
- Size 0 - 0000FF (16 bits)		**PL306 module**				
93C14_4K	Atmel	No	10mS	x 0	5	
93C14_3V_4K	Atmel	No	10mS	x 0	3	
93C66_3V	Atmel	No	10mS	x 0	3	
- Size 0 - 0003FF (16 bits)		**PL308 or PL309 module**				
						Use Rear Position
93C86	Catalyst	No	5mS	x 0	5	

MICROCONTROLLERS

- Size 0 - 000003FF (14 bits) PL480 module

48R06A	Holtek	No	500uS	x 0	12.5	

- Size 0 - 000007FF PL201 module

47P201	Toshiba	No	1mS	x 3	12.5	

- Size 0 - 000007FF PL242 module

47P242	Toshiba	No	1mS	x 3	12.5	

- Size 0 - 00001FFF PL808 or PL818 module

87P808	Toshiba	No	100uS	x 0	12.7	
87P809	Toshiba	No	100uS	x 0	12.7	

- Size 0 - 00000FFF PL300 module + Special adaptor

63701V	Hitachi	No	1mS	x 3	12.5	)
63701V	Renesas	No	1mS	x 3	12.5	)
63705V	Hitachi	No	1mS	x 3	12.5	)
63705V	Renesas	No	1mS	x 3	12.5	)
				(***See programming parameter notes***)		

- Size 0 - 0001FFF PL374 or PL300 module + Mitsubishi PCA4752 adaptor

37451E4	Mitsubishi	No	1mS	x 3	12.5
37451E4	Renesas	No	1mS	x 3	12.5

- Size C080 – FFFD PL375 module

38223E4	Mitsubishi	No	1mS	x 3	12.5
38223E4	Renesas	No	1mS	x 3	12.5

- Size 4080 – FFFD PL375 module

38227EC	Mitsubishi	No	200uS	x 1	12.5
38227EC	Renesas	No	200uS	x 1	12.5

Device	*Make*	*Int. Ident.*	*Programming Parameters*			*Comment*
			Initial Pulse	*Overprog. Multiplier*	*Vpp V*	
- Size 00000 - 3FFFF PL376 module						
30624FGA	Mitsubishi	No			3.3	Flash Status
30624FGA	Renesas	No			3.3	Flash Status
30624FGM	Mitsubishi	No			3.3	Flash Status
30624FGM	Renesas	No			3.3	Flash Status
306NAFG	Mitsubishi	No			3.3	Flash Status
306NAFG	Renesas	No			3.3	Flash Status
- Size 0 – 00001FFF PL521 module						
R5F21272	Renesas	No			5.0	Flash Status
- Size 0 – 00003FFF PL521 module						
R5F21274	Renesas	No			5.0	Flash Status
- Size 0 – 00005FFF PL521 module						
R5F21275	Renesas	No			5.0	Flash Status
- Size 0 – 00007FFF PL521 module						
R5F21276	Renesas	No			5.0	Flash Status
- Size 0 – 0000FFFF PL522 module						
R5F2L387	Renesas	No			5.0	Flash Status
- Size 00000 –FFFFF PL376A module						
R5F3640	Mitsubishi	No			5.0	+ DATA Array
R5F3640	Renesas	No			5.0	+ DATA Array
- Size 0 - 00001FFF PL847 module						
47P847	Toshiba	No	1mS	x 3	12.5	
- Size 0 - 00007FFF PL432 module						
4074329	Hitachi	No	1mS	x 3	12.5	
4074329	Renesas	No	1mS	x 3	12.5	
- Size 0 - 0000FFFF PL304 module						
6473042	Hitachi	No	200uS	x 1	12.5	
6473042	Renesas	No	200uS	x 1	12.5	

- Size 0 - 0001FFFF PL304 module

6473048	Hitachi	No	200uS	x 1	12.5	
6473048	Renesas	No	200uS	x 1	12.5	
64F3048	Hitachi	No	80uS	x 1	12.0	Flash
64F3048	Renesas	No	80uS	x 1	12.0	Flash

- Size 0 - 0000BFFF PL325 or PL326 module ***(Use software version 2.DE or higher)***

6473256	Hitachi	No	200uS	x 1	12.5	
6473256	Renesas	No	200uS	x 1	12.5	

Device	*Make*	*Int. Ident.*	*Programming Parameters*			*Comment*
			Initial Pulse	*Overprog. Multiplier*	*Vpp V*	

- Size 0 - 0000EFFF PL325 or PL326 module ***(Use software version 2.DE or higher)***

6473257	Hitachi	No	200uS	x 1	12.5	
6473257	Renesas	No	200uS	x 1	12.5	

- Size 0 - 00007FFF PL325 or PL326 module ***(Use software version 2.DE or higher)***

6473258	Hitachi	No	1mS	x 3	12.5	
6473258	Renesas	No	1mS	x 3	12.5	

- Size 0 - 00007FFF PL329 module

6473294	Hitachi	No	200uS	x 1	12.5	
6473294	Renesas	No	200uS	x 1	12.5	

- Size 0 - 0000F77F PL329 module

6473297	Hitachi	No	200uS	x 1	12.5	
6473297	Renesas	No	200uS	x 1	12.5	

- Size 0 - 00003FFF PL330 or PL331 module

6473308	Hitachi	No	1mS	x 3	12.5	
6473308	Renesas	No	1mS	x 3	12.5	

- Size 0 - 00007FFF PL597 module

6473334	Hitachi	No	200uS	x 1	12.5	
6473334	Renesas	No	200uS	x 1	12.5	
64F3334	Hitachi	No	25uS	x 0	12.0	Flash
64F3334	Renesas	No	25uS	x 0	12.0	Flash

- Size 0 - 0000F77F PL597 module

6473337	Hitachi	No	200uS	x 1	12.5	
6473337	Renesas	No	200uS	x 1	12.5	
64F3337	Hitachi	No	25uS	x 0	12.0	Flash
64F3337	Renesas	No	25uS	x 0	12.0	Flash

- Size 0 - 00007FFF PL380 module

64738024	Hitachi	No	200uS	x 1	12.5	
64738024	Renesas	No	200uS	x 1	12.5	

- Size 0 - 00007FFF PL330 or PL331 module

6473378	Hitachi	No	200uS	x 1	12.5	

6473378	Renesas	No	200uS	x 1	12.5	

- Size 0 - 0000BFFF PL330 or PL331 module

6473388	Hitachi	No	200uS	x 1	12.5	
6473388	Renesas	No	200uS	x 1	12.5	

- Size 0 - 00007FFF PL364 or PL365 module

6473644	Hitachi	No	1mS	x 3	12.5	
6473644	Renesas	No	1mS	x 3	12.5	

Device	*Make*	*Int. Ident.*	*Programming Parameters*			*Comment*
			Initial Pulse	*Overprog. Multiplier*	*Vpp V*	

- Size 0 - 00007FFF PL366 module

64F3644	Hitachi	No	25uS	x 0	12.0	Flash
64F3644	Renesas	No	25uS	x 0	12.0	Flash

- Size 0 - 0000EDFF PL385 module

6473827	Hitachi	No	200uS	x 1	12.5	
6473827	Renesas	No	200uS	x 1	12.5	

- Size 0 - 00007FFF PL383 or PL384 module

6473834	Hitachi	No	1mS	x 3	12.5	
6473834	Renesas	No	1mS	x 3	12.5	

- Size 0 - 0000EDFF PL383 or PL384 module

6473837	Hitachi	No	200uS	x 1	12.5	
6473837	Renesas	No	200uS	x 1	12.5	

- Size 0 - 00007FFF PL532 module

6475328	Hitachi	No	1mS	x 3	12.5	
6475328	Renesas	No	1mS	x 3	12.5	
6475348	Hitachi	No	1mS	x 3	12.5	
6475348	Renesas	No	1mS	x 3	12.5	

- Size 0 - 00007FFF PL367 or PL369 module

64F3664	Hitachi	No	1mS		5.0	Flash Polling
64F3664	Renesas	No	1mS		5.0	Flash Polling
64F3694	Hitachi	No	1mS		5.0	Flash Polling
64F3694	Renesas	No	1mS		5.0	Flash Polling

- Size 0 - 00007FFF PL381 module

64F38024	Hitachi	No	25uS	x 0	3.0	Flash
64F38024	Renesas	No	25uS	x 0	3.0	Flash
64F38124	Hitachi	No	25uS	x 0	3.0	Flash
64F38124	Renesas	No	25uS	x 0	3.0	Flash

- Size 0 - 0000DFFF PL370 module

64F36037	Hitachi	No	1mS		5.0	Flash Polling
64F36037	Renesas	No	1mS		5.0	Flash Polling
64F36057	Hitachi	No	1mS		5.0	Flash Polling

64F36057	Renesas	No	1mS		5.0	Flash Polling
64F3687	Hitachi	No	1mS		5.0	Flash Polling
64F3687	Renesas	No	1mS		5.0	Flash Polling
- Size 0 - 0000EFFF PL382 module						
64F38327	Hitachi	No	25uS	x 0	3.0	Flash
64F38327	Renesas	No	25uS	x 0	3.0	Flash

Device	*Make*	*Int. Ident.*	*Programming Parameters*			*Comment*
			Initial Pulse	*Overprog. Multiplier*	*Vpp V*	
- Size 0 - 0001FFFF PL213 module						
64F2134	Hitachi	No	25uS	x 0	5.0	Flash
64F2134	Renesas	No	25uS	x 0	5.0	Flash
64F2134A	Hitachi	No	25uS	x 0	3.0	Flash
64F2134A	Renesas	No	25uS	x 0	3.0	Flash
- Size 0 - 0001FFFF PL264 module						
64F2648	Hitachi	No	25uS	x 0	5.0	Flash
64F2648	Renesas	No	25uS	x 0	5.0	Flash
64F2692	Hitachi	No	25uS	x 0	5.0	Flash
64F2692	Renesas	No	25uS	x 0	5.0	Flash
- Size 0 - 0003FFFF PL262 module						
64F2623	Hitachi	No	25uS	x 0	3.0	Flash
64F2623	Renesas	No	25uS	x 0	3.0	Flash
64F2626	Hitachi	No	25uS	x 0	3.0	Flash
64F2626	Renesas	No	25uS	x 0	3.0	Flash
- Size 0 - 0003FFFF PL239 module						
64F2698	Hitachi	No	25uS	x 0	3.0	Flash
64F2698	Renesas	No	25uS	x 0	3.0	Flash
- Size 0 - 00003FFF PL218 module						
17P218	NEC	No	1mS	x 1	12.5	
- Size 0 - 00003F7F PL316 module						
75P316	NEC	No	1mS	x 1	12.5	
75P316A	NEC	No	1mS	x 1	12.5	
- Size 0 - 00001F7F PL116 or PL117 module						
75P008	NEC	No	1mS	x 1	12.5	
- Size 0 - 00003FFF PL116 or PL117 module						
75P0016	NEC	No	1mS	x 1	12.5	
- Size 0 - 00007FFF PL018 module						
75P3018A	NEC	No	1mS	x 1	12.5	

- Size 0 - 00003FFF PL311 module

75P3116	NEC	No	1mS	x 1	12.5

- Size 0 - 00001FFF PL430 module

75P4308	NEC	No	1mS	x 1	12.5

- Size 2000 - 0001FFFF PL992 module

7500P10S	Sony	No	100uS	x 0	12.5

- Size 0 - 00007FFF PL014 module

78P014	NEC	No	1mS	x 3	12.5

Device	*Make*	*Int. Ident.*	*Programming Parameters*			*Comment*
			Initial Pulse	*Overprog. Multiplier*	*Vpp V*	

- Size 0 - 00007FFF PL054 module

78P054	NEC	No	100uS	x 0	12.5

- Size 0 - 0000EFFF PL054 module

78P058	NEC	No	100uS	x 0	12.5

- Size 0 - 00007FFF PL064 module

78P064	NEC	No	100uS	x 0	12.5

- Size 0 - 0000EFFF PL064A module

78P0308	NEC	No	100uS	x 0	12.5

- Size 0 - 00005FFF PL083 module

78P083	NEC	No	100uS	x 0	12.5

- Size 0 - 00001FFF PL312 module

78P312A	NEC	No	1mS	x 3	12.5

- Size 0 - 0001FFFF PL838 module ***(See Section 14.17.)***

87PS38	Toshiba	No	100uS	x 0	12.7

- Size 0 - 00003FFF PL846 module

87PH46	Toshiba	No	100uS	x 0	12.7

- Size 0 - 00003FFF PL845 module

87PH47	Toshiba	No	100uS	x 0	12.7

- Size 004000 - 00007FFF PL849 module ***(See Section 14.17.)***

87PM40AN	Toshiba	No	1mS	x 3	12.5

- Size 0011100 - 0001FFFF PL848 module

87PS64F	Toshiba	No	100uS	x 0	12.7

- Size 0000000 - 0001FFFF PL912 module

91PW12F	Toshiba	No	100uS	x 0	12.7

- Size 0 - 000001FF PL860 Mk2 or PL861 module ***(See Section 14.13.)***

Z86E02	Zilog	No	1mS	x 3	13.0

Z86E02_1925	Zilog	No	1mS	x 3	13.0
Z86E03	Zilog	No	1mS	x 3	13.0

- Size 0 - 000003FF PL860 Mk2 or PL861 module ***(See Section 14.13.)***

Z86E04	Zilog	No	1mS	x 3	13.0
Z86E06	Zilog	No	1mS	x 3	13.0

- Size 0 - 000007FF PL860 Mk2 or PL861 module ***(See Section 14.13.)***

Z86E08	Zilog	No	1mS	x 3	13.0

- Size 0 - 00000FFF PL863, PL864 or PL865 module ***(See Section 14.13.*****)**

Z86E30_1873	Zilog	No	1mS	x 3	13.2

Device	***Make***	***Int. Ident.***	***Programming Parameters***			***Comment***
			Initial Pulse	***Overprog. Multiplier***	***Vpp V***	

