SECTION 9: FIRMWARE SECURITY

One of the most unique features of the Secure Microcontroller is its firmware security. The family far surpasses the standard offering of ROM based microcontrollers in keeping system attackers or competitors from viewing the contents of memory. In a standard EPROM based microcontroller, a knowledgeable attacker can disable the EPROM security bit and have access to the entire memory contents. The Secure Microcontroller's improved security makes it a natural choice for systems with high security requirements such as financial transaction terminals. However, the firmware security can also be employed to keep competitors from copying proprietary algorithms. Allowing access to these algorithms can create an instant competitor. This section describes the security features and their application. Also included are guidelines to using microcontroller security within the framework of total system security.

As with memory map control, there are variations between the different Secure Microcontroller versions. The original DS5000 has a high level of firmware security and the DS5002 has added several distinct improvements. Note that the DS5001 has only minimal security and should only be applied when other physical security is used or when security is not needed. The table below provides a brief summary of the versions and their security features. A detailed description of each feature follows. In the description, elements that are unique to a particular Secure Microcontroller version have that version underlined.

FEATURE	DS5001	DS5000	DS5002
Security Lock	Yes	Yes	Yes
RAM memory	Yes	Yes	Yes
Encrypted memory	None	Yes, user must enable	Yes
Encryption Key	None	48 bits	64 bits
Encryption Key Selection	None	User selected	True random number
Encryption Keys loaded	N/A	When user selects	Automatic, any new load, dump
Dummy bus access	None	Yes, when encrypted	Yes
On–chip Vector RAM	None	Yes, when encrypted	Yes
Self–Destruct Input	None	None	Yes
Die Top Coating	None	None	Optional (DS5002FPM)
Random Number Generator	Yes	None	Yes

SECURITY OVERVIEW

Security features are useful if an application dispenses services on a pay per service basis. Electronically bypassing the security would allow the dispensing of the service for free, resulting in lost revenue to the system owner. Another common application is the transmission of secret information. The user's algorithm and key data could be observed in a unsecured system, resulting in a break in the secure transmission. The Secure Microcontroller Family is designed to protect the contents of memory from being viewed. This is done with a combination of circuit techniques and physical security. The combination is a formidable defense. Regardless of the application, the secure microcontroller protects the contents of memory from tampering and observation. This preserves secret information, access to services, critical algorithms etc. The security features of the Secure Microcontroller include physical security against probe, memory security through cryptographic scrambling, and memory bus security preventing analysis of the CPU's operation. The features mentioned above and described below protect the application code and data.

SECURITY LOCK

Ordinarily, the easiest way to dump (view) the memory contents of a Secure Microcontroller is using the Bootstrap Loader. On request, the Loader will transfer the contents of memory to a host PC. This is prevented by the Security Lock. The lock is the minimal security feature, available even in the DS5001. Once set, the Security Lock prevents the Loader from gaining access to memory. In fact, no Loader commands (except Unlock) will work while the Lock is set. The Security Lock is similar in function to an EPROM security bit on a single chip microcontroller. It prevents a programmer from reading the memory. In addition, the Security Lock prevents the microcontroller from executing code on the Expanded bus of Ports 0 and 2. Thus an attacker can not add a memory and use MOVC instructions that would force the microcontroller to read out the contents of protected memory. However, the Secure Microcontroller Security Lock does provide one important difference from EPROM security bits. When the Security Lock is cleared, it destroys the RAM contents. If a knowledgeable user were to physically erase the security bit in an EPROM–based microcontroller, the memory contents would remain to be read. The Security Lock consists of a multiple bit latch distributed throughout the microprocessor with circuits that collapse the lock in the event of tampering. Clearing the lock starts an irreversible destructive process that acts differently for each device as described below.

In a DS5001 clearing the lock causes the loader to manually write over the first 32K bytes of NV RAM with zeros. Thus the contents of memory would be erased. This is obviously a low level of security but would deter casual inspection. In a DS5000 or DS5002, clearing the lock causes an instantaneous erasure of the Encryption Key and Vector RAM. This action is unpreventable once the lock is cleared and happens independent of V_{CC} or crystal. Once the erasure has occurred, a DS5000, assumes a non–secure (brand–new) state. In a DS5002, the Loader proceeds to load a new Encryption Key once the erasure has occurred. In both, the Bootstrap Loader will then proceed to overwrite the first 32K bytes of RAM if power is available and the crystal is still present. This last action is for thoroughness. In systems that really require security, the Lock should be combined with Memory Encryption (discussed below). Thus the instantaneous erasure of the Encryption Key renders the contents of memory useless since it can no longer be properly deciphered.

The Security Lock is set via the Bootstrap Loader using the "Z" command. Once issued, the Loader will continue to communicate with a user but will not perform other commands. The Loader will respond with an error message in the event that further commands are issued. While the Lock is set, the Loader has no access to the Byte–wide bus memory. The Security Lock can be cleared using the "U" command. Issuing this command to a locked part results in the destructive process described above. No confirmation is requested. The status of the Security Lock can be read by application software at MCON.0. This bit is only a status flag and can not be affected by the software.

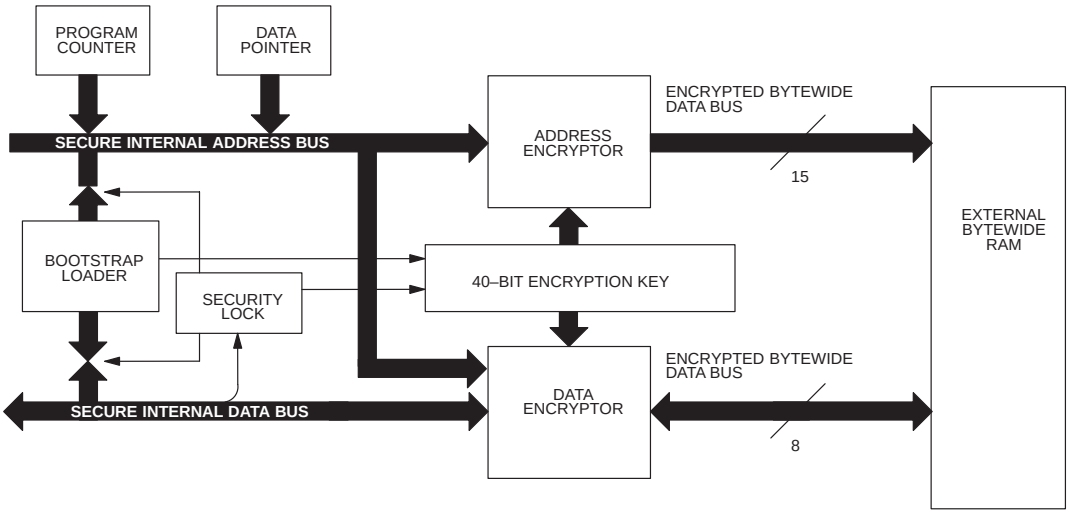
RAM Memory

NV RAM provides a useful way to store program and data. The contents can be retained for a long period, but can be changed when desired. This attribute is important when considering security. No matter what probing techniques are used on a ROM, the contents remain unaffected. With resources and patience, a determined attacker will obtain the contents of a ROM based product. NV RAM can be destroyed on demand. The user's physical security must simply remove the power (V_{CC} and V_{BAT}) from a microprocessor chip to eliminate the memory contents. Thus NV RAM provides flexibility as well as security. Enough physical security can be combined with even a DS5001 to provide a very secure system. The DS5002 even provides a direct facility to destroy memory discussed below.