- Size 0 - 00001FFF PL863, PL864 or PL865 module ***(See Section 14.13.*****)**

Z86E33	Zilog	No	1mS	x 3	13.2

- Size 0 - 00001FFF PL863, PL864 or PL865 module ***(See Section 14.13.)***

Z86733	Zilog	No	1mS	x 3	13.2

- Size 0 - 00003FFF PL863, PL864 or PL865 module ***(See Section 14.13.)***

Z86E34	Zilog	No	1mS	x 3	13.2

- Size 0 - 000007FF PL874 Mk2 module

8742	Intel	No	51mS	x 0	21.0

- Size 0 - 000003FF PL874 or Mk2 module

8748H	Intel	No	51mS	x 0	21.0
8748H	NEC	No	51mS	x 0	21.0

- Size 0 - 000007FF PL874 or Mk2 module

8749H	Intel	No	51mS	x 0	21.0
8749H	NEC	No	51mS	x 0	21.0

- Size 0 - 000003FF PL751 module

87C750	Philips	No	25x 100uS	+ 0	12.7
87C750	Signetics	No	25x 100uS	+ 0	12.7

- Size 0 - 000007FF PL751 module

87C748	Philips	No	25x 100uS	+ 0	12.7
87C748	Signetics	No	25x 100uS	+ 0	12.7
87C749	Philips	No	25x 100uS	+ 0	12.7
87C749	Signetics	No	25x 100uS	+ 0	12.7
87C751	Philips	No	25x 100uS	+ 0	12.7
87C751	Signetics	No	25x		

			100uS	+ 0	12.7	
87C752	Philips	No	25x			
			100uS	+ 0	12.7	
87C752	Signetics	No	25x			
			100uS	+ 0	12.7	

Device	*Make*	*Int.*	*Programming Parameters*			*Comment*
			Ident.	*Initial Pulse*	*Overprog. Multiplier*	*Vpp V*
- Size 0 - 00000FFF PL875 or PL876 module						
8751BH	Intel	No	25x			
			100uS	+ 0	12.7	
8751H	AMD	No	1mS	+ 2	21.0	
8751H	Intel	No	50mS	+ 0	21.0	
87C51	AMD	Yes	25x			
			100uS	+ 0	12.7	
87C51	Intel	Yes	25x			
			100uS	+ 0	12.7	
87C51_FX	Intel	Yes	5x			
			100uS	+ 0	12.7	(***See programming parameter notes***)
87C51	Philips	Yes	25x			
			100uS	+ 0	12.7	
87C51SB	Signetics	Yes	5x			
			100uS	+ 0	12.7	
87C51	Philips	Yes	25x			
			100uS	+ 0	12.7	
87C51SB	Signetics	Yes	5x			
			100uS	+ 0	12.7	
87C51	Temic	Yes	5x			
			100uS	+ 0	12.7	
89C51	Atmel	Yes	100uS	+ 0	12.0	Flash
89LV51	Atmel	Yes	100uS	+ 0	12.0	Flash
89S51	Atmel	Yes	100uS	+ 0	12.0	Flash
- Size 0 - 00001FFF PL875 or PL876 module						
87C51FA	Intel	Yes	25x			
			100uS	+ 0	12.7	
87C51FA_FX	Intel	Yes	5x			
			100uS	+ 0	12.7	(***See programming parameter notes***)
87C51FA	Philips	Yes	25x			
			100uS	+ 0	12.7	
87C51FA	Signetics	Yes	25x			
			100uS	+ 0	12.7	
- Size 0 - 00001FFF PL878 module						

87C51GB	Intel	Yes	5x 100uS	+ 0	12.7	

- Size 0 - 00001FFF PL875 or PL876 module

8752BH	Intel	Yes	25x 100uS	+ 0	12.7	
87C52	Philips	Yes	25x 100uS	+ 0	12.7	
87C52	Signetics	Yes	25x 100uS	+ 0	12.7	

Device	*Make*	*Int. Ident.*	*Programming Parameters* *Initial Pulse*	*Overprog. Multiplier*	*Vpp V*	*Comment*

- Size 0 - 00001FFF PL875 or PL876 module ***(Cont.)***

87C52	Temic	Yes	5x 100uS	+ 0	12.7	
87C52X2	Atmel	Yes	100uS	+ 0	12.7	
87C52X2	Temic	Yes	100uS	+ 0	12.7	
87C521	AMD	Yes	25x 100uS	+ 0	12.7	
87C52T2	AMD	Yes	25x 100uS	+ 0	12.7	
8753H	AMD	No	50mS	+ 0	21.0	
87C652	Philips	No	25x 100uS	+ 0	12.7	
87C652	Signetics	No	25x 100uS	+ 0	12.7	
89C52	Atmel	Yes	100uS	+ 0	12.0	Flash
89C52_5V	Atmel	Yes	100uS	+ 0	5.0	Flash
89LV52	Atmel	Yes	100uS	+ 0	12.0	Flash
89S52	Atmel	Yes	100uS	+ 0	12.0	Flash
C501	Infineon	Yes	25x 100uS	+ 0	12.7	
C501	Siemens	Yes	25x 100uS	+ 0	12.7	Correct Code
C501_	Siemens	Yes	25x 100uS	+ 0	12.7	Man. Code 57

- Size 0 - 000027FF PL875 or PL876 module

89S8252	Atmel	Yes	100uS	+ 0	12.0	Flash
		(***To select EEPROM or option bit see Section 14.7.***)				

- Size 0 - 00002FFF PL875 or PL876 module

89S53	Atmel	Yes	100uS	+ 0	12.0	Flash
		(***To select option bit see Section 14.7.***)				

- Size 0 - 00003FFF PL875 or PL876 module

87C51FB	Intel	Yes	25x 100uS	+ 0	12.7	
87C51FB_FX	Intel	Yes	5x			

			100uS	+ 0	12.7	(***See programming parameter notes***)
87C51FB	Philips	Yes	25x			
			100uS	+ 0	12.7	
87C51FB	Signetics	Yes	25x			
			100uS	+ 0	12.7	
87C51RB2	Philips	Yes	5x			
			100uS	+ 0	12.7	FX Device
87C51RB2	Signetics	Yes	5x			
			100uS	+ 0	12.7	FX Device

Device	***Make***	***Int. Ident.***	***Programming Parameters***			***Comment***
			Initial Pulse	***Overprog. Multiplier***	***Vpp V***	
- Size 0 - 00003FFF	**PL875 or PL876 module** *(Cont.)*					
87C520	Dallas	Yes	5x			
			100uS	+ 0	12.7	FX Device
		(***To select option bit see Section 14.7.***)				
87C54	Intel	Yes	5x			
			100uS	+ 0	12.7	FX Device
87C54	Philips	Yes	5x			
			100uS	+ 0	12.7	
87C54	Signetics	Yes	5x			
			100uS	+ 0	12.7	
87C54X2	Temic	Yes	100uS	+ 0	12.7	
87C541	AMD	Yes	25x			
			100uS	+ 0	12.7	
87C654	Philips	Yes	25x			
			100uS	+ 0	12.7	
87C654	Signetics	Yes	25x			
			100uS	+ 0	12.7	
- Size 0 - 00004FFF	**PL875 or PL876 module**	***(See Section 14.7.)***				
89C55	Atmel	Yes	100uS	+ 0	12.0	Flash
89C55WD	Atmel	Yes	1uS	+ 0	12.0	Flash
- Size 0 - 00007FFF	**PL875 or PL876 module**					
87C51FC	Intel	Yes	5x			
			100uS	+ 0	12.7	*
87C51FC	Philips	Yes	5x			
			100uS	+ 0	12.7	*
87C51FC	Signetics	Yes	5x			
			100uS	+ 0	12.7	*
87C51RC	Philips	Yes	5x			
			100uS	+ 0	12.7	*
87C51RC2	Philips	Yes	5x			
			100uS	+ 0	12.7	*
87C51RC	Signetics	Yes	5x			
			100uS	+ 0	12.7	*
87C51RC2	Signetics	Yes	5x			
			100uS	+ 0	12.7	*

87C528	Philips	Yes	25x 100uS	+ 0	12.7	
87C528	Signetics	Yes	25x 100uS	+ 0	12.7	
			(See programmer parameter notes*)			

- Size 0 - 00003FFF PL875 or PL876 module ***(See Section 14.7.)***

89C51RB2	Atmel	Yes	10mS	+ 0	5.0	Flash Page

- Size 0 - 00007FFF PL875 or PL876 module ***(See Section 14.7.)***

89C51RC2	Atmel	Yes	10mS	+ 0	5.0	Flash Page

Device	***Make***	***Int. Ident.***	***Programming Parameters*** ***Initial Pulse***	***Overprog. Multiplier***	***Vpp V***	***Comment***

- Size 0 - 0000FFFF PL875 or PL876 module ***(See Section 14.7.)***

89C51RD2	Atmel	Yes	10mS	+ 0	5.0	Flash Page

- Size 0 - 00007FFF PL505 module

C505CA	Infineon	Yes	100uS	+ 0	11.5	
C505CA	Siemens	Yes	100uS	+ 0	11.5	
C505CA_S1	Siemens	Yes	100uS	+ 0	12.0	

- Size 0 - 00007FFF PL506 module

C505L	Infineon	Yes	100uS	+ 0	11.5	
C505L	Siemens	Yes	100uS	+ 0	11.5	

- Size 0 - 00001FFF PL552 module

P87C552	Philips	Yes	5x 50uS	+ 0	12.7	
P87C552	Signetics	Yes	5x 50uS	+ 0	12.7	
S87C552	Philips	Yes	25x 100uS	+ 0	12.7	
S87C552	Signetics	Yes	25x 100uS	+ 0	12.7	

- Size 0 - 00003FFF PL592 module

87C592	Philips	Yes	25x 100uS	+ 0	12.7	
87C592	Signetics	Yes	25x 100uS	+ 0	12.7	

- Size 0 - 000007FF PL870 module

89C2051	Atmel	Yes	100uS	+ 0	12.0	Flash

- Size 0 - 00000FFF PL625 module ***(See Section 14.9.)***

62T00C	ST	No	2mS	+ 0	13.0	1.0K EPROM OPTION BYTES
62E01C	ST	No	2mS	+ 0	13.0	1.5K EPROM OPTION BYTES
62T01C	ST	No	2mS	+ 0	13.0	1.5K EPROM

						OPTION BYTES
62T03C	ST	No	2mS	+ 0	13.0	1.0K EPROM
						OPTION BYTES
62T00C	SGS_Thomson	No	2mS	+ 0	13.0	1.0K EPROM
						OPTION BYTES
62E01C	SGS_Thomson	No	2mS	+ 0	13.0	1.5K EPROM
						OPTION BYTES
62T01C	SGS_Thomson	No	2mS	+ 0	13.0	1.5K EPROM
						OPTION BYTES
62T03C	SGS_Thomson	No	2mS	+ 0	13.0	1.0K EPROM
						OPTION BYTES

Device	*Make*	*Int. Ident.*	*Programming Parameters*			*Comment*
			Initial Pulse	*Overprog. Multiplier*	*Vpp V*	
- Size 0 - 00000FFF	**PL620 or PL621 module**		*(See Section 14.9.)*			
62E20	ST	No	1mS	+ 0	13.0	4K EPROM
62E25	ST	No	1mS	+ 0	13.0	4K EPROM
62T08C	ST	No	2mS	+ 0	13.0	1K OTP ROM
						OPTION BYTES
62T09C	ST	No	2mS	+ 0	13.0	1K OTP ROM
						OPTION BYTES
62T10	ST	No	1mS	+ 0	13.0	2K OTP ROM
62T10C	ST	No	2mS	+ 0	13.0	2K OTP ROM
						OPTION BYTES
62T15	ST	No	1mS	+ 0	13.0	2K OTP ROM
62T15C	ST	No	2mS	+ 0	13.0	2K OTP ROM
						OPTION BYTES
62T20	ST	No	1mS	+ 0	13.0	4K OTP ROM
62T20C	ST	No	2mS	+ 0	13.0	4K OTP ROM
						OPTION BYTES
62T25	ST	No	1mS	+ 0	13.0	4K OTP ROM
62T25C	ST	No	2mS	+ 0	13.0	4K OTP ROM
						OPTION BYTES
- Size 0 - 00000FFF	**PL620 or PL621 module**		*(See Section 14.9.)*			
62E20	SGS_Thomson	No	1mS	+ 0	13.0	4K EPROM
62E25	SGS_Thomson	No	1mS	+ 0	13.0	4K EPROM
62T08C	SGS_Thomson	No	2mS	+ 0	13.0	1K OTP ROM
						OPTION BYTES
62T09C	SGS_Thomson	No	2mS	+ 0	13.0	1K OTP ROM
						OPTION BYTES
62T10	SGS_Thomson	No	1mS	+ 0	13.0	2K OTP ROM
62T10C	SGS_Thomson	No	2mS	+ 0	13.0	2K OTP ROM
						OPTION BYTES
62T15	SGS_Thomson	No	1mS	+ 0	13.0	2K OTP ROM
62T15C	SGS_Thomson	No	2mS	+ 0	13.0	2K OTP ROM
						OPTION BYTES
62T20	SGS_Thomson	No	1mS	+ 0	13.0	4K OTP ROM
62T20C	SGS_Thomson	No	2mS	+ 0	13.0	4K OTP ROM
						OPTION BYTES
62T25	SGS_Thomson	No	1mS	+ 0	13.0	4K OTP ROM
62T25C	SGS_Thomson	No	2mS	+ 0	13.0	4K OTP ROM