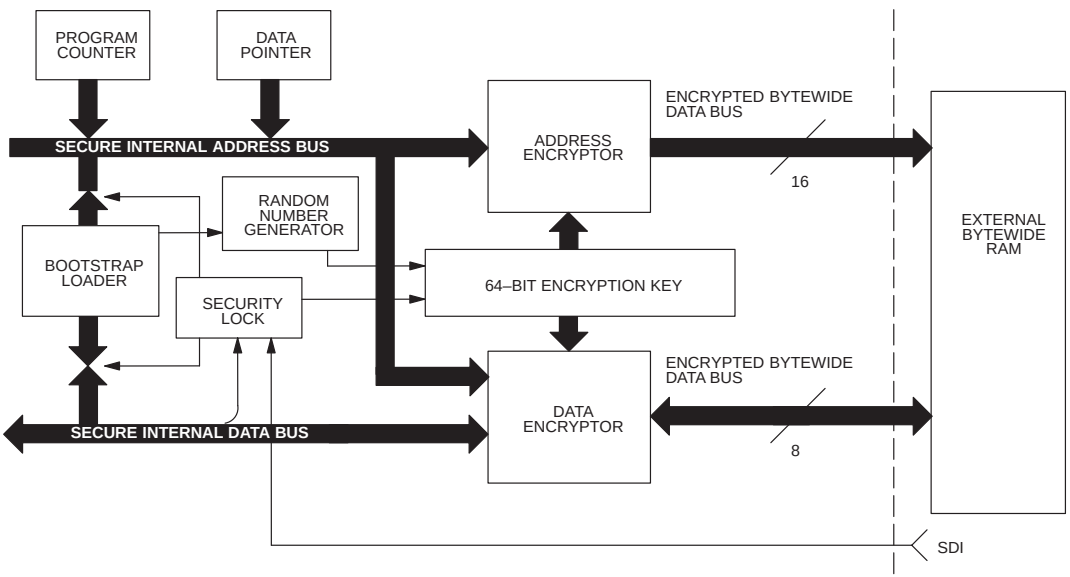
Encrypted Memory

The heart of Secure Microcontroller security is the memory encryption function. Since the NV RAM is visible, the memory contents and memory bus are encrypted. That is, in real time, the addresses and data moving between the RAM and the microcontroller are scrambled by on–chip encryption circuits. Thus an attacker that observes the RAM contents or memory bus will see unintelligible addresses and data. Figure 9–1 shows the conceptual diagram of the memory encryptor for a DS5000 series device. Figure 9–2 shows the encryptor for a DS5002.

DS5000 SOFTWARE ENCRYPTION BLOCK DIAGRAM Figure 9–1



DS5002 SOFTWARE ENCRYPTION BLOCK DIAGRAM Figure 9–2



In a DS5000, the encryption feature is optional. A DS5000 can be locked irrespective of its encryption and encrypted irrespective of the lock. Neither makes much sense by itself. The encryption process is enabled by loading an Encryption Key for the first time. Prior to loading a Key, the DS5000 remains in a non–encrypted state. Once encrypted, the memory interface will remain so until a part is locked, then unlocked. The process of clearing the Security Lock deactivates the encryption circuits. Note that an Encryption Key of zero is still a valid Key. A DS5002 has encryption enabled at all times. No extra steps are required to invoke it. As discussed below, the DS5002 generates its own security Keys.

Encryption logic consists of an address encryptor and a data encryptor using separate but related algorithms. These encryptors are high speed circuits that are transparent to the application software. They are bidirectional and repeatable. That is, addresses and data that are scrambled prior to writing to RAM will be correctly unscrambled when reading in reverse. Each encryptor operates with its own algorithm but both are dependent on the Encryption Key. Encryptors operate while programs are being loaded so that the memory contents are stored in its scrambled form. When program memory is fetched, the process is reversed. Thus the actual program or data is only present in its "true" form while inside the microcontroller.

The address encryptor translates each "logical" address, i.e., the normal sequence of addresses that are generated in the logical flow of a program, into an encrypted address (or physical address) at which the byte is actually stored in RAM. Each time a logical address is generated either during program loading or during execution, the address encryptor circuits use the Encryption Key value and the address itself to form the physical address that will be presented to the RAM on the Byte–wide bus. The encryption algorithm is such that there is one and only one physical address for every possible logical address. The address encryptor operates over the entire memory range.

The Data Encryptor operates in a similar manner to the address encryptor. As each byte including opcode, operand, or data is received during Bootstrap Loading, its value is scrambled prior to storing it in RAM. The value that is actually written in RAM is an encrypted representation. All values that are subsequently stored in RAM during execution also are encrypted. As each byte is read back to the CPU during execution, the internal Data Encryptor restores it to its original value. This encryptor uses the Encryption Key and the data value itself, but also the logical address. Thus the same data with the same Key will have different physical values at different address locations. The data encryption algorithm is repeatable and reversible so that with the same key, data and address, the same encrypted value will be obtained. Note however that there are many possible encrypted data values for each possible true value due to the algorithms dependency on Key and address.

Using the combination of address and data encryption, the normal flow of program code is unintelligible in the NV RAM. What had been a sequential flow of addresses is now apparently random. The values stored in each memory location appear to have no relation to the original data. Another factor that makes analysis more difficult is that all 256 possible values in each memory are valid possibilities. Thus an encrypted value is not only scrambled, but it becomes another potentially valid byte.

Different memory areas are encrypted in the DS5000 and DS5002. For a DS5000, all memory accessed under $\overline{CE1}$ can be encrypted. $\overline{CE2}$ is not encrypted. This allows access to peripherals such as a Real–time Clock to be performed using $\overline{CE2}$.

For the DS5002, encryption is performed on all bytes stored under $\overline{CE1}$ through $\overline{CE4}$. The memory or peripherals accessed by $\overline{PE1}$ through $\overline{PE4}$ on a DS5002 are not encrypted.

Encryption Algorithm

The Secure Microcontroller family uses a proprietary algorithm to encrypt memory. The DS5000FP and DS5002FP use different encryption algorithms. They are the result of improvements made over time in the proprietary encryptor circuits. The original DS5000FP (circa 1988) has the first version of encryptor. This was soon improved with a second version encryptor in 1989, and remains in production today. A substantial improvement was made in the DS5002FP, which uses a wider Key and a more non–linear algorithm. The DS5002FP memory encryptor uses elements of the DES (Data Encryption Standard) although not the entire algorithm. Full DES is impractical as memory encryption must be performed in real–time on a one–to–one substitution and not a block cypher basis. The encryption algorithm is supported by the fact that both address and data are encrypted, the algorithm and key are both secret, the most critical data can be stored on chip in vector RAM (discussed below), and the bus activity is scrambled using dummy access (discussed below). For this reason, a security analysis of the DS5002FP is not simply a mathematical treatment of the encryption algorithm.

Encryption Key

The DS5000FP uses a 40–bit Encryption Key that is stored on–chip. As mentioned above, the Key is the basis of the encryption algorithm. The resulting physical addresses and data are dependent on this value. Tampering with or unlocking the microcontroller will cause the Key to be instantaneously destroyed. If the memory contents are encrypted, they become useless without this Key. A user selects the 40–bit Key and loads it via the Bootstrap Loader. Selecting this Key enables the encryption feature. The DS5002FP uses a 64–bit Key. It is similarly stored on–chip in tamper resistant circuits. In much the same way, this Key is the basis for the physical values that are presented on the bus. Using a wider Key gives the encryption more complexity and more permutations that must be analyzed by an attacker. Apart from the width of the Key and complexity of the encryptor, the principal differences between the DS5000FP and DS5002FP are discussed below under Key Selection and Loading.

Encryption Key Selection and Loading

One of the significant differences between DS5000FP and DS5002FP lies in Encryption Key Management. In the case of a DS5000FP, the user must select a 40–bit Key during program loading. This Key must be selected prior to loading the microcontroller, as the memory will be encrypted as it is loaded. The Key selection process must be protected since an attacker that learns the Key can reproduce the user's code. This would be done by loading the correct Key in an unlocked DS5000FP, attaching the encrypted memory chip, and dumping the code using the Bootstrap Loader.