						OPTION BYTES
- Size 0 - 00001FFF	**PL620 or PL621 module**		***(See Section 14.9.)***			
62T28C	ST	No	2mS	+ 0	13.0	8K OTP ROM
						OPTION BYTES
62T30B	ST	No	2mS	+ 0	13.0	8K OTP ROM
					5.0	128B EEPROM
						OPTION BYTES
62T28C	SGS_Thomson	No	2mS	+ 0	13.0	8K OTP ROM
						OPTION BYTES
62T30B	SGS_Thomson	No	2mS	+ 0	13.0	8K OTP ROM
					5.0	128B EEPROM
						OPTION BYTES
Device	***Make***	***Int. Ident.***	***Programming Parameters*** ***Initial Pulse***	***Overprog. Multiplier***	***Vpp V***	***Comment***
- Size 0 - 00000FFF	**PL623 or PL625 module**		***(See Section 14.9.)***			
62T55B	ST	No	2mS	+ 0	13.0	4K OTP ROM
						OPTION BYTES
62T55C	ST	No	2mS	+ 0	13.0	4K OTP ROM
						OPTION BYTES
62T55B	SGS_Thomson	No	2mS	+ 0	13.0	4K OTP ROM
						OPTION BYTES
62T55C	SGS_Thomson	No	2mS	+ 0	13.0	4K OTP ROM
						OPTION BYTES
- Size 0 - 00000FFF	**PL626 *(rear)* module**		***(See Section 14.9.)***			
62T52C	ST	No	2mS	+ 0	13.0	2K OTP ROM
						OPTION BYTES
62T52C	SGS_Thomson	No	2mS	+ 0	13.0	2K OTP ROM
						OPTION BYTES
- Size 0 - 00000FFF	**PL622 or PL626 *(front)* module**		***(See Section 14.9.)***			
62T53C	ST	No	2mS	+ 0	13.0	2K OTP ROM
						OPTION BYTES
62T60B	ST	No	2mS	+ 0	13.0	4K OTP ROM
					5.0	128B EEPROM
						OPTION BYTES
62T60C	ST	No	2mS	+ 0	13.0	4K OTP ROM
					5.0	128B EEPROM
						OPTION BYTES
- Size 0 - 00000FFF	**PL622 or PL626 *(front)* module**		***(See Section 14.9.)***			
62T53C	SGS_Thomson	No	2mS	+ 0	13.0	2K OTP ROM
						OPTION BYTES
62T60B	SGS_Thomson	No	2mS	+ 0	13.0	4K OTP ROM
					5.0	128B EEPROM
						OPTION BYTES
62T60C	SGS_Thomson	No	2mS	+ 0	13.0	4K OTP ROM
					5.0	128B EEPROM
						OPTION BYTES
- Size 0 - 00000FFF	**PL626 *(rear)* module**		***(See Section 14.9.)***			
62T62C	ST	No	2mS	+ 0	13.0	2K OTP ROM

					5.0	64B EEPROM OPTION BYTES
62T62C	SGS_Thomson	No	2mS	+ 0	13.0	2K OTP ROM
					5.0	64B EEPROM OPTION BYTES

- Size 0 - 00000FFF PL622 or PL626 *(front)* module ***(See Section 14.9.)***

62T63C	ST	No	2mS	+ 0	13.0	2K OTP ROM
					5.0	64B EEPROM OPTION BYTES
62T63C	SGS_Thomson	No	2mS	+ 0	13.0	2K OTP ROM
					5.0	64B EEPROM OPTION BYTES

Device	***Make***	***Int. Ident.***	***Programming Parameters***			***Comment***
			Initial Pulse	***Overprog. Multiplier***	***Vpp V***	

- Size 0 - 00000FFF PL623 or PL625 module ***(See Section 14.9.)***

62T65B	ST	No	2mS	+ 0	13.0	4K OTP ROM
					5.0	128B EEPROM OPTION BYTES
62T65C	ST	No	2mS	+ 0	13.0	4K OTP ROM
					5.0	128B EEPROM OPTION BYTES
62T65B	SGS_Thomson	No	2mS	+ 0	13.0	4K OTP ROM
					5.0	128B EEPROM OPTION BYTES
62T65C	SGS_Thomson	No	2mS	+ 0	13.0	4K OTP ROM
					5.0	128B EEPROM OPTION BYTES

- Size 0 - 0000FFFF PL630 module ***(See Section 14.34.)***

7FLITEBC	ST	No	5.0	Self Timed
7FLITE02	ST	No	5.0	Self Timed
7FLITE09	ST	No	5.0	Self Timed
7FLITEBC	SGS_Thomson	No	5.0	Self Timed
7FLITE02	SGS_Thomson	No	5.0	Self Timed
7FLITE09	SGS_Thomson	No	5.0	Self Timed

- Size 0 - 0000FFFF PL631 module ***(See Section 14.34.)***

7FLITE15	ST	No	5.0	Self Timed
7FLIT15BF1	ST	No	5.0	Self Timed
7FLITE19	ST	No	5.0	Self Timed
7FLIT19BF1	ST	No	5.0	Self Timed
7FLITE25	ST	No	5.0	Self Timed
7FLITE29	ST	No	5.0	Self Timed
7FLITE35	ST	No	5.0	Self Timed
7FLITE39	ST	No	5.0	Self Timed
7FLITE15	SGS_Thomson	No	5.0	Self Timed
7FLIT15BF1	SGS_Thomson	No	5.0	Self Timed
7FLITE19	SGS_Thomson	No	5.0	Self Timed
7FLIT19BF1	SGS_Thomson	No	5.0	Self Timed
7FLITE25	SGS_Thomson	No	5.0	Self Timed
7FLITE29	SGS_Thomson	No	5.0	Self Timed

7FLITE35	SGS_Thomson	No			5.0	Self Timed
7FLITE39	SGS_Thomson	No			5.0	Self Timed

- Size 0 - 00001FFF PL715, PL716 or PL717 module ***(See Section 14.12.)***

68HC705B5	Freescale	No	10mS	+ 0	15.5	
68HC705B5	Motorola	No	10mS	+ 0	15.5	

- Size 0 - 00003FFF PL715, PL716 or PL717 module ***(See Section 14.12.)***

68HC705B16	Freescale	No	10mS	+ 0	15.5	EPROM
			10ms	+ 0	15.5	EEPROM
68HC705B16	Motorola	No	10mS	+ 0	15.5	EPROM
			10ms	+ 0	15.5	EEPROM

Device	***Make***	***Int. Ident.***	***Programming Parameters***			***Comment***
			Initial Pulse	***Overprog. Multiplier***	***Vpp V***	

- Size 0 - 00003FFF PL715, PL716 or PL717 module ***(See Section 14.12.) (Cont.)***

68HC705B16N	Freescale	No	10mS	+ 0	15.5	EPROM
			10ms	+ 0	15.5	EEPROM
68HC705B16N	Motorola	No	10mS	+ 0	15.5	EPROM
			10ms	+ 0	15.5	EEPROM

- Size 0 - 00007FFF PL715, PL716 or PL717 module ***(See Section 14.12.)***

68HC705B32	Freescale	No	10mS	+ 0	15.5	EPROM
			10ms	+ 0	15.5	EEPROM
68HC705B32	Motorola	No	10mS	+ 0	15.5	EPROM
			10ms	+ 0	15.5	EEPROM
68HC705X32	Freescale	No	10mS	+ 0	15.5	EPROM
			10ms	+ 0	15.5	EEPROM
68HC705X32	Motorola	No	10mS	+ 0	15.5	EPROM
			10ms	+ 0	15.5	EEPROM

- Size 0 - 0000FFFF PL732 module ***(See Section 14.18.)***

68HC705F32	Freescale	No	5mS	+ 0	17.0	EPROM
			15ms	+ 0	10.0	EEPROM
68HC705F32	Motorola	No	5mS	+ 0	17.0	EPROM
			15ms	+ 0	10.0	EEPROM

- Size 0 - 00001FFF PL700 or PL701 module ***(See Section 14.5.)***

68HC705C8	Freescale	No	2mS	+ 0	14.7
68HC705C8	Motorola	No	2mS	+ 0	14.7
68HC705C8A	Freescale	No	2mS	+ 0	14.7
68HC705C8A	Motorola	No	2mS	+ 0	14.7
68HSC705C8A	Freescale	No	2mS	+ 0	14.7
68HSC705C8A	Motorola	No	2mS	+ 0	14.7

- Size 0 - 00001FFF PL707 or PL708 module ***(See Section 14.23.)***

68HC705JJ7	Freescale	No	5mS	+ 0	16.5
68HC705JJ7	Motorola	No	5mS	+ 0	16.5
68HC705JP7	Freescale	No	5mS	+ 0	16.5
68HC705JP7	Motorola	No	5mS	+ 0	16.5

- Size 0 - 000007FF PL740 module ***(See Section 14.26.)***

68HC705KJ1	Freescale	No	5mS	+ 0	16.5	
68HC705KJ1	Motorola	No	5mS	+ 0	16.5	
68HLC705KJ1	Freescale	No	5mS	+ 0	16.5	
68HLC705KJ1	Motorola	No	5mS	+ 0	16.5	

- Size 0 - 00001FFF PL720, PL721, PL722 or PL723 module ***(See Section 14.14.)***

68HC705P6	Freescale	No	4mS	+ 0	16.5	
68HC705P6	Motorola	No	4mS	+ 0	16.5	
68HC705P6A	Freescale	No	4mS	+ 0	16.5	
68HC705P6A	Motorola	No	4mS	+ 0	16.5	
68HC705P9	Freescale	No	4mS	+ 0	15.0	
68HC705P9	Motorola	No	4mS	+ 0	15.0	
Device	***Make***	***Int. Ident.***	***Programming Parameters Initial Pulse***	***Overprog. Multiplier***	***Vpp V***	***Comment***

- Size 0 - 00003FFF PL720, PL721, PL722 or PL723 module ***(See Section 14.14.)***

68HC805P18	Freescale	No	4mS	+ 0	12.0	
68HC805P18	Motorola	No	4mS	+ 0	12.0	

- Size 0 - 00003FFF PL700 or PL701 module ***(See Section 14.5.)***

68HC705C9	Freescale	No	2mS	+ 0	14.7	
68HC705C9	Motorola	No	2mS	+ 0	14.7	
68HC705C9A	Freescale	No	2mS	+ 0	14.7	
68HC705C9A	Motorola	No	2mS	+ 0	14.7	

- Size 0 - 0000FFFF PL836 module ***(See Section 14.19.)***

68HC708XL36	Freescale	No	1mS	+ 0	13.0	
68HC708XL36	Motorola	No	1mS	+ 0	13.0	

- Size 0 - 0000FFFF PL760 module ***(See Section 14.25.)***

68HC908AB32	Freescale	No	40uS	+ 0	9.0	FLASH & EEPROM
68HC908AB32	Motorola	No	40uS	+ 0	9.0	FLASH & EEPROM
68HC908AS60A	Freescale	No	40uS	+ 0	8.5	FLASH & EEPROM
68HC908AS60A	Motorola	No	40uS	+ 0	8.5	FLASH & EEPROM
68HC908AZ32A	Freescale	No	40uS	+ 0	9.0	FLASH & EEPROM
68HC908AZ32A	Motorola	No	40uS	+ 0	9.0	FLASH & EEPROM
68HC908AZ60	Freescale	No	1mS	+ 0	12.0	FLASH & EEPROM
68HC908AZ60	Motorola	No	1mS	+ 0	12.0	FLASH & EEPROM
68HC908AZ60A	Freescale	No	40uS	+ 0	8.5	FLASH & EEPROM
68HC908AZ60A	Motorola	No	40uS	+ 0	8.5	FLASH & EEPROM

- Size 0 - 0000FFFF PL761 module ***(See Section 14.25.)***

68HC908GR4	Freescale	No	40uS	+ 0	8.5	FLASH
68HC908GR4	Motorola	No	40uS	+ 0	8.5	FLASH
68HC908GR8	Freescale	No	40uS	+ 0	8.5	FLASH
68HC908GR8	Motorola	No	40uS	+ 0	8.5	FLASH

- Size 0 - 0000FFFF PL762 module ***(See Section 14.25.)***

68HC908JK3	Freescale	No	40uS	+ 0	8.5	FLASH
68HC908JK3	Motorola	No	40uS	+ 0	8.5	FLASH
68HC908JK3E	Freescale	No	40uS	+ 0	8.5	FLASH
68HC908JK3E	Motorola	No	40uS	+ 0	8.5	FLASH

68HRC908JK3E	Freescale	No	40uS	+ 0	8.5	FLASH
68HRC908JK3E	Motorola	No	40uS	+ 0	8.5	FLASH
68HC908JK8	Freescale	No	40uS	+ 0	8.5	FLASH/MOR
68HC908JK8	Motorola	No	40uS	+ 0	8.5	FLASH/MOR

- Size 0 - 0000FFFF PL763 or PL764 module ***(See Section 14.25.)***

68HC908JL3	Freescale	No	40uS	+ 0	8.5	FLASH
68HC908JL3	Motorola	No	40uS	+ 0	8.5	FLASH
68HC908JL3E	Freescale	No	40uS	+ 0	8.5	FLASH
68HC908JL3E	Motorola	No	40uS	+ 0	8.5	FLASH
68HRC908JL3	Freescale	No	40uS	+ 0	8.5	FLASH

Device	***Make***	***Int. Ident.***	***Programming Parameters***			***Comment***
			Initial Pulse	***Overprog. Multiplier***	***Vpp V***	

- Size 0 - 0000FFFF PL763 or PL764 module ***(See Section 14.25.) (Cont.)***

68HRC908JL3	Motorola	No	40uS	+ 0	8.5	FLASH
68HRC908JL3E	Freescale	No	40uS	+ 0	8.5	FLASH
68HRC908JL3E	Motorola	No	40uS	+ 0	8.5	FLASH
68HC908JL8	Freescale	No	40uS	+ 0	8.5	FLASH
68HC908JL8	Motorola	No	40uS	+ 0	8.5	FLASH

- Size 0 - 0000FFFF PL71D module ***(See Section 14.10.)***

MC68HC711D3	Freescale	No	3mS	+ 0	12.0	EPROM
MC68HC711D3	Motorola	No	3mS	+ 0	12.0	EPROM

- Size 0 - 0000FFFF PL71E module ***(See Section 14.8.)***

68HC11A0	Freescale	No	10mS	+ 0	5.0	CONFIG Only
68HC11A0	Motorola	No	10mS	+ 0	5.0	CONFIG Only
68HCP11A0	Freescale	No	10mS	+ 0	5.0	CONFIG Only
68HCP11A0	Motorola	No	10mS	+ 0	5.0	CONFIG Only
68HC11A1	Freescale	No	10mS	+ 0	5.0	EEPROM & CONFIG
68HC11A1	Motorola	No	10mS	+ 0	5.0	EEPROM & CONFIG
68HC11E1	Freescale	No	10mS	+ 0	5.0	EEPROM & CONFIG
68HC11E1	Motorola	No	10mS	+ 0	5.0	EEPROM & CONFIG
68HC711E9	Freescale	No	3mS	+ 0	12.0	EPROM
			10mS	+ 0	5.0	EEPROM & CONFIG
68HC711E9	Motorola	No	3mS	+ 0	12.0	EPROM
			10mS	+ 0	5.0	EEPROM & CONFIG
68S711E9	Freescale	No	3mS	+ 0	12.0	EPROM
			10mS	+ 0	5.0	EEPROM & CONFIG
68S711E9	Motorola	No	3mS	+ 0	12.0	EPROM
			10mS	+ 0	5.0	EEPROM & CONFIG