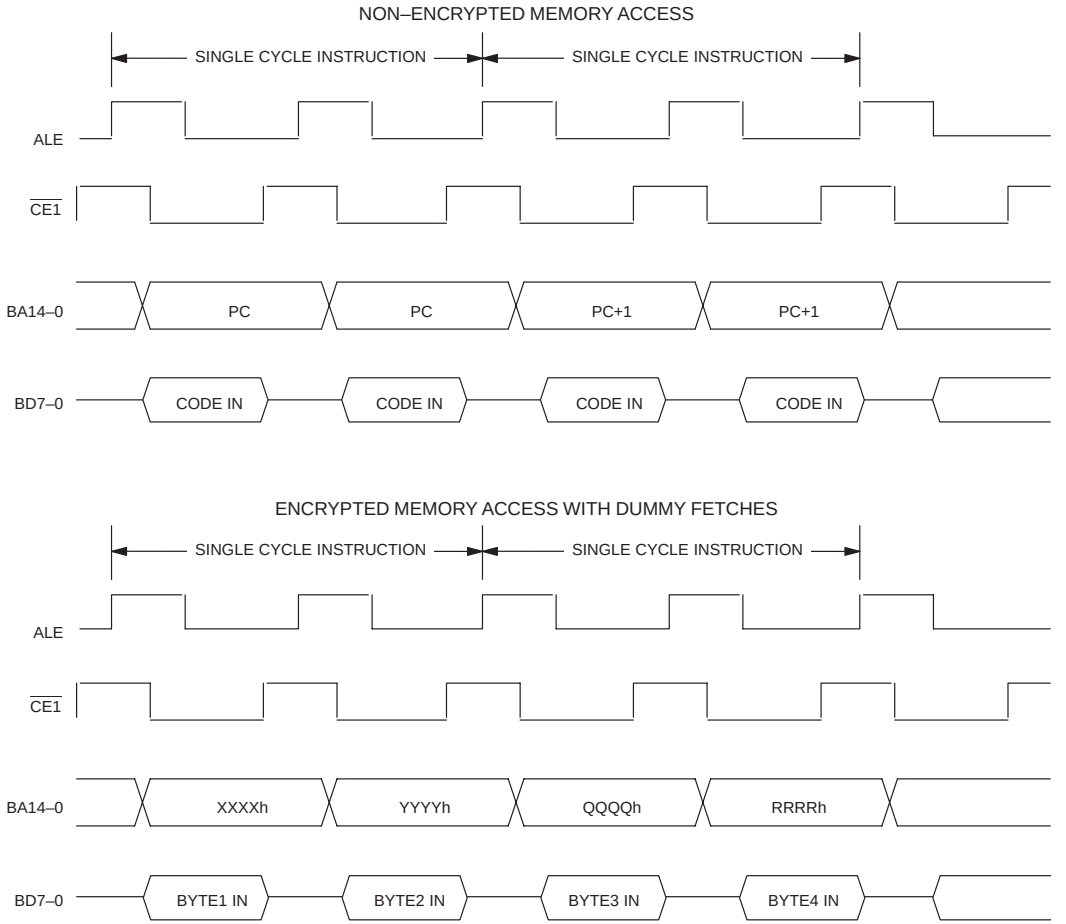
The DS5002FP provides an improved Key management system. The microcontroller chooses its own 64–bit Encryption Key from a number that is internally generated and secret. The Keys come from a true hardware random number generator. It is based on frequency differences between two on–chip ring oscillators and the user's crystal. At any time, it is unlikely that any two DS5002FPs have the same key with 2^{64} ($1.84 * 10^{19}$) combinations. There is no method to discover the Key value. No attacker can force the DS5002 to a particular Key. In addition, no one can "forget" to enable the encryptor, since it is always enabled. An additional advantage of the secret Key is that an attacker can not "characterize" the encryptor by repeatedly loading known Keys and observing the result.

As mentioned above, encryption is always enabled on the DS5002FP. Each time the Bootstrap Loader is invoked, a new random number is prepared. If a Fill, Load, Dump, Verify, or CRC command is requested, the Loader selects the random number as a new Encryption Key prior to accessing the memory. Execution of a Load or Fill command will result in a the data being loaded in an encrypted form determined by the value of the newly–generated Key. Any subsequent Dump, Verify, or CRC within the same Bootstrap session will cause the contents of the encrypted RAM to be read out and properly decrypted by the micro. Once a new Key is loaded, it will allow all commands to work properly within the same Bootstrap session since memory access is done using the correct Key. Exiting and re–entering the Bootstrap Loader, then doing a Dump will not work since this action would first result in Loading a new Encryption Key. The microcontroller would no longer be able to decrypt the RAM contents. This extra precaution is used regardless of the Security Lock. It prevents an attacker from retrieving memory through the Bootstrap Loader even if the programmer forgets to lock the DS5002FP. Once the Security Lock is set, all Bootstrap Loader access to the memory is prohibited.

Dummy Bus Access

The Secure Microcontroller makes its memory contents obscure through encryption. Additional steps are also to prevent analysis of the bus activity by 8051–familiar hackers. Both the DS5000FP and DS5002FP insert dummy memory operations when possible. In the 8051 architecture, there are typically two identical memory accesses per instruction cycle, but most operations so nothing with the second program fetch. In the Secure Microcontroller, a pseudo–random address is generated for the dummy cycle and this random memory address is actually fetched, but the dummy data is discarded. The order of the real and dummy accesses are switched according to a pseudo–random process. This is repeatable so that the execution always appears the same. During these pseudo–random cycles, the RAM is to all appearance read. Thus by repeatedly switching between real and dummy access, it is impossible to distinguish a dummy cycle from a real one. In analyzing bus activity, a large percentage of the memory fetches will be garbage that has no meaning. The dummy accesses are always performed on a DS5002FP, but are only used on a DS5000FP when encryption is enabled. Naturally, dummy accesses are always read operations since the dummy address might contain valid data.

DUMMY BUS ACCESS TIMING Figure 9–3



Either XXXX or YYYY is real but encrpted, the other is pseudo–random.
Either QQQQ or RRRR is real but encrypted, the other is pseudo–random.
Either Byte1 or Byte2 is used, the other is a dummy fetch and is not used. Both are encrypted.
Either Byte3 or Byte4 is used, the other is a dummy fetch and is not used. Both are encrypted.

On–chip Vector RAM

A 48–byte RAM area is incorporated inside the DS5000FP and DS5002FP. This area maps to the first 48 locations of program memory to store reset and interrupt vectors. Any other data stored in the first 48 locations will be contained in this Vector RAM. The principal reason for the Vector RAM is that the reset and interrupt vectors are known logical addresses in the 8051 family. Thus an attacker could force a reset or interrupt and discover the encrypted address generated by the Secure Microcontroller. By storing these Vectors in on–chip RAM, it is impossible to observe such relationships. Although it is very unlikely that an application program could be deciphered by observing the vector addresses, the Vector RAM eliminated this possibility. Note that the dummy accesses discussed above also occur while the Vector area is being accessed.

The Vector RAM is automatically loaded with the reset and interrupt vectors during Bootstrap Loading. This feature is transparent to operation and no action is required to use it. However, considering the Vector area feature can improve overall system security. As mentioned above, the Vector RAM is instantaneously destroyed in the event of an unlock (also by a self–destruct on DS5002FP). Since it is hidden and subject to destruction, the 48 bytes are the most secure memory in a system. Thus the most critical constants can also be stored there. This is an ideal location for storing DES keys for applications involving data encryption such as electronic funds transfer.

The Vector RAM is always used on a DS5002FP. The data stored between logical location 00h and 30h will be loaded into and executed for the Vector RAM. This data will not be duplicated in NV RAM accessed by the Byte–wide bus. The operation of DS5000FP Vector RAM is the same, but only when the encryption feature is enabled. When a DS5000FP has not had an Encryption Key loaded, the Vector RAM is left unused.

Self–Destruct Input

The Self–Destruct Input (SDI) is an active high input pin that is used to clear the security lock on a DS5002FP in response to an external event. The SDI is intended to be used with external tamper detection circuitry. It can be activated by an active high signal with or without operating power applied to the V_{CCI} pin. Activation of the SDI pin instantaneously clears the Security Lock initiating the sequence of events described above. In addition, power is momentarily removed from all Byte–wide bus interface signals including the V_{CCO} pin, resulting in loss of data by the external RAM. Address and data lines are also pulled low to remove any excess charge that could help retain data in that RAM. The SDI pin is deglitched so that a 2 µs pulse is required to activate it. However, this pin is sensitive so it should be grounded if not used. It is only available on the DS5002FP and DS2252FP products.

Microprobe/Die Top Coating

The DS5002FPM is provided with a special top–layer coating that is designed to prevent a microprobe attack. The coating is implemented with a second layer of metal on the microcontroller die. This metal will result in a short circuit of critical functions if probing is attempted. The probing action destroys the data that is secret. Also, security circuits and Vector RAM derive their power from this screen. Therefore they will be de–powered if the top coating is removed, also destroying the secret data. In this event, any critical data stored on–chip will be destroyed and off–chip data is rendered useless.