- Size 0 - 0000FFFF PL71E module ***(See Section 14.8.)***

68HC711E20	Freescale	No	3mS	+ 0	12.0	EPROM
			10mS	+ 0	5.0	EEPROM & CONFIG
68HC711E20	Motorola	No	3mS	+ 0	12.0	EPROM
			10mS	+ 0	5.0	EEPROM & CONFIG
68HC711E32	Freescale	No	3mS	+ 0	12.0	EPROM
			10mS	+ 0	5.0	EEPROM & CONFIG
68HC711E32	Motorola	No	3mS	+ 0	12.0	EPROM

			10mS	+ 0	5.0	EEPROM & CONFIG
- Size 0 - 0000FFFF	**PL71E Mk2 module**		***(See Section 14.8.)***			
68HC711EA9	Freescale	No	3mS	+ 0	12.0	EPROM
			10mS	+ 0	5.0	EEPROM & CONFIG
68HC711EA9	Motorola	No	3mS	+ 0	12.0	EPROM
			10mS	+ 0	5.0	EEPROM & CONFIG
- Size 0 - 0000FFFF	**PL71E module**		***(See Section 14.8.)***			
68HC811E2	Freescale	No	10mS	+ 0	5.0	EEPROM & CONFIG
68HC811E2	Motorola	No	10mS	+ 0	5.0	EEPROM & CONFIG

Device	***Make***	***Int. Ident.***	***Programming Parameters***			***Comment***
			Initial Pulse	***Overprog. Multiplier***	***Vpp V***	
- Size 0 - 0000FFFF	**PL71K module**		***(See Section 14.11.)***			
68HC711KA4	Freescale	No	3mS	+ 0	12.0	EPROM
			10mS	+ 0	5.0	EEPROM & CONFIG
68HC711KA4	Motorola	No	3mS	+ 0	12.0	EPROM
			10mS	+ 0	5.0	EEPROM & CONFIG
- Size 0 - 0000FFFF	**PL71L module**		***(See Section 14.8.)***			
68HC711L6	Freescale	No	4mS	+ 0	12.3	EPROM
			10mS	+ 0	5.0	EEPROM & CONFIG
68HC711L6	Motorola	No	4mS	+ 0	12.3	EPROM
			10mS	+ 0	5.0	EEPROM & CONFIG
- Size 0 - 000003FF	**PL855 module**		***(See Sections 14.20. & 16.4.)***			
COP87L20CJ	National	Yes	50uS	+ A/R	12.7	
- Size 0 - 00007FFF	**PL855 module**		***(See Sections 14.20. & 16.4.)***			
COP87L20RJ	National	Yes	50uS	+ A/R	12.7	
- Size 0 - 00003FFF	**PL855 module**		***(See Sections 14.20. & 16.4.)***			
COP87L84BC	National	Yes	50uS	+ A/R	12.7	
COP87L84C_G	National	Yes	50uS	+ A/R	12.7	
- Size 0 - 00007FFF	**PL855 module**		***(See Sections 14.20. & 16.4.)***			
COP87L84H_R	National	Yes	50uS	+ A/R	12.7	
- Size 0 - 000003FF	**PL850 or PL851 module**		***(See Sections 14.16. & 16.4.)***			
COP8SAA716	National	Yes	50uS	+ 200uS	12.7	
COP8SAA720	National	Yes	50uS	+ 200uS	12.7	
COP8SAA728	National	Yes	50uS	+ 200uS	12.7	
- Size 0 - 000007FF	**PL850 or PL851 module**		***(See Sections 14.16. & 16.4.)***			
COP8SAB720	National	Yes	50uS	+ 200uS	12.7	
COP8SAB728	National	Yes	50uS	+ 200uS	12.7	
- Size 0 - 00000FFF	**PL850 or PL851 module**		***(See Sections 14.16. & 16.4.)***			
COP8SAC720	National	Yes	50uS	+ 200uS	12.7	

COP8SAC728	National	Yes	50uS	+ 200uS	12.7	

- Size 0 - 00000FFF PL850 module ***(See Sections 14.16. & 16.4.)***

COP8SAC740	National	Yes	50uS	+ 200uS	12.7	

- Size 0 - 00001FFF PL850 or PL851 module ***(See Sections 14.16. & 16.4.)***

COP8SGE728	National	Yes	50uS	+ 200uS	12.7	

- Size 0 - 00000FFF PL850 module ***(See Sections 14.16. & 16.4.)***

COP8SGE740	National	Yes	50uS	+ 200uS	12.7	

- Size 0 - 00007FFF PL850 or PL851 module ***(See Sections 14.16. & 16.4.)***

COP8SGR728	National	Yes	50uS	+ 200uS	12.7	

Device	***Make***	***Int. Ident.***	***Programming Parameters***			***Comment***
			Initial Pulse	***Overprog. Multiplier***	***Vpp V***	

- Size 0 - 00000FFF PL850 module ***(See Sections 14.16. & 16.4.)***

COP8SGR740	National	Yes	50uS	+ 200uS	12.7	

- Size 0 - 000003FF (16 bit) PL229 or PL230 module ***(See Sections 14.21. & 16.4.)***

90S2313	Atmel	Yes	1mS	x 0	12.0	FLASH, EE & Fuse

- Size 0 - 00000FFF (16 bit) PL231 module ***(See Sections 14.21. & 16.4.)***

90S8535	Atmel	Yes	1mS	x 0	12.0	FLASH, EE & Fuse

- Size 0 - 000FFF (16 bit) PL231 Mk3 module ***(See Sections 14.21. & 16.4.)***

MEGA8535	Atmel	Yes	1uS	x 0	12.0	FLASH, EE & Fuse

- Size 0 - 0003FF (16 bit) PL227 module ***(See Sections 14.32. & 16.4.)***

TINY2313_ICP	Atmel	Yes	1uS	x 0	12.0	FLASH, EE & Fuse

- Size 0 - 000FFF (16 bit) PL266 module ***(See Sections 14.21. & 16.4.)***

TINY26	Atmel	Yes	1uS	x 0	12.0	FLASH, EE & Fuse

- Size 0 - 000FFF (16 bit) PL227 module ***(See Sections 14.32. & 16.4.)***

MEGA8_ICP	Atmel	Yes	1uS	x 0	12.0	FLASH, EE & Fuse

- Size 0 - 001FFF (16 bit) PL231 Mk3 module ***(See Sections 14.21. & 16.4.)***

MEGA16	Atmel	Yes	1uS	x 0	12.0	FLASH, EE & Fuse

- Size 0 - 001FFF (16 bit) PL231 Mk2 module ***(See Sections 14.21. & 16.4.)***

MEGA163	Atmel	Yes	1uS	x 0	12.0	FLASH, EE & Fuse

- Size 0 - 007FFF (16 bit) PL231 Mk3 module ***(See Sections 14.21. & 16.4.)***

MEGA644P	Atmel	Yes	1uS	x 0	12.0	FLASH, EE & Fuse
MEGA644	Atmel	Yes	1uS	x 0	12.0	FLASH, EE & Fuse

- Size 0 - 008FFF (16 bit) PL233 module ***(See Sections 14.21. & 16.4.)***

XMEGA32D4	Atmel	Yes	1uS	x 0	3.3V	FLASH, EE & Fuse

- Size 0 - 000000FE (12 bit) PL679 module ***(See Sections 14.15. & 16.4.)***

PIC10F200	Microchip	No	2mS	x 0	13.0	
PIC10F204	Microchip	No	2mS	x 0	13.0	

PIC10F220	Microchip	No	2mS	x 0	13.0

- Size 0 - 000001FE (12 bit) PL679 module ***(See Sections 14.15. & 16.4.)***

PIC10F202	Microchip	No	2mS	x 0	13.0
PIC10F206	Microchip	No	2mS	x 0	13.0
PIC10F222	Microchip	No	2mS	x 0	13.0

Device	*Make*	*Int. Ident.*	*Programming Parameters*			*Comment*
			Initial Pulse	*Overprog. Multiplier*	*Vpp V*	
- Size 0 - 000001FE (12 bit)		**PL651A, PL653, PL657 or PL665 module**				***(See Sections 14.15. & 16.4.)***
PIC12C508	Microchip	No	100uS	x 11	13.0	
PIC12C508A	Microchip	No	100uS	x 11	13.0	
PIC12CE518	Microchip	No	100uS	x 11	13.0	
- Size 0 - 000003FE (12 bit)		**PL651A, PL653, PL657 or PL665 module**				***(See Sections 14.15. & 16.4.)***
PIC12C509	Microchip	No	100uS	x 11	13.0	
PIC12C509A	Microchip	No	100uS	x 11	13.0	
PIC12CE519	Microchip	No	100uS	x 11	13.0	
- **Size 0 - 000003FE (14 bit)**		**PL651A, PL653, PL657 or PL665 module**				***(See Sections 14.15. & 16.4.)***
PIC12C671	Microchip	No	100uS	x 3	13.0	
PIC12F629	Microchip	No	2.5mS	x 0	13.0	
PIC12F675	Microchip	No	2.5mS	x 0	13.0	
PIC12LF629	Microchip	No	2.5mS	x 0	13.0	
PIC12LF675	Microchip	No	2.5mS	x 0	13.0	
- Size 0 - 000007FF (14 bit)		**PL651A, PL653, PL657 or PL665 module**				***(See Sections 14.15. & 16.4.)***
PIC12F683	Microchip	No	2.5mS	x 0	11.0	
- Size 0 - 000007FE (14 bit)		**PL651A, PL653, PL657 or PL665 module**				***(See Sections 14.15. & 16.4.)***
PIC12C672	Microchip	No	100uS	x 3	13.0	
- Size 0 - 000001FE (12 bit)		**PL679 module**	***(See Sections 14.15. & 16.4.)***			
PIC10F206	Microchip	No	2mS	x 0	13.0	
- Size 0 - 000001FE (12 bit)		**PL651A, PL653, PL657 or PL665 module**				***(See Sections 14.15. & 16.4.)***
PIC12F508	Microchip	No	2mS	x 0	13.0	
- Size 0 - 000003FE (12 bit)		**PL651A, PL653, PL657 or PL665 module**				***(See Sections 14.15. & 16.4.)***

PIC12F509	Microchip	No	2mS	x 0	13.0	
PIC12F510	Microchip	No	2mS	x 0	13.0	
PIC12F519	Microchip	No	2mS	x 0	13.0	

- Size 0 - 3FF (12 bit) PL651A, PL653, PL657 or PL665 module ***(See Sections 14.15. & 16.4.)***

PIC12F609	Microchip	No	3.5mS	x 0	12.0	
PIC12HV609	Microchip	No	3.5mS	x 0	12.0	
PIC12F615	Microchip	No	3.5mS	x 0	12.0	
PIC12HV615	Microchip	No	3.5mS	x 0	12.0	

Device	***Make***	***Int. Ident.***	***Programming Parameters***			***Comment***
			Initial Pulse	***Overprog. Multiplier***	***Vpp V***	
- Size 0 - 000003FE (12 bit)		**PL659, PL679 or PL665 module**			***(See Sections 14.15. & 16.4.)***	
PIC16C505	Microchip	No	100uS	x 11	13.0	
PIC16F505	Microchip	No	2mS	x 0	13.0	
PIC16LF505	Microchip	No	2mS	x 0	13.0	
PIC16F506	Microchip	No	2mS	x 0	13.0	
PIC16F630	Microchip	No	2.5mS	x 0	13.0	
PIC16F676	Microchip	No	2.5mS	x 0	13.0	
PIC16LF630	Microchip	No	2.5mS	x 0	13.0	
PIC16LF676	Microchip	No	2.5mS	x 0	13.0	
- Size 0 - 0000017F (12 bit)		**PL650 or PL668 module**				
PIC16C52	Microchip	No	100uS	x 11	13.0	Group 2
PIC16C52_XT	Microchip	No	100uS	x 11	13.0	
- Size 0 - 000001FF (12 bit)		**PL650 or PL668 module**			***(See Sections 14.4. & 16.4.)***	
PIC16C54	Microchip	No	100uS	x 11	13.0	Group 2
PIC16C54_HS	Microchip	No	100uS	x 11	13.0	
PIC16C54_LP	Microchip	No	100uS	x 11	13.0	
PIC16C54_XT	Microchip	No	100uS	x 11	13.0	
PIC16C54A	Microchip	No	100uS	x 11	13.0	Group 2
PIC16C54A_HS	Microchip	No	100uS	x 11	13.0	
PIC16C54A_LP	Microchip	No	100uS	x 11	13.0	
PIC16C54A_XT	Microchip	No	100uS	x 11	13.0	
PIC16C54C	Microchip	No	100uS	x 11	13.0	Group 2
PIC16C54C_HS	Microchip	No	100uS	x 11	13.0	
PIC16C54C_LP	Microchip	No	100uS	x 11	13.0	
PIC16C54C_XT	Microchip	No	100uS	x 11	13.0	
- Size 0 - 000001FF (12 bit)		**PL655, PL657, PL660 or PL665 modu**le				
PIC16F54	Microchip	No	2mS	x 0	13.0	
- Size 0 - 000001FF (12 bit)		**PL650 or PL668 module**			***(See Sections 14.4. & 16.4.)***	
PIC16C55	Microchip	No	100uS	x 11	13.0	Group 3
PIC16C55_HS	Microchip	No	100uS	x 11	13.0	
PIC16C55_LP	Microchip	No	100uS	x 11	13.0	
PIC16C55_XT	Microchip	No	100uS	x 11	13.0	
PIC16C55A	Microchip	No	100uS	x 11	13.0	Group 3

PIC16C55A_HS	Microchip	No	100uS	x 11	13.0	
PIC16C55A_LP	Microchip	No	100uS	x 11	13.0	
PIC16C55A_XT	Microchip	No	100uS	x 11	13.0	

- Size 0 - 000001FF (14 bit) PL650, PL655, PL657, PL660 or PL665 module
(See Sections 14.4. & 16.4.)