Random Number Generator

As mentioned above, the DS5002FP incorporates a hardware random number generator used by the Bootstrap Loader to generate Encryption Keys. The Random Number Generator is not a security circuit perse, but it is available to the application and can be used to improve the overall system security. Random numbers have numerous applications with respect to security. For example, to prevent an attacker from developing a histogram of code execution, the Random Number Generator could be used to decide how long to spend on particular activities. The random number is created 8 bits at a time. They are obtained by the application code at SFR location 0CFh. The random number takes 160 µs to develop. Reading a byte from register 0CFh will start the generation of another random number. After the random number is read, another will be available approximately 160 µs later. The RNR bit (RPCTL.7; 0D8h) will be set to a logic 1 each time a new number is available. If the random number is read prior to RNR being set, the value will be 00.

Security Summary by Part

The preceding information outlined each of the security features. Their inclusion in various parts is shown in the table at the beginning of this chapter. For completeness, the following is a summary description of security features for each part in the Secure Microcontroller Family.

DS5000FP / DS5000(T) / DS2250(T)

The DS5000 is the second generation of a microcontroller with security. The first is an earlier version of DS5000 circa 1988, now obsolete. The DS5000 incorporates a combination of real–time memory encryption and Security Lock. The memory encryption is optional however. To invoke the encryption, the user must select a 48–bit Encryption Key using the Bootstrap Loader. A user then loads the memory which will be automatically encrypted using this Key. After the memory is loaded and verified, the DS5000 can be locked. Locking the micro prevents an attacker from using the Bootstrap Loader to decrypt and dump the memory contents. Unlocking the DS5000 destroys the Encryption Key and Vector RAM. Vector RAM is 48 bytes of secret storage on–chip. It is used to hold reset and interrupt vectors as well as any application values than must be hidden. In addition to encrypting the memory, the DS5000 generates dummy bus cycles to obscure the actual program flow. Dummy cycles appear to be actual memory fetches but are not actually used inside the microcontroller. Also fundamental to the security of a DS5000 is its basis on RAM. This allows all security features to be changed frequently. The strategy is that an attacker must spend a long time breaking into the DS5000, but the user can simply change system security at any time. Thus any stolen information has a very limited lifetime.

DS5001FP / DS2251T

The DS5001 is a newer product than the DS5000, but has less security. It is useful in systems that need a large memory, but that provide sufficient physical security for all needs. The DS5001 incorporates a Security Lock. This is used to prevent the Bootstrap Loader from dumping memory. Once locked, the Bootstrap Loader can not access the memory. Unlocking the DS5001 causes the Bootstrap Loader to write over the NV RAM. The RAM nature of the DS5001 product allows a user to vary security frequently and to manually destroy it if necessary.

DS5002FP / DS2252(T)

The DS5002 adopts the memory and I/O improvements of the DS5001 and improves on the security of the DS5000. It is a high security version of the DS5001. This device is intended for maximum security and has numerous improvements to the DS5000. The security is always enabled on a DS5002. Thus an attacker can not characterize the security and the user can not forget to enable the security. The DS5002 follows a similar scheme of memory encryption and Security Lock. The DS5002 encryptor is a superior algorithm using a 64–bit Encryption Key. In addition, the Key is managed by the DS5002. Using the Bootstrap Loader, each part generates a random number for its 64–bit Key prior to loading memory. Leaving and re–entering the Bootstrap loader causes the DS5002 to select a new number as a potential Key. Any subsequent memory access with the Loader causes the new Key to be installed. Like the DS5000, the DS5002 also uses dummy bus access and Vector RAM to further hide memory bus activity. The Security Lock of a DS5002 is similar in nature to the DS5000. Once locked, the DS5002 Bootstrap Loader does not have access to memory. Unlocking the DS5002 destroys the Encryption Key and Vector RAM. The NV RAM accessed by the Byte–wide bus is also manually erased under Bootstrap Loader control. The DS5002 provides an external method to clear the Security Lock using its Self–Destruct Input (SDI). This causes the erasure of the Key and Vector RAM and also removes power from the NV RAM. The DS5002FPM provides a internal metal microprobe shield to prevent microprobing of the die.