PIC16C554	Microchip	No	100uS	x 3	13.0	Group 1

- Size 0 - 000003FF (14 bit) PL650, PL655, PL657, PL660 or PL665 module
(See Sections 14.4. & 16.4.)

PIC16C556	Microchip	No	100uS	x 3	13.0	Group 1

Device	*Make*	*Int. Ident.*	*Programming Parameters* *Initial Pulse*	*Overprog. Multiplier*	*Vpp V*	*Comment*

- Size 0 - 000007FF (14 bit) PL650, PL655, PL657, PL660 or PL665 module
(See Sections 14.4. & 16.4.)

PIC16C558	Microchip	No	100uS	x 3	13.0	Group 1

- Size 0 - 000003FF (12 bit) PL650 or PL668 module ***(See Sections 14.4. & 16.4.)***

PIC16C56	Microchip	No	100uS	x 11	13.0	Group 2
PIC16C56_HS	Microchip	No	100uS	x 11	13.0	
PIC16C56_LP	Microchip	No	100uS	x 11	13.0	
PIC16C56_XT	Microchip	No	100uS	x 11	13.0	

- Size 0 - 000007FF (12 bit) PL650 or PL668 module ***(See Sections 14.4. & 16.4.)***

PIC16C57	Microchip	No	100uS	x 11	13.0	Group 3
PIC16C57_HS	Microchip	No	100uS	x 11	13.0	
PIC16C57_LP	Microchip	No	100uS	x 11	13.0	
PIC16C57_XT	Microchip	No	100uS	x 11	13.0	
PIC16C57C	Microchip	No	100uS	x 11	13.0	
PIC16C57C_HS	Microchip	No	100uS	x 11	13.0	
PIC16C57C_LP	Microchip	No	100uS	x 11	13.0	
PIC16C57C_XT	Microchip	No	100uS	x 11	13.0	
PIC16C58A	Microchip	No	100uS	x 11	13.0	Group 2
PIC16C58A_HS	Microchip	No	100uS	x 11	13.0	
PIC16C58A_LP	Microchip	No	100uS	x 11	13.0	
PIC16C58A_XT	Microchip	No	100uS	x 11	13.0	
PIC16C58B	Microchip	No	100uS	x 11	13.0	Group 2
PIC16C58B_HS	Microchip	No	100uS	x 11	13.0	
PIC16C58B_LP	Microchip	No	100uS	x 11	13.0	
PIC16C58B_XT	Microchip	No	100uS	x 11	13.0	

- Size 0 - 000003FF (14 bit) PL650, PL655, PL657, PL660 or PL665 module
(*See Sections 14.4. & 16.4.)*

PIC16C61	Microchip	No	100uS	x 3	13.0	Group 1

- Size 0 - 000007FF (14 bit) PL651, PL652, PL665 or PL667 module ***(See Sections 14.15. & 16.4.)***

PIC16C62	Microchip	No	100uS	x 3	13.0	
PIC16C62A	Microchip	No	100uS	x 3	13.0	
PIC16C62B	Microchip	No	100uS	x 3	13.0	
PIC16LC62	Microchip	No	100uS	x 3	13.0	

PIC16LC62A	Microchip	No	100uS	x 3	13.0	

- Size 0 - 000001FF (14 bit) PL650, PL655, PL657, PL660 or PL665 module

PIC16C620	Microchip	No	100uS	x 3	13.0	Group 1
PIC16C620A	Microchip	No	100uS	x 3	13.0	Group 1

- Size 0 - 000003FF (14 bit) PL650, PL655, PL657, PL660 or PL665 module
(See Sections 14.4. & 16.4.)

PIC16C621	Microchip	No	100uS	x 3	13.0	Group 1
PIC16LC621	Microchip	No	100uS	x 3	13.0	Group 1
PIC16C621A	Microchip	No	100uS	x 3	13.0	Group 1
PIC16LC621A	Microchip	No	100uS	x 3	13.0	Group 1
PIC16CE624	Microchip	No	100uS	x 3	13.0	Group 1

Device	***Make***	***Int. Ident.***	***Programming Parameters***			***Comment***
			Initial Pulse	***Overprog. Multiplier***	***Vpp V***	

- Size 0 - 000007FF (14 bit) PL650, PL655, PL657, PL660 or PL665 module
(See Sections 14.4. & 16.4.)

PIC16C622	Microchip	No	100uS	x 3	13.0	Group 1
PIC16LC622	Microchip	No	100uS	x 3	13.0	Group 1
PIC16C622A	Microchip	No	100uS	x 3	13.0	
PIC16LC622A	Microchip	No	100uS	x 3	13.0	
PIC16CE625	Microchip	No	100uS	x 3	13.0	

- Size 0 - 00000FFF (14 bit) PL651, PL652, PL665 or PL667 module ***(See Sections 14.15. & 16.4.)***

PIC16C63	Microchip	No	100uS	x 3	13.0	
PIC16LC63	Microchip	No	100uS	x 3	13.0	
PIC16C63A	Microchip	No	100uS	x 3	13.0	
PIC16LC63A	Microchip	No	100uS	x 3	13.0	

- Size 0 - 000007FF (14 bit) PL651, PL654, PL662, PL664 or PL665 module
(See Sections 14.15. & 16.4.)

PIC16C64	Microchip	No	100uS	x 3	13.0	
PIC16C64A	Microchip	No	100uS	x 3	13.0	
PIC16LC64	Microchip	No	100uS	x 3	13.0	
PIC16LC64A	Microchip	No	100uS	x 3	13.0	

- Size 0 - 00000FFF (14 bit) PL651, PL654, PL662, PL664 or PL665 module
(See Sections 14.15. & 16.4.)

PIC16C65	Microchip	No	100uS	x 3	13.0	
PIC16C65A	Microchip	No	100uS	x 3	13.0	
PIC16C65B	Microchip	No	100uS	x 3	13.0	
PIC16LC65	Microchip	No	100uS	x 3	13.0	
PIC16LC65A	Microchip	No	100uS	x 3	13.0	
PIC16LC65B	Microchip	No	100uS	x 3	13.0	
PIC16C662	Microchip	No	100uS	x 3	13.0	

- Size 0 - 000001FF (14 bit) PL650, PL655, PL657, PL660 or PL665 module
(See Sections 14.4. & 16.4.)

PIC16C710	Microchip	No	100uS	x 3	13.0	Group 1

- Size 0 - 000003FF (14 bit) PL650, PL655, PL657, PL660 or PL665 module
(See Sections 14.4. & 16.4.)

PIC16C71	Microchip	No	100uS	x 3	13.0	Group 1
PIC16C711	Microchip	No	100uS	x 3	13.0	Group 1
PIC16C712	Microchip	No	100uS	x 3	13.0	Group 1

- Size 0 - 000007FF (14 bit) PL650, PL655, PL657, PL660 or PL665 module
(See Sections 14.4. & 16.4.)

PIC16C715	Microchip	No	100uS	x 3	13.0	Group 1

- Size 0 - 00001FFF (14 bit) PL663 or PL665 module ***(See Sections 14.15. & 16.4.)***

PIC16C926	Microchip	No	100uS	x 3	13.0	
PIC16LC926	Microchip	No	100uS	x 3	13.0	

Device	*Make*	*Int. Ident.*	*Programming Parameters*			*Comment*
			Initial Pulse	*Overprog. Multiplier*	*Vpp V*	

- Size 0 - 00001FFF (14 bit) PL651, PL652, PL665 or PL667 module ***(See Sections 14.15. & 16.4.)***

PIC16C66	Microchip	No	100uS	x 3	13.0	

- Size 0 - 00001FFF (14 bit) PL651, PL654, PL662, PL664 or PL665 module
(See Sections 14.15. & 16.4.)

PIC16C67	Microchip	No	100uS	x 3	13.0	

- Size 0 - 00001FFF (14 bit) PL651, PL652, PL665 or PL667 module ***(See Sections 14.15. & 16.4.)***

PIC16C745	Microchip	No	100uS	x 3	13.0	
PIC16C76	Microchip	No	100uS	x 3	13.0	

- Size 0 - 00001FFF (14 bit) PL651, PL654, PL662, PL664 or PL665 module
(See Sections 14.15. & 16.4.)

PIC16C77	Microchip	No	100uS	x 3	13.0	

- Size 0 - 1FF (12 bit) PL655, PL657, PL660 or PL665 module ***(See Sections 14.15. & 16.4.)***

PIC16F54	Microchip	No	2mS	x 0	13.0	
PIC16LF54	Microchip	No	2mS	x 0	13.0	

- Size 0 - 3FF (12 bit) PL659, PL679 or PL665 module ***(See Sections 14.15. & 16.4.)***

PIC16F610	Microchip	No	3.5mS	x 0	12.0	

- Size 0 - 7FF (12 bit) PL659, PL679 or PL665 module ***(See Sections 14.15. & 16.4.)***

PIC16F616	Microchip	No	3.5mS	x 0	12.0	

- Size 0 - 000003FF (14 bit) PL655, PL657, PL660 or PL665 module ***(See Sections 14.15. & 16.4.)***

PIC16F627	Microchip	No	8mS	x 0	13.0	
PIC16LF627	Microchip	No	8mS	x 0	13.0	
PIC16F627A	Microchip	No	2.5mS	x 0	13.0	
PIC16LF627A	Microchip	No	2.5mS	x 0	13.0	
PIC16F818	Microchip	No	2mS	x 0	13.0	
PIC16LF818	Microchip	No	2mS	x 0	13.0	

- Size 0 - 000007FF (14 bit) PL655, PL657, PL660 or PL665 module ***(See Sections 14.15. & 16.4.)***

PIC16F628	Microchip	No	8mS	x 0	13.0	
PIC16LF628	Microchip	No	8mS	x 0	13.0	
PIC16F628A	Microchip	No	2.5mS	x 0	13.0	

PIC16LF628A	Microchip	No	2.5mS	x 0	13.0	
PIC16F819	Microchip	No	2mS	x 0	13.0	
PIC16LF819	Microchip	No	2mS	x 0	13.0	

- Size 0 - 00000FFF (14 bit) **PL655, PL657, PL660 or PL665 module** ***(See Sections 14.15. & 16.4.)***

PIC16F648A	Microchip	No	2.5mS	x 0	13.0	
PIC16LF648A	Microchip	No	2.5mS	x 0	13.0	
PIC16F87	Microchip	No	1mS	x 0	13.0	
PIC16LF87	Microchip	No	1mS	x 0	13.0	
PIC16F88	Microchip	No	1mS	x 0	13.0	
PIC16LF88	Microchip	No	1mS	x 0	13.0	

Device	*Make*	*Int. Ident.*	*Programming Parameters*			*Comment*
			Initial Pulse	*Overprog. Multiplier*	*Vpp V*	

- Size 0 - 000001FF (14 bit) **PL650 module** ***(See Sections 14.4. & 16.4.)***

PIC16F83	Microchip	No	10mS	x 0	13.0	Group 1

- Size 0 - 000003FF (14 bit) **PL650 module** ***(See Sections 14.4. & 16.4.)***

PIC16C84	Microchip	No	10mS	x 0	13.0	Group 1

- Size 0 - 000003FF (14 bit) **PL650 module** ***(See Sections 14.4. & 16.4.)***

PIC16F84	Microchip	No	10mS	x 0	13.0	Group 1

- Size 0 - 000003FF (14 bit) **PL650, PL655, PL657, PL660 or PL665 module (*See Sections 14.4. & 16.4.)***

PIC16F84A	Microchip	No	4mS	x 0	13.0	Group 1

- Size 0 - 000007FF (14 bit) **PL659, PL679 or PL665 module** **(*See Sections 14.15. & 16.4.)***

PIC16F677	Microchip	No	2.5mS	x 0	11.0	
PIC16F684	Microchip	No	2.5mS	x 0	11.0	
PIC16F687	Microchip	No	2.5mS	x 0	11.0	

- Size 0 - 000007FF (14 bit) **PL651, PL652, PL665 or PL667 module** ***(See Sections 14.15. & 16.4.)***

PIC16F72	Microchip	No	3mS	x 0	13.0	
PIC16LF72	Microchip	No	3mS	x 0	13.0	
PIC16F722	Microchip	No	2mS	x 0	8.5	Page Write
PIC16LF722	Microchip	No	2mS	x 0	8.5	Page Write
PIC16F870	Microchip	No	6mS	x 0	13.0	
PIC16LF870	Microchip	No	6mS	x 0	13.0	
PIC16F871	Microchip	No	6mS	x 0	13.0	
PIC16LF871	Microchip	No	6mS	x 0	13.0	
PIC16F872	Microchip	No	6mS	x 0	13.0	
PIC16LF872	Microchip	No	6mS	x 0	13.0	

- Size 0 – 00000FFF (14 bit) **PL659, PL679 or PL665 module** **(*See Sections 14.15. & 16.4.)***

PIC16F685	Microchip	No	2.5mS	x 0	11.0	
PIC16F688	Microchip	No	2.5mS	x 0	11.0	
PIC16F689	Microchip	No	2.5mS	x 0	11.0	
PIC16F690	Microchip	No	2.5mS	x 0	11.0	

- Size 0 - 00000FFF (14 bit) **PL651, PL652, PL665 or PL667 module** ***(See Sections 14.15. & 16..4)***

PIC16F1933	Microchip	No	2.5mS	x 0	8.5	Page Write
PIC16LF1933	Microchip	No	2.5mS	x 0	8.5	Page Write
PIC16F723	Microchip	No	2mS	x 0	8.5	Page Write
PIC16LF723	Microchip	No	2mS	x 0	8.5	Page Write
PIC16F73	Microchip	No	1mS	x 0	13.0	
PIC16LF73	Microchip	No	1mS	x 0	13.0	
PIC16F873	Microchip	No	6mS	x 0	13.0	
PIC16LF873	Microchip	No	6mS	x 0	13.0	
PIC16F873A	Microchip	No	4mS	x 0	13.0	Page Write
PIC16LF873A	Microchip	No	4mS	x 0	13.0	Page Write
PIC16F883	Microchip	No	2.5mS	x 0	11.0	Page Write
PIC16LF883	Microchip	No	2.5mS	x 0	11.0	Page Write