APPLICATION: ADVANCED SECURITY TECHNIQUES

The Secure Microcontroller family has been used for numerous applications requiring security. Different levels of security are required depending on the sensitivity of the application and the value of the protected information. As mentioned above, the goal of the microcontroller security is to make stealing the protected information more difficult than the information is worth. This task actually has two pieces. First, the Secure Microcontroller makes attack difficult. This is combined with the user's physical security to make information retrieval difficult. The second part is to make the protected information less valuable. To this end, the NV RAM nature allows a user to frequently alter the firmware based security aspects of the system. Thus if the critical information changes before the security can be broken, the information that is actually retrieved will be worthless.

To assess the security of a system, the total implementation must be examined. The DS5000FP or DS5002FP provide a high level of security, but the user's firmware can accidentally defeat some features. Below are a sampling of implementation issues that will make the DS5000FP or DS5002FP more difficult to crack. There are also suggestions on making a system more secure using external circuits.

Avoid Clear Text

The encryption algorithms used by DS5000FP or DS5002FP are generally adequate to prevent analysis when combined with well developed code. However, the encryption is defeated to some extent if the user stores text that appears on a display in encrypted form. This gives the pirate a starting point to look for the clear text in encrypted storage and analyze the encryption algorithm. The "data answer" is already known. If clear text is required, then preferably store it in nonencrypted memory. If this is impractical, then disperse it so that it is hard to find. Avoid at all costs reading the clear text from memory then immediately displaying it. This is a sure means to identify the encrypted values of the text for the attacker.

Avoid CRC or Checksum

Running a checksum on power up provides the pirate with a sequential listing of the addresses in encrypted form. Therefore the attacker has a great advantage in deciphering the Address Encryptor. Preferably avoid a checksum. If one is needed, then check the minimum amount of memory and perform the check in non–sequential fashion.

Avoid Long Straight Runs of Code

A common coding practice is to run numerous sequential operations. This is common knowledge and should be avoided. The pirate can use this in the same way as a checksum process. It provides a sequential listing of encrypted addresses and assists with analysis of the address encryption.

Use Jumps

To address the prior problem, jumps are advised. These can be jumps for no reason other than to space out straight runs of code. However, using jumps also provides several other techniques to make bus analysis more difficult. As an example, the code can jump into Vector RAM. While in this area, dummy access will occur on the bus.

Use Random values

The Random Number Generator of the DS5002FP can be used to make a pirate's task more difficult. When time is available, the software should perform random actions at random time intervals. As an example, the Random Number Generator can be used to select a timer interrupt value. Thus the microprocessor will be interrupted at random intervals making characterization very difficult. Software can elect to out of Vector RAM for a random period of time. Also as discussed above, the microprocessor generates dummy RAM reads when possible. However, it can not generate dummy writes. However the user's code can. Random numbers can be written to address that are known to be unused. If this is done while the microprocessor is visibly performing a meaningful task, it will make analysis very difficult.

Vector RAM

As mentioned above, the Vector RAM can be used for many things beside vectors. This is the most secure storage in the system. It resides on–chip behind tamper protection. Thus it is useful for storing the most sensitive data. Thus even an attacker could break the encryption, this information would still be secret. For EFT or similar applications, this is a good location for the storage of DES keys. Since DES is a public algorithm, the real protection is keeping the DES key secret. As this is only 8 bytes, it fits well within the Vector RAM.

Change Code

Perhaps most importantly, the user should reprogram portions of the Secure Microcontroller that deal with security. For example, if the microprocessor is performing DES, the user can change DES keys. Any security system can be broken with enough time and resources. By altering the security features, this threat can be minimized.

External Circuits

A variety of external circuits can support secure operation. For example, the DS2400 is a unique 48–bit Silicon Serial Number. If it is installed with the microprocessor, it can be read when the system is first powered up, then stored inside the Secure Microcontroller. This serializes the system. If the software ever finds a different serial number (or missing number) from the stored one, it can refuse to work. This would mean that the microprocessor had been moved.

Tamper Protection

Using a variety of tamper sensors in conjunction with the DS5002 makes the system very difficult to crack. These circuits vary from simple switches to light, temperature, pressure, or oxygen sensors. When the physical security is violated, the SDI pin is activated and the memory contents are destroyed.