Device	***Make***	***Int. Ident.***	***Programming Parameters***			***Comment***
			Initial Pulse	***Overprog. Multiplier***	***Vpp V***	
- Size 0 - 00000FFF (14 bit)		**PL651, PL654, PL662, PL664 or PL665 module**				
					(See Sections 14.15. & 16.4.)	
PIC16F1934	Microchip	No	2.5mS	x 0	8.5	Page Write
PIC16LF1934	Microchip	No	2.5mS	x 0	8.5	Page Write
PIC16F724	Microchip	No	2mS	x 0	8.5	Page Write
PIC16LF724	Microchip	No	2mS	x 0	8.5	Page Write
PIC16F74	Microchip	No	1mS	x 0	13.0	
PIC16LF74	Microchip	No	1mS	x 0	13.0	
PIC16F874	Microchip	No	6mS	x 0	13.0	
PIC16LF874	Microchip	No	6mS	x 0	13.0	
PIC16F874A	Microchip	No	4mS	x 0	13.0	Page Write
PIC16LF874A	Microchip	No	4mS	x 0	13.0	Page Write
PIC16F884	Microchip	No	2.5mS	x 0	11.0	Page Write
PIC16LF884	Microchip	No	2.5mS	x 0	11.0	Page Write
PIC16F913	Microchip	No	3.5mS	x 0	11.0	Page Write
PIC16LF913	Microchip	No	3.5mS	x 0	11.0	Page Write
PIC16F914	Microchip	No	3.5mS	x 0	11.0	Page Write
PIC16LF914	Microchip	No	3.5mS	x 0	11.0	Page Write
- Size 0 - 00001FFF (14 bit)		**PL651, PL652, PL665 or PL667 module**			***(See Sections 14.15. & 16.4.)***	
PIC16F1936	Microchip	No	2.5mS	x 0	8.5	Page Write
PIC16LF1936	Microchip	No	2.5mS	x 0	8.5	Page Write
PIC16F726	Microchip	No	2mS	x 0	8.5	Page Write
PIC16LF726	Microchip	No	2mS	x 0	8.5	Page Write
PIC16F876	Microchip	No	6mS	x 0	13.0	
PIC16LF876	Microchip	No	6mS	x 0	13.0	
PIC16F876A	Microchip	No	4mS	x 0	13.0	Page Write
PIC16LF876A	Microchip	No	4mS	x 0	13.0	Page Write
PIC16F886	Microchip	No	2.5mS	x 0	11.0	Page Write
PIC16LF886	Microchip	No	2.5mS	x 0	11.0	Page Write
- Size 0 - 00001FFF (14 bit)		**PL651, PL654, PL662, PL664 or PL665 module**				
					(See Sections 14.15. & 16.4.)	
PIC16F1937	Microchip	No	2.5mS	x 0	8.5	Page Write
PIC16LF1937	Microchip	No	2.5mS	x 0	8.5	Page Write
PIC16F727	Microchip	No	2mS	x 0	8.5	Page Write
PIC16LF727	Microchip	No	2mS	x 0	8.5	Page Write
PIC16F76	Microchip	No	1mS	x 0	13.0	

PIC16LF76	Microchip	No	1mS	x 0	13.0	
PIC16F77	Microchip	No	1mS	x 0	13.0	
PIC16LF77	Microchip	No	1mS	x 0	13.0	
PIC16F877	Microchip	No	6mS	x 0	13.0	
PIC16LF877	Microchip	No	6mS	x 0	13.0	
PIC16F887	Microchip	No	2.5mS	x 0	11.00	Page Write
PIC16LF887	Microchip	No	2.5mS	x 0	11.00	Page Write
PIC16F877A	Microchip	No	4mS	x 0	13.0	Page Write
PIC16LF877A	Microchip	No	4mS	x 0	13.0	Page Write
PIC16F916	Microchip	No	3.5mS	x 0	11.00	Page Write
PIC16LF916	Microchip	No	3.5mS	x 0	11.00	Page Write
PIC16F917	Microchip	No	3.5mS	x 0	11.00	Page Write
PIC16LF917	Microchip	No	3.5mS	x 0	11.00	Page Write
Device	***Make***	***Int. Ident.***	***Programming Parameters***			***Comment***
			Initial Pulse	***Overprog. Multiplier***	***Vpp V***	

- Size 0 - 00003FFF PL651, PL652, PL665 or PL667 module ***(See Sections 14.15. & 16.4.)***

PIC16F1938	Microchip	No	2.5mS	x 0	8.5	Page Write
PIC16LF1938	Microchip	No	2.5mS	x 0	8.5	Page Write

- Size 0 - 00003FFF PL651, PL654, PL662, PL664 or PL665 module
(See Sections 14.15. & 16.4.)

PIC16F1939	Microchip	No	2.5mS	x 0	8.5	Page Write
PIC16LF1939	Microchip	No	2.5mS	x 0	8.5	Page Write

- Size 0 - 000007FF (16 bit) PL650 or PL658 module ***(See Sections 14.4. & 16.4.)***

PIC17C42	Microchip	No	100uS	x 3	11.0	Group 4
PIC17C42_LP	Microchip	No	100uS	x 3	11.0	
PIC17C42_RC	Microchip	No	100uS	x 3	11.0	
PIC17C42_XT	Microchip	No	100uS	x 3	11.0	
PIC17C42A	Microchip	No	100uS	x 3	13.0	Group 4
PIC17C42A_LP	Microchip	No	100uS	x 3	13.0	
PIC17C42A_RC	Microchip	No	100uS	x 3	13.0	
PIC17C42A_XT	Microchip	No	100uS	x 3	13.0	

- Size 0 - 00000FFF (16 bit) PL650 or PL658 module ***(See Sections 14.4. & 16.4.)***

PIC17C43	Microchip	No	100uS	x 3	13.0	Group 4
PIC17C43_LP	Microchip	No	100uS	x 3	13.0	
PIC17C43_RC	Microchip	No	100uS	x 3	13.0	
PIC17C43_XT	Microchip	No	100uS	x 3	13.0	

- Size 0 - 00001FFF (16 bit) PL650 or PL658 module ***(See Sections 14.4. & 16.4.)***

PIC17C44	Microchip	No	100uS	x 3	13.0	Group 4
PIC17C44_LP	Microchip	No	100uS	x 3	13.0	
PIC17C44_RC	Microchip	No	100uS	x 3	13.0	
PIC17C44_XT	Microchip	No	100uS	x 3	13.0	

- Size 0 - 00001FFF (16 bit) PL656 module ***(See Sections 14.4. & 16.4.)***

PIC17C752	Microchip	No	100uS	x 3	13.0	

- Size 0 - 00003FFF (16 bit) PL656 module ***(See Sections 14.4. & 16.4.)***

PIC17C756	Microchip	No	100uS	x 3	13.0	
PIC17C756A	Microchip	No	100uS	x 3	13.0	

- Size 0 - 000007FF (16 bit)		**PL655, PL657, PL660 or PL665 module**				***(See Sections 14.15. & 16.4.)***
PIC18F1220	Microchip	No	1mS	x 0	13.0	Uses 32M RAM
- Size 0 - 000007FF (16 bit)		**PL651, PL652, PL665 or PL667 module**				***(See Sections 14.15. & 16.4.)***
PIC18F2220	Microchip	No	1mS	x 0	13.0	Uses 32M RAM
- Size 0 - 000007FF (16 bit)		**PL651, PL654, PL662, PL664 or PL665 module**				***(See Sections 14.15. & 16.4.)***
PIC18F4220	Microchip	No	1mS	x 0	13.0	Uses 32M RAM
- Size 0 - 00000FFF (16 bit)		**PL655, PL657, PL660 or PL665 module**				***(See Sections 14.15. & 16.4.)***
PIC18F1320	Microchip	No	1mS	x 0	13.0	Uses 32M RAM
Device	***Make***	***Int. Ident.***	***Programming Parameters Initial Pulse***	***Overprog. Multiplier***	***Vpp V***	***Comment***
- Size 0 - 00000FFF (16 bit)		**PL679 or PL665 module** ***(See Sections 14.15. & 16.4.)***				
PIC18F13K22	Microchip	No	1mS	x 0	8.5	Uses 32M RAM
- Size 0 - 00001FFF (16 bit)		**PL679 or PL665 module** ***(See Sections 14.15. & 16.4.)***				
PIC18F14K22	Microchip	No	1mS	x 0	8.5	Uses 32M RAM
- Size 0 - 00000FFF (16 bit)		**PL651, PL652, PL665 or PL667 module** ***(See Sections 14.15. & 16.4.)***				
PIC18F2320	Microchip	No	1mS	x 0	13.0	Uses 32M RAM
PIC18F2331	Microchip	No	1mS	x 0	13.0	Uses 32M RAM
- Size 0 - 00000FFF (16 bit)		**PL651, PL654, PL662, PL664 or PL665 module**				***(See Sections 14.15. & 16.4.)***
PIC18F4320	Microchip	No	1mS	x 0	13.0	Uses 32M RAM
PIC18F4331	Microchip	No	1mS	x 0	13.0	Uses 32M RAM
- Size 0 - 00001FFF (16 bit)		**PL651, PL652, PL665 or PL667 module** ***(See Sections 14.15. & 16.4.)***				
PIC18C242	Microchip	No	1mS	x 0	13.0	Uses 32M RAM
PIC18F242	Microchip	No	1mS	x 0	13.0	Uses 32M RAM
PIC18F248	Microchip	No	1mS	x 0	13.0	Uses 32M RAM
PIC18F2410	Microchip	No	1mS	x 0	12.0	Uses 32M RAM
PIC18F2420	Microchip	No	1mS	x 0	12.0	Uses 32M RAM
PIC18F24K20	Microchip	No	1mS	x 0	8.5	Uses 32M RAM
PIC18F2423	Microchip	No	1mS	x 0	12.0	Uses 32M RAM
PIC18F2431	Microchip	No	1mS	x 0	13.0	Uses 32M RAM **+**

*(**+ Note that** a hardware modification is required when using this device in a PL651 or PL652 module shipped before June 2005. Section 14.15 refers.)*

- Size 0 - 00001FFF (16 bit)		**PL651, PL654, PL662, PL664 or PL665 module**				***(See Sections 14.15. & 16.4.)***
PIC18C442	Microchip	No	1mS	x 0	13.0	Uses 32M RAM
PIC18F442	Microchip	No	1mS	x 0	13.0	Uses 32M RAM
PIC18F448	Microchip	No	1mS	x 0	13.0	Uses 32M RAM
PIC18F4410	Microchip	No	1mS	x 0	12.0	Uses 32M RAM
PIC18F4420	Microchip	No	1mS	x 0	12.0	Uses 32M RAM
PIC18F44K20	Microchip	No	1mS	x 0	8.5	Uses 32M RAM
PIC18F4423	Microchip	No	1mS	x 0	12.0	Uses 32M RAM

PIC18F4431	Microchip	No	1mS	x 0	13.0	Uses 32M RAM

- Size 0 - 00002FFF (16 bit) PL651, PL652, PL665 or PL667 module ***(See Sections 14.15. & 16.4.)***

PIC18F2455	Microchip	No	1mS	x 0	12.0	Uses 32M RAM
PIC18F2458	Microchip	No	1mS	x 0	12.0	Uses 32M RAM

- Size 0 - 00003FFF (16 bit) PL651, PL652, PL665 or PL667 module ***(See Sections 14.15. & 16.4.)***

PIC18C252	Microchip	No	1mS	x 0	13.0	Uses 32M RAM
PIC18F2480	Microchip	No	1mS	x 0	12.0	Uses 32M RAM
PIC18F252	Microchip	No	1mS	x 0	13.0	Uses 32M RAM
PIC18F258	Microchip	No	1mS	x 0	13.0	Uses 32M RAM
PIC18F2520	Microchip	No	1mS	x 0	12.0	Uses 32M RAM
PIC18LF2520	Microchip	No	1mS	x 0	12.0	Uses 32M RAM Vcc=3.3

Device	***Make***	***Int. Ident.***	***Programming Parameters***			***Comment***
			Initial Pulse	***Overprog. Multiplier***	***Vpp V***	

- Size 0 - 00003FFF (16 bit) PL651, PL652, PL665 or PL667 module ***(See Sections 14.15. & 16.4.)*** ***(Cont.)***

PIC18F25K20	Microchip	No	1mS	x 0	8.5	Uses 32M RAM
PIC18F2523	Microchip	No	1mS	x 0	12.0	Uses 32M RAM
PIC18F2550	Microchip	No	1mS	x 0	12.0	Uses 32M RAM
PIC18F2553	Microchip	No	1mS	x 0	12.0	Uses 32M RAM
PIC18F2580	Microchip	No	1mS	x 0	12.0	Uses 32M RAM

- Size 0 - 00003FFF (16 bit) PL651, PL654, PL662, PL664 or PL665 module ***(See Sections 14.15. & 16.4.)***

PIC18C452	Microchip	No	1mS	x 0	13.0	Uses 32M RAM
PIC18F4480	Microchip	No	1mS	x 0	12.0	Uses 32M RAM
PIC18F452	Microchip	No	1mS	x 0	13.0	Uses 32M RAM
PIC18F458	Microchip	No	1mS	x 0	13.0	Uses 32M RAM
PIC18F4520	Microchip	No	1mS	x 0	12.0	Uses 32M RAM
PIC18F45K20	Microchip	No	1mS	x 0	8.5	Uses 32M RAM
PIC18F4523	Microchip	No	1mS	x 0	12.0	Uses 32M RAM
PIC18F4550	Microchip	No	1mS	x 0	12.0	Uses 32M RAM
PIC18F4553	Microchip	No	1mS	x 0	12.0	Uses 32M RAM
PIC18F4580	Microchip	No	1mS	x 0	12.0	Uses 32M RAM

- Size 0 - 00003FFF (16 bit) PL665 or PL666 module ***(See Sections 14.15. & 16.4.)***

PIC18C858	Microchip	No	1mS	x 0	13.0	Uses 32M RAM

- Size 0 - 00005FFF (16 bit) PL651, PL652, PL665 or PL667 module ***(See Sections 14.15. & 16.4.)***

PIC18F2515	Microchip	No	1mS	x 0	12.0	Uses 32M RAM
PIC18F2525	Microchip	No	1mS	x 0	12.0	Uses 32M RAM
PIC18F2585	Microchip	No	1mS	x 0	12.0	Uses 32M RAM

- Size 0 - 00005FFF (16 bit) PL651, PL654, PL662, PL664 or PL665 module ***(See Sections 14.15. & 16.4.)***

PIC18F4515	Microchip	No	1mS	x 0	12.0	Uses 32M RAM
PIC18F4525	Microchip	No	1mS	x 0	12.0	Uses 32M RAM
PIC18F4585	Microchip	No	1mS	x 0	12.0	Uses 32M RAM

- Size 0 - 00002FFF (16 bit) PL651, PL654, PL662, PL664 or PL665 module ***(See Sections 14.15. & 16.4.)***

PIC18F4455	Microchip	No	1mS	x 0	12.0	Uses 32M RAM
PIC18F4458	Microchip	No	1mS	x 0	12.0	Uses 32M RAM

- Size 0 - 00007FFF (16 bit) PL651, PL652, PL665 or PL667 module ***(See Sections 14.15. & 16.4.)***

PIC18F2610	Microchip	No	1mS	x 0	12.0	Uses 32M RAM
PIC18F2620	Microchip	No	1mS	x 0	12.0	Uses 32M RAM
PIC18F26K20	Microchip	No	1mS	x 0	8.5	Uses 32M RAM
PIC18F2680	Microchip	No	1mS	x 0	12.0	Uses 32M RAM

Device	***Make***	***Int. Ident.***	***Programming Parameters***			***Comment***
			Initial Pulse	***Overprog. Multiplier***	***Vpp V***	

- Size 0 - 00007FFF (16 bit) PL651, PL654, PL662, PL664 or PL665 module ***(See Sections 14.15. & 16.4.)***

PIC18F4610	Microchip	No	1mS	x 0	12.0	Uses 32M RAM
PIC18F4620	Microchip	No	1mS	x 0	12.0	Uses 32M RAM
PIC 18F46K20	Microchip	No	1mS	x 0	8.5	Uses 32M RAM
PIC18F4680	Microchip	No	1mS	x 0	12.0	Uses 32M RAM

- Size 0 - 00001FFF PL320 module

TMS320E	Texas	No	1mS	x 3	12.5	

- Size 0 - 00000FFF x 8 PL800 module

PSD301	No longer supported
PSD311	No longer supported

- Size 0 - 00001FFF x 8 PL800 module

PSD302	No longer supported
PSD312	No longer supported

- Size 0 - 00003FFF x 8 PL800 module

PSD303	No longer supported
PSD313	No longer supported

EPROM CARDS

- Size 0 - 00001FFF PL900 module

A_4131F	Cardpro	Yes	1mS	x 0	5.0	Deactivate SDP
A_4131F_Page	Cardpro	Yes	1mS	x 0	5.0	Deactivate SDP

- Size 0 - 00003FFF PL950 module

647128E_FAST	Microchip	Yes	1mS	x 3	12.5	
647128E_EXP	Microchip	Yes	100uS	x 0	13.0	

- Size 0 - 00007FFF PL950 module

647256E_FAST	Microchip	Yes	1mS	x 3	12.5	
647256E_EXP	Microchip	Yes	100uS	x 0	13.0	
E_222T	Jactron	Yes	100uS	x 0	12.7	MXIC

- Size 0 - 0000FFFF PL950 module

470512M_FAST	Delamere	Yes	1mS	x 3	12.5	Microchip
470512M_EXP	Delamere	Yes	100uS	x 0	13.0	Microchip
470512S	Delamere	Yes	100uS	x 0	12.7	Signetics
470512X	Delamere	Yes	100uS	x 0	12.7	MXIC
647512E_FAST	Microchip	Yes	1mS	x 3	12.5	
647512E_EXP	Microchip	Yes	100uS	x 0	13.0	
E_234V	MIPS	Yes	100uS	x 0	12.7	Toshiba
PC1X512EM	Datakey	Yes	100uS	x 0	13.0	Microchip

Device	*Make*	*Int. Ident.*	*Programming Parameters* *Initial*	*Overprog.*	*Vpp*	*Comment*

- Size 0 - 0000FFFF PL900 module

A_234	Microtech	Yes	100uS	x 0	12.7	MXIC
A_234A	MIPS	Yes	100uS	+ A/R	13.0	Atmel
A_234V	MIPS	No	100uS	x 0	12.7	NEC

- Size 0 - 0000FFFF x 2 PL950 module

647512X2E_F	Microchip	Yes	1mS	x 3	12.5	
647512X2E_E	Microchip	Yes	100uS	x 0	13.0	
E_242L	MIPS	No	100uS	+ 100uS	12.7	

- Size 0 - 0001FFFF PL950 module

128F01_1	VIX	Yes	10uS	x 0	12.7	Flash Mitsubishi
128F01_2	VIX	Yes	10uS	x 0	12.7	Flash Intel
128P01	VIX	Yes	200uS	x 0	12.5	Mitsubishi
47001MA	Delamere	Yes	100uS	x 0	13.0	Atmel
47001MS	Delamere	Yes	100uS	x 0	12.7	Signetics
47001MX	Delamere	Yes	100uS	x 0	12.7	MXIC

- Size 0 - 0001FFFF PL900 module

A_244	Microtech	Yes	100uS	x 0	12.7	MXIC
A_244L	MIPS	No	100uS	+ 100uS	12.7	Sharp
A_244M	MIPS	Yes	200uS	x 1	12.5	Mitsubishi
A_244MY	MIPS	Yes	100uS	x 0	12.5	NEC
A_244V	MIPS	Yes	100uS	x 0	12.7	Toshiba
MC20009	Centennial	Yes	100uS	x 0	12.7	National
MC20070	Centennial	Yes	100uS	x 0	12.7	Atmel
A_344M_29	MIPS	Yes	10uS	x 0	12.0	Flash MXIC
A_344M	MIPS	Yes	10uS	x 0	12.0	Flash Intel

- Size 0 - 0001FFFF PL950 module

E_244T	MIPS	Yes	100uS	x 0	12.7	Signetics
E_244V	MIPS	Yes	100uS	x 0	12.7	Toshiba
E_244MY	MIPS	Yes	100uS	x 0	12.5	NEC
E_344	MIPS	Yes	10uS	x 0	12.0	Flash MXIC
E_344M	MIPS	Yes	10uS	x 0	12.0	Flash Intel
E_344ST	Cardpro	Yes	10uS	x 0	12.0	Flash ST
E_344T	Jactron	Yes	10uS	x 0	12.0	Flash MXIC

- Size 0 - 0003FFFF PL900 module

A_254	Microtech	Yes	100uS	x 0	12.7	MXIC
A_254M	MIPS	No	100uS	x 0	12.5	NEC

- Size 0 - 0003FFFF PL900 module

A_354M	MIPS	Yes	10uS	x 0	12.0	Flash Intel

- Size 0 - 0003FFFF PL950 module

E_354MX	Jactron	Yes	10uS	x 0	12.0	Flash MXIC
E_354ST	Cardpro	Yes	10uS	x 0	12.0	Flash ST

Device	*Make*	*Int. Ident.*	*Programming Parameters* *Initial*	*Overprog.*	*Vpp*	*Comment*

- Size 0 - 0007FFFF PL900 Mk2 module

A_264MA	MIPS	Yes	100uS	+ A/R	13.0	Atmel
A_264CP	MIPS	Yes	100uS	x 0	12.7	MXIC

- Size 0 - 0007FFFF PL950 Mk2 module

E_264	Jactron	Yes	100uS	x 0	12.7	MXIC
E_264AT	Centennial	Yes	100uS	+ A/R	13.0	Atmel
E_264ST	Centennial	Yes	100uS	x 0	12.7	ST
E_36489CP	Cardpro	Yes	Polled		12.0	Flash Intel

- Size 0 - 000000FF PL950 Mk2 module

E_264_SER	Jactron	Yes	100uS	x 0	12.7	MXIC
E_264AT_SER	Centennial	Yes	100uS	+ A/R	13.0	Atmel
E_264ST_SER	Centennial	Yes	100uS	x 0	12.7	ST

- Size 0 - 000FFFFF PL950 Mk2 module

E_374I	Centennial	Yes	Polled		12.0	Flash Intel
E_374M	MIPS	Yes	Polled		12.0	Flash Intel
E_374MXIC	MIPS	Yes	Polled		5.0	Flash MXIC

- Size 0 - 001FFFFF PL950 Mk2 module

E_38489	Cardpro	Yes	Polled		12.0	Flash Intel

15.3. Programming Parameter Notes

General

Some devices such as Texas 27C010 and 27C210 use a two pass programming algorithm. These devices are only programmed on the second pass as required (A/R).

Some devices such as Texas 27C010 and many flash parts are programmed with more than one byte at a time - page programming. This speeds up the programming sequence.

Devices such as Texas 27C292 can be programmed and verified using the M9000. These devices use a differential cell construction in order to achieve high speed. They are ideal replacements for bi-polar proms. However, the M9000 cannot check the erased state of these devices before attempting to program them. The *Blank Check* function is also suppressed.

Devices such as NEC 27HC65 cannot distinguish an erased bit as either high or low during a normal read or verify cycle. It is, therefore, only possible to blank check or program an erased part.

Intel 87C51FC was added to the device list in November 1991. PL875 and PL876 modules shipped before this date require a small hardware modification to accommodate this device. However, all modules shipped after this date can program this device. The Intel 87C51 FX family of devices added in November 1992 requires the same hardware modification as the 87C51FC.

Atmel 89C55 was added to the device list in July 1997. PL875 and PL876 modules shipped before this date require a small hardware modification to accommodate this device. However, all modules shipped after this date can program this device. Please refer to Section 14.7. for the modification details required.

Greenwich Emulator Modules

The Greenwich emulator modules contain a ram chip and a battery. They are programmed as eproms but the illegal bit test is omitted. Once programmed, they behave as eproms and can, therefore, be used as a low cost emulation system.

DO NOT ATTEMPT TO PROGRAM GREENWICH NV RAMS USING THE EMULATOR SETTING BECAUSE THEY WILL BE DAMAGED.

ST Eproms

In September 2000, ST changed the verification levels for most of their eproms to 4.2V and 6.0V. This change was implemented in Software Version 5.0C. The programming algorithms were also changed for 27C512 and 27C801. However, ST devices made before this date have the same part number and identifier codes. According to ST, earlier devices should pass verification at these levels but they cannot be guaranteed to pass. The odd device which fails to verify could be verified using the equivalent Intel setting. ***Note that*** the Identifier Check may need to be turned off during device selection.

Page Mode Devices

Some devices can only be programmed in page mode. While this is very quick, there is sometimes a limitation on the set size which can be programmed, e.g. Atmel 29C256 can be programmed as a 2 IC set but not with 4 or 8 devices per set. If a set of perhaps 4 devices needs to be programmed, the RAM must be split before programming. It is, however, then necessary to program two sets with 16 bit data.

Sector Protection (_SP)

The M9000 can stop some parts of flash devices being programmed 'in system' by protecting individual sectors of a device. Details of individual devices are given in manufacturers' data sheets. Devices which can be (un)protected in this way have a suffix **_SP** added to the part number for device selection. ***Note that*** sector protection is ***NOT*** changed or checked for part numbers which do ***NOT*** have a suffix **_SP** except for the device type - Sharp 28F400SUL - which is unprotected in the erase cycle. As the M9000 can over-ride the normal sector protection for some devices by using temporary sector unprotect, even protected sectors can be programmed on the M9000. Therefore, in order to check that a part is fully erased ***INCLUDING SECTOR PROTECTION,*** the user must select the part number with a suffix **_SP**.

There are two versions of the AMD 29F040.

1. 29F040 98401 die manufactured with AMD's 0.85 um process CS-19AFDS, and
2. 29F040 98403 die manufactured with AMD's 0.5 um process CS-19AF.

Note that version 2 uses the same sector unprotect algorithm as the 29F040B AMD. If the wrong type is selected, the message – '*Use device type 29F040B*' - will be displayed. All three devices have the same identifier codes. The same algorithm is used for programming and device protection. The only difference in the algorithm is the unprotect mechanism.

Temporary Sector Unprotect

Some devices such as AMD 29F400B/T have a temporary sector unprotect facility whereby the device can be erased and programmed while maintaining the level of sector protection.

Software Data Protection (_SDP)

Some eeproms can be protected against accidental writes by means of a 'software lock' known as Software Data Protection. The M9000 unlocks such devices, programs them and relocks them if the part number suffix is **_SP**. No attempt is made to unlock devices unless the part number suffix selected is **_SP**. Therefore, a protected device cannot be programmed or erased using the device setting without the suffix **_SDP**.

16. APPLICATION NOTES

16.1. Remote Control Of M9000

General Philosophy

In general, remote control programs enable a piece of equipment to be driven from a remote computer, typically a PC. In many cases, remote control is used to compensate for an inadequate keyboard and little more. The M9000 is equipped with 31 function keys and, hence, there is little to be gained from such a facility. The approach which we have, therefore, taken enables the user to drive the programmer from a batch file using very simple commands. At one stroke, this provides the R&D user with the facility to download a file at the end of a compilation and to program a set of eproms. On the other hand, the Production Manager can use a batch file to set the required device type, download a particular file and then leave the programmer ready for manual use.

This Application Note outlines the general facilities available on the M9000 and gives some examples. For precise details of commands, the user is referred to the body of the M9000 Instruction Manual.

Remote Control Commands

All remote control commands are straightforward Ascii characters. When the programmer is in 'Remote' mode, any character received at the control port is examined. Unless the programmer is currently downloading data, the character is assumed to be a control instruction. There is a single Ascii character for each of the 31 keys, e.g. Hex keys - 0 to F - are Ascii characters - 0 to F - and P is used for program from RAM, etc. There are a few special commands which enable the user to use a batch file rather than a programming language. These include the ability to set the device type or download format directly. There is also a facility to read back the message currently being displayed. A special command has been introduced to enable the keyboard to be totally disabled in case of inadvertent manual operation.

Control Ports

The M9000 is fitted with a serial and a high speed Centronics port. Remote control commands are always received on whichever port is selected for input. Data and/or remote control prompts are output to whichever port has been selected for output. It is not,

however, recommended that data or prompts are output to the Centronics port unless the programmer is connected to a printer because standard PCs cannot receive data on the Centronics port.

Remote Control Handshaking

Remote control commands are normally echoed back to the sending device. However, in some instances, this may be undesirable. In particular, when using the Centronics port of a PC, it is essential to disable the echo because the standard port of a PC can only transmit data. For some applications, it is desirable to know when the current instruction has finished. This is indicated by the M9000 sending a prompt sign '+' back to the host machine. If this prompt is not required, it can be turned off. In some cases, an instruction may have ended in failure. For example, a device may have failed to program in which case the prompt '+' is changed to '-'. If the controlling machine then sends a '-' to the M9000, the programmer sends a two line message back to the controller showing the appropriate error message. Both lines are prefixed with a '*?*'. In the case of multiple errors, the M9000 always reports the first error. The act of reading the error message clears the error buffer.

Changing From 'Local only' To 'Local and remote' To 'Remote only'

The M9000 always powers up in 'Local only' mode. It can be switched into 'Local and remote' mode in one of three ways -

1. By using the Special Function facility on the programmer.

2. By sending a '^' character to the input port. This character and all future control commands are echoed back. Whenever a command string has been completed, a prompt character is transmitted.

3. By sending a '*%*' character to the input port. This character and all future control commands are ***NOT*** echoed back. The prompt character is also inhibited.

If the Centronics port is selected for output, the M9000 will not accept a remote command to switch into remote mode through the Centronics port because a printer may be connected to it. If the user switches to remote control on the keyboard and if the Centronics port is selected for input, the user must ***NOT*** connect a Centronics printer to the M9000 with this configuration.

The local keyboard can be disabled by sending an '*H*' to the programmer when it is in remote mode at the INITIAL state.

Examples:

These examples assume that a PC using DOS is being used.

EXAMPLE 1

This first example downloads a file of data in Motorola Hex formats over the address range of 0 to 7FFFF Hex, selects a device type - Toshiba 571000 - and programs one device.

A batch file - M9000.BAT - is created using three commands:-

```
REM                         Sets up remote control, starts download, etc.
COPY START.       PRN:
REM                         Downloads the data file - DATA.
COPY DATA.MOT     PRN:
REM                         Sets device type and starts programming.
COPY END.         PRN:
```

The above example uses the Centronics port. To use the COM1 serial RS232 port, replace PRN: by COM1: Naturally, the M9000 input port must be selected correctly using the *SET COMMS* function.

To use the COMPRESS software in the above example, replace the line '*COPY DATA.MOT PRN:*' with '*COMPRESS DATA.MOT PRN:*'

The file – START - contains the following:-

```
%                           ;Enter remote control without echo or prompt.
I                           ;Download.
MG                          ;Motorola format.
0G                          ;Load from address 0.   (The G is for ACCEPT.)
7FFFFG                      ;Load to address 7FFFF.
0G                          ;Load data at ram start 0.
```

The above file would work perfectly satisfactorily but it could be presented more simply on a single line:-

```
%                           ;Enter remote control without echo or prompt.
```

```
IMG0G7FFFFG0G          ;Download in Motorola format from address 0.
                       ;to 7FFFF with data loading at ram start 0.
```

Note that all characters after the '*;*' to the end of line are ignored and can be used for notes.

The file – END - contains the following:-

```
E]G                    ;Select R & D mode.
T"571000 Toshiba"      ;Device type 571000 Toshiba
G                      ;1 IC per set
G                      ;8 bits per word
]]G                    ;Identifier check compatible
P0G                    ;Program from ram start address 0
```

The above file would work perfectly satisfactorily but it could be presented more simply on a single line:-

```
E]G                    ;Select R & D mode.
T"571000 Toshiba"GG]]G ;Device type 571000 Toshiba with 1 IC
                       ;per set and identifier check compatible.
P0G                    ;Program from ram start 0.
```

NOTE: It is essential to have at least one space between the device type and the make. The make must have at least three characters. The first three characters of the make must be as used by the M9000. However, the characters are not case sensitive.

EXAMPLE 2

This example downloads an Intel Hex file, edits four bytes in ram (perhaps a serial number) and then starts programming a single device. The Centronics port is used throughout. As before, three files are used - START.JB1, the data file, and END.JB1.

A batch file - JOB1.BAT - is created using three commands:-

```
REM                         Sets up remote control, starts download, etc.
COPY START.JB1   PRN:
REM                         Downloads the data file - DATA.INT
COPY DATA.INT    PRN:
REM                         Sets device type and starts programming.
COPY END.JB1     PRN:
```

START.JB1

```
 %                          ;Enter remote control without echo or prompt.
IIG0G7FFFG0G                ;Download, Intel format, Load from 0 to 7FFF
                            ;Ram start 0.
```

The next file to be sent is the data file.

Then the last command file - END.JB1 - is sent.

```
E]G                         ;Select R & D mode.
1100G44]56]G                ;Change bytes 100 and 101 to 44 and 56 hex.
T"27c256 tex"GG]]G          ;Set device type to 27C256 Texas. (Lower case
                            ;characters  can be used.  Name must have at
                            ;least 3 characters) with 1 IC per set and
                            ;identifier check compatible.
P0G                         ;Program from ram start 0.
```

EXAMPLE 3

In this example, the serial port is used and it is assumed that the user has written a control program. The example shows the response of the M9000. The application requires a file to be downloaded in Intel Hex and a set of four devices - 27C010 Texas - to be programmed in 16 bit mode.

DATA SENT TO M9000	*DATA FROM M9000*	*COMMENT*
^		Enter remote control with
	+	prompt and echo. The + confirms the M9000 is OK.
%		Turn echo and prompt off.
S]GGGGGGG		Switch input to Centronics.
IIG0G7FFFFG0G		Set up for downloading.
Data in Intel hex.		Data sent to Centronics port.
S[GGGGGGG		Switch back to serial port.
^		Turn prompt and echo on.
	+	
60G7FFFFG/		Checksum RAM from 0 to 7FFFF.
	? Checksum - ram	
	?00000000--0007FFFF = A234	
	+	
T"27C010 TEX"4G[G]]G		Set device type to 4 x 27C010.
	+	16 bit mode and identifier check compatible.
P0G		Program from RAM start 0.
	+	+ indicates devices verified.
L		
	....PPPP	P = devices passed.

EXAMPLE 4

This example downloads a file of data in Binary (no header) format over the address range of 0 to 7FFFF Hex, selects a device type - Toshiba 571000 - and programs one device.

A batch file - JB4.BAT - is created using three commands:-

```
REM                         Sets up remote control, starts download, etc.
COPY START.JB4   PRN:
REM                         Downloads the data file - DATA.BIN - as Binary file.
COPY DATA.BIN   PRN: /B
REM                         Sets device type and starts programming.
COPY END.          PRN:
```

The file - START.JB4 - contains the following:-

```
%                           ;Enter remote control without echo or prompt.
B                           ;Special function
9                           ;Do remote command after
0A                          ;Hex code for Line feed.  (See note below.)
G                           ;Accept
I                           ;Download
BG                          ;Binary (no header)
0G                          ;Load from address 0  (The G is for ACCEPT.)
7FFFFG                      ;Load to address 7FFFF
0G                          ;Load data at ram start 0
```

The above file would work perfectly satisfactorily but it could be presented more simply as follows:-

```
 %                          ;Enter remote control without echo or prompt.
B90AG                       ;Special function.  (See note below.)
IBG0G7FFFFG0G               ;Download
```

The above Special Function - B90AG - is required for Binary (no header) format files on a PC. This is because the PC generates a 'Line feed' character after the 'Carriage return'. If the Special Function was not set up, the M9000 would receive the 'Line feed' character as the first byte of the binary file.

Please ***note that***, when the '*Do remote command after*' is used to change the character from 'Carriage return' - 0DH - to 'Line feed' - 0AH, every command line must have a comment character '*;*' before the end of line. This will ensure that the carriage return is treated as a comment and not a command.

The file - END - contains the following:-

```
E]G                     ;Select R & D mode.
T"571000 Toshiba"       ;Device type 571000 Toshiba
G                       ;1 IC per set
G                       ;8 bits per word
 ]]G                    ;Identifier check compatible
P0G                     ;Program from ram start address 0
```

The above file would work perfectly satisfactorily but it could be presented more simply on a single line:-

```
E]G                     ;Select R & D mode.
T"571000 Toshiba"GG]]G  ;Device type 571000 Toshiba with 1 IC
                        ;per set and identifier check compatible.
P0G                     ;Program from ram start 0.
```

16.2. Intelligent Identifier Check

Most eproms introduced over the last few years contain two codes. The first code identifies the manufacturer of the eprom and the second code identifies the device type. Unfortunately, the system has not been used rigorously by the eprom manufacturers and, therefore, cannot be relied upon to set the device type. Some 27513s, for example, contain the device code for 27512s! However, it is a useful double check.

If the Intelligent Identifier check were to be implemented as such, it would be very restrictive because, on many occasions, it is possible to mix different device types with different codes. Furthermore, since all eproms of a particular type are read-compatible, it would be illogical to insist that the master has the same programming algorithm as the copies. However, errors could occur if the master device has not been checked. The M9000, therefore, checks the Intelligent Identifier on two levels.

Intelligent Identifier Check for Device Size and Type

Since all eproms of a particular type are read-compatible, the following functions are allowed providing that the selected type and the type being used are in the same family, for example 27256s:

Read master or
Blank check or
Verify with RAM.

It would, therefore, be possible to have a master eprom such as Fujitsu 27C256 to program Texas 27C256s. Similarly, it would be possible to blank check or verify any of these devices on this setting. However, please ***note that*** a few devices are specified to operate at a supply voltage setting of +/- 10% and, if one of these settings is chosen, the blank check and verify will be performed at these limits. The device list shows eproms with a 10% voltage margin. Some devices such as Holtek use a different Intelligent Identifier algorithm and may, therefore, not be read-compatible to other families.

Intelligent Identifier Check for Programming Compatibility

It is very often required to program a number of devices of different makes. If, for example, a test was applied for say Intel 27C256s, parts such as Atmel 27C256 would be rejected because they have a different device code. Their algorithms, however, are exactly the same and, hence, they can be programmed together. One of two methods can be chosen

for testing the Intelligent Identifier when selecting the device. If the '*Yes*' option is chosen, the manufacturer and device code must be exactly correct. This ensures that alternatives cannot be used. If , however, '*Compatible*' is chosen, the M9000 will allow any mix of devices to be programmed together providing that they all use the same algorithm.

The Intelligent Identifier Default

Whenever a new device type is chosen (including remotely), the Intelligent Identifier is set to the default condition (*Yes*, *No* or *Compatible*). The default condition can be set using Special Function 6. When the M9000 is supplied, the default is to check the Intelligent Identifier. The default can be set for each user.

Setting the Intelligent Identifier Check for a Particular Device

When a new device is selected, the check will be set according to the default. If a change is required, the setting can be changed by answering the appropriate question at the end of the *SET TYPE* function. However, please note that this question is only asked if the device is specified as having an Intelligent Identifier. (***See device list.***)

16.3. Programming Eprom Cards

The M9000 provides for two different card formats, namely MIPS A and ECS4 formats. Both cards look identical so the part number is the only means of telling the difference!

There are several important differences between eproms and eprom cards. These differences/procedures are sometimes necessary to ensure that the cards are read and/or programmed correctly. If you require an additional setting, contact your distributor or Lloyd Research Limited.

Where applicable, the Intelligent Identifier is checked. At a later date, the M9000 will read a Binary code from the card which confirms card size and type, etc.

Unlike eproms, the M9000 programs every byte of each card. This means that a marginally programmed card can be over-programmed to make it work correctly. (Unlike eproms, cards cannot be erased.)

Cards are always verified at both high and low Vcc.

Master data can be loaded from a card or an eprom using a PL300 module, etc.

Always return eprom cards to their anti-static wallets and use sensible anti-static handling precautions.

Always make sure devices are fully inserted into their sockets. Do ***NOT*** withdraw them when the red 'LIVE' light is on.

16.4. Programming Lock Bits In Microcontrollers

Software version 2.39 onwards provides facilities for programming the lock bits of microcontrollers such as the 87C51. The choice of whether or not to program these bits must be made when selecting the device type using the *SET TYPE* function.

After successful verification of ***ALL*** devices, the M9000 locks all devices and displays a message - '*Locked @ xxxx*' where xxxx is the checksum indicating successful verification of ***ALL*** devices. ***Note that***,

as each device has different data, it is possible to use the *STEP* keys to display the checksum of each device.

Microchip PICs

As a general rule, there is only one lock bit which scrambles the device data being read. The M9000, therefore, attempts to verify the device after it has been locked. If the verify fails, the lock has worked and the '*Locked @ xxxx*' message is, therefore, displayed. If, however, the device does verify, the lock process has failed so the program run ends with the message – '*Lock fail*'.

Some devices such as the PIC16C622 have more than one lock bit level.

For the PIC16C621:-

Level	***Code Protection***
None	Code Protection off
1	Upper ½ of memory protected
1 & 2	All memory protected

For the PIC16C622:-

Level	***Code Protection***
None	Code Protection off
1	Upper ½ of memory protected
1 & 2	Upper ¾ of memory protected
1, 2, & 3	All of memory protected

Some devices such as PIC16F62x and PIC16F87x have levels of FLASH and EEPROM lock bits. For these devices:-

Level	***Code Protection***
No	Code Protection off
Yes	Select individual lock bits CPx

For the PIC18Fxxx family :- (***See Section 14.16. for details***.)

87C51 Family

As a general rule, the first lock bit stops further programming of the device. As the M9000 is unable to check this, no check is made.

The second lock bit usually stops the device being read. The M9000, therefore, attempts to verify the device after it has been locked. If the verify fails, the lock has worked and the '*Locked @ xxxx*' message is, therefore, displayed. If, however, the device does verify, the lock process has failed so the program run ends with a message – '*Lock fail*'.

Motorola

If the security/lock bit of the Motorola 68HC705C8, 68HC705B16 or Waferscale PSD301 is programmed, ***note that*** the M9000 cannot detect the device in the programming socket